Copyright Statement

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

This thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognize the author's right to be identified as the author of this thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from their thesis.

General copyright and disclaimer

In addition to the above conditions, authors give their consent for the digital copy of their work to be used subject to the conditions specified on the Library Thesis Consent Form and Deposit Licence.
A Toolbox for Precise and Robust Deformation Measurement

Amir HajiRassouliha

Supervised by Prof. Poul Nielsen, Prof. Martyn Nash, and A/Prof. Andrew Taberner

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in Bioengineering, the University of Auckland

2017
Abstract

The measurement of shape and deformation of objects has many applications in various areas of computer vision. One of the most important applications of measuring deformation is in experimental mechanics, where it has been used to determine or assess mechanical properties of objects. The information gathered about the mechanical properties of objects can provide input data to inform or construct computational models, which can then be used to describe or predict the behaviour of the object. Digital image correlation (DIC) and digital volume correlation (DVC) are two main techniques used for measuring deformations. These techniques are widely used in bioengineering to identify the mechanical properties of human body tissues, such as skin, breast, bone, muscle, heart, and liver. The measured material properties are often then used to develop computational models of tissues.

The precision and robustness of these techniques are thus key factors in many applications. For instance, the accuracy of deformation measurements used to construct a computer model is an important measure of how well the model can describe the behaviour of the object.

Although there is great potential using DIC and DVC techniques in bioengineering applications, measuring the deformation of biological tissues is challenging. Human body tissues have a complex behaviour that varies significantly across different individuals, depending on factors, such as age, gender, and ethnicity. The precision and robustness of the techniques used to measure deformations are thus essential to addressing the needs of bioengineering applications.
Prior to this thesis, existing techniques had shortcomings and limitations that prevented their effective use in many applications, including bioengineering applications. For example, existing methods have significant problems with:

- estimating large shifts;
- measuring deformations from poorly-textured images;
- identifying parameters without a good initial estimate of the parameters;
- providing sufficient precision;
- performing fast computations; and
- calibrating more than two cameras in a multi-camera system in 3D DIC applications.

Addressing the abovementioned limitations is not simple. DIC and DVC techniques comprise several algorithms that together affect the precision and robustness of the measurements. Hence, the improvements need to be made at every step to improve the overall performance of each technique.

This thesis aims to tackle the problem of addressing the limitations and shortcomings of DIC and DVC techniques, and thereby to improve the overall performance and applicability of these techniques, particularly for bioengineering applications. This has been achieved by developing a set of novel methods and tools that significantly improve the precision and robustness of measurements. The main contributions of this thesis include:

1. developing a novel 2D subpixel image registration;
2. proposing a method to evaluate the performance of deformation measurements in practical applications and in the presence of noise;
3. tracking skin deformations using only intrinsic features;
4. performing motion correction in videos;
5. developing a novel method for subunit registration of arbitrary dimensional data;
6. developing a hand-held device for measuring skin deformations in vivo;
7. developing novel and accurate multi-camera calibration and 3D reconstruction methods; and
suggesting solutions to decrease the computation time of DIC and DVC techniques.

The developed methods have been comprehensively evaluated and tested throughout this thesis, where they demonstrated significant improvements over existing state-of-the-art algorithms. The high precision and robustness of the methods developed in this thesis enable the use of DIC and DVC techniques in many applications that were not previously possible or feasible. Even though the methods were developed specifically to address the limitations of DIC and DVC techniques, they could be used in many similar computer vision tasks that demand high precision or robustness.
Acknowledgments

Foremost, I would like to thank my main supervisor Prof. Poul Nielsen and my co-supervisors Prof. Martyn Nash and A/Prof. Andrew Taberner. They have all been incredibly generous with their time, devoting many hours to my work, guiding me, provide me with feedback, and reviewing and proofreading my papers and thesis.

I am very grateful to Prof. Poul Nielsen for patiently teaching me how to seek solutions to complicated scientific problems, develop my critical thinking skills, and target the highest possible quality in my work. He always had innovative ideas about how to solve problems or how to make improvements to methods. Poul never hesitated to give his time to answer my questions, or to discuss the technical details of my work. I would also like to express my thanks to Prof. Martyn Nash for his exceptional attention to all aspects of my work. His comments and feedback have always been exciting for me as well as useful to improve my work. I would like to thank Martyn for the important role that he had in the success of our National Science Challenge seed fund project. I would like to express my gratitude to A/Prof. Andrew Taberner for his practical, feasible, and helpful comments and suggestions. He helped me to remember the priorities of my project and not get lost in details. Andrew was always very supportive and caring. I would like to express my heartfelt thanks that he trusted my abilities when I was looking for a PhD position, recommended me to Poul and Martyn, and gave me the chance to study my PhD at the Auckland Bioengineering Institute (ABI).

I am very appreciative of the financial support provided to me during my PhD studies by the University of Auckland Foundation and the ABI.
Thanks also go to my dear friends in the ABI’s soft tissue group: Dr Thiranja Prasad Babarenda Gamage, Dr Matthew Parker, Samuel Richardson, Alex Dixon, and Dr Nikini Puhulwelle Gamage. I am thankful for your support and the good times we had together. I would also like to express my thanks to Dr Paul Roberts for his advice, Dr Ming Cheuk and Barbara Kmiecik for the constructive collaborations that we had, and Bahareh Madadkhahsalmasi for the memorable times we had together. I am also thankful to Professor John Windsor for giving me the opportunity to learn from clinicians during the Engineering in Clinical Residence (ECR) program.

I would like to thank the members of ABI administration team: Suman Nath and Kate Harsant, and the ABI finance team: Maria Fung, Ambita Balanon, and Ruchira Eriagama for helping me to sort out admin/finance forms and applications. I would also like to thank Stephen Flint and Nicola Johnston from UniServices for the time they spent with us on our patent applications and commercialisation of our technologies.

Finally and significantly, I want to express my gratitude to my lovely wife Farzaneh and my wonderful family. Farzaneh has always supported me and encouraged me through these years. She believed in my abilities, loved me, took care of me, and shared in my excitement and happiness when I achieved my goals. Special thanks go to my parents, Ali and Ferial for all they have done for me in my life. My Mum and my Dad always supported me, and devoted their lives to build a good future for me. I will always be grateful for the unconditional love you gifted me. Lastly, I would like to thank my brother Iman, and my sisters Ava and Ayeh for caring about me.
This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.


<table>
<thead>
<tr>
<th>Nature of contribution by PhD candidate</th>
<th>Design of experiments, developing codes, data analysis, and manuscript preparation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extent of contribution by PhD candidate (%)</td>
<td>90</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CO-AUTHORS</th>
<th>Nature of Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew J. Taberner</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
<tr>
<td>Martyn P. Nash</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
<tr>
<td>Poul M. F. Nielsen</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Certification by Co-Authors**

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate’s contribution to this work, and the nature of the contribution of each of the co-authors; and

- that the candidate wrote all or the majority of the text.

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amir HajiRassouliha</td>
<td></td>
<td>2017/3/30</td>
</tr>
<tr>
<td>Andrew J. Taberner</td>
<td></td>
<td>April 3, 2017</td>
</tr>
<tr>
<td>Martyn P. Nash</td>
<td></td>
<td>April 5, 2017</td>
</tr>
<tr>
<td>Poul M. F. Nielsen</td>
<td></td>
<td>2017-03-30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Last updated: 19 October 2015
Co-Authorship Form

This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. Please include one copy of this form for each co-authored work. Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

Chapter 3: "The Effect of Camera Settings on the Image Noise Level and Subpixel Deformation Measurements"

<table>
<thead>
<tr>
<th>Nature of contribution by PhD candidate</th>
<th>Design of experiments, developing codes, data analysis, and manuscript preparation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extent of contribution by PhD candidate (%)</td>
<td>90</td>
</tr>
</tbody>
</table>

CO-AUTHORS

<table>
<thead>
<tr>
<th>Name</th>
<th>Nature of Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samuel Richardson</td>
<td>Preparation of the LabView code for changing the camera settings</td>
</tr>
<tr>
<td>Andrew J. Taberner</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
<tr>
<td>Martyn P. Nash</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
<tr>
<td>Poul M. F. Nielsen</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
</tbody>
</table>

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate’s contribution to this work, and the nature of the contribution of each of the co-authors; and
- that the candidate wrote all or the majority of the text.

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amir Hajirassouliha</td>
<td></td>
<td>2017/3/30</td>
</tr>
<tr>
<td>Samuel P. Richardson</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Martyn P. Nash</td>
<td></td>
<td>April 5, 2017</td>
</tr>
<tr>
<td>Poul M. F. Nielsen</td>
<td></td>
<td>2017-03-30</td>
</tr>
<tr>
<td>Andrew J. Taberner</td>
<td></td>
<td>April 3, 2017</td>
</tr>
</tbody>
</table>

Last updated: 19 October 2015
This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

**Chapter 4: Subpixel Measurement of Living Skin Deformation Using Intrinsic Features. Computational Biomechanics of Medicine XI Workshop, MICCAI (2016).**

<table>
<thead>
<tr>
<th>Nature of contribution by PhD candidate</th>
<th>Design of experiments, developing codes, data analysis, and manuscript preparation</th>
</tr>
</thead>
</table>

| Extent of contribution by PhD candidate (%) | 90 |

## CO-AUTHORS

<table>
<thead>
<tr>
<th>Name</th>
<th>Nature of Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew J. Taberner</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
<tr>
<td>Martyn P. Nash</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
<tr>
<td>Poul M. F. Nielsen</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
</tbody>
</table>

**Certification by Co-Authors**

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- that the candidate wrote all or the majority of the text.

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amir HajiRassouliha</td>
<td></td>
<td>2017/3/30</td>
</tr>
<tr>
<td>Andrew J. Taberner</td>
<td></td>
<td>April 3, 2017</td>
</tr>
<tr>
<td>Martyn P. Nash</td>
<td></td>
<td>April 5, 2017</td>
</tr>
<tr>
<td>Poul M. F. Nielsen</td>
<td></td>
<td>2017-03-30</td>
</tr>
</tbody>
</table>
This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

**Chapter 5: Motion Correction Using Subpixel Image Registration. International Workshop on Reconstruction and Analysis of Moving Body Organs, MICCAL pp. 14–23 (2017).**

<table>
<thead>
<tr>
<th>Nature of contribution by PhD candidate</th>
<th>Design of experiments, developing codes, data analysis, and manuscript preparation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extent of contribution by PhD candidate (%)</td>
<td>90</td>
</tr>
</tbody>
</table>

**CO-AUTHORS**

<table>
<thead>
<tr>
<th>Name</th>
<th>Nature of Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew J. Taberner</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
<tr>
<td>Martyn P. Nash</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
<tr>
<td>Poul M. F. Nielsen</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
</tbody>
</table>

**Certification by Co-Authors**

The undersigned hereby certify that:
- the above statement correctly reflects the nature and extent of the PhD candidate’s contribution to this work, and the nature of the contribution of each of the co-authors; and
- that the candidate wrote all or the majority of the text.

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amir HajiRassoulia</td>
<td></td>
<td>2017/3/31</td>
</tr>
<tr>
<td>Andrew J. Taberner</td>
<td></td>
<td>April 3, 2017</td>
</tr>
<tr>
<td>Martyn P. Nash</td>
<td></td>
<td>April 5, 2017</td>
</tr>
<tr>
<td>Poul M. F. Nielsen</td>
<td></td>
<td>2017-03-30</td>
</tr>
</tbody>
</table>
Co-Authorship Form

This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

Chapter 6: "An Accurate and Fast Algorithm for Subunit Registration of Arbitrary Dimensional Data"

<table>
<thead>
<tr>
<th>Nature of contribution by PhD candidate</th>
<th>Co-developing the method, design of experiments, developing codes, data analysis, and manuscript preparation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extent of contribution by PhD candidate (%)</td>
<td>70</td>
</tr>
</tbody>
</table>

### CO-AUTHORS

<table>
<thead>
<tr>
<th>Name</th>
<th>Nature of Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew J. Taberner</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
<tr>
<td>Martyn P. Nash</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
<tr>
<td>Poul M. F. Nielsen</td>
<td>Co-developing the method, advice on concept, data analysis, and review of manuscript.</td>
</tr>
</tbody>
</table>

### Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- that the candidate wrote all or the majority of the text.

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amir HajiRassouliha</td>
<td></td>
<td>2017/3/30</td>
</tr>
<tr>
<td>Andrew J. Taberner</td>
<td></td>
<td>April 3, 2017</td>
</tr>
<tr>
<td>Martyn P. Nash</td>
<td></td>
<td>April 5, 2017</td>
</tr>
<tr>
<td>Poul M. F. Nielsen</td>
<td></td>
<td>2017-03-30</td>
</tr>
</tbody>
</table>

Last updated: 19 October 2015
This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. Please include one copy of this form for each co-authored work. Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.


<table>
<thead>
<tr>
<th>Nature of contribution by PhD candidate</th>
<th>Design of experiments, developing codes, data analysis, and manuscript preparation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extent of contribution by PhD candidate (%)</td>
<td>85</td>
</tr>
</tbody>
</table>

**CO-AUTHORS**

<table>
<thead>
<tr>
<th>Name</th>
<th>Nature of Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barbara Kmiecik</td>
<td>Designing the camera rig solidworks model</td>
</tr>
<tr>
<td>Andrew J. Taberner</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
<tr>
<td>Martyn P. Nash</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
<tr>
<td>Poul M. F. Nielsen</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
</tbody>
</table>

**Certification by Co-Authors**

The undersigned hereby certify that:
- the above statement correctly reflects the nature and extent of the PhD candidate’s contribution to this work, and the nature of the contribution of each of the co-authors; and
- that the candidate wrote all or the majority of the text.

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amir HajiRassouliha</td>
<td></td>
<td>2017/3/30</td>
</tr>
<tr>
<td>Barbara Kmiecik</td>
<td></td>
<td>April 3, 2017</td>
</tr>
<tr>
<td>Andrew J. Taberner</td>
<td></td>
<td>April 3, 2017</td>
</tr>
<tr>
<td>Martyn P. Nash</td>
<td></td>
<td>April 5, 2017</td>
</tr>
<tr>
<td>Poul M. F. Nielsen</td>
<td></td>
<td>2017-03-30</td>
</tr>
</tbody>
</table>
This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. Please include one copy of this form for each co-authored work. Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

Chapter 8: "Robust and Accurate Multiple Camera Stereographic Calibration"

<table>
<thead>
<tr>
<th>Nature of contribution by PhD candidate</th>
<th>Design of experiments, developing codes, data analysis, and manuscript preparation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extent of contribution by PhD candidate (%)</td>
<td>90</td>
</tr>
</tbody>
</table>

**CO-AUTHORS**

<table>
<thead>
<tr>
<th>Name</th>
<th>Nature of Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew J. Taberner</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
<tr>
<td>Martyn P. Nash</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
<tr>
<td>Poul M. F. Nielsen</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
</tbody>
</table>

**Certification by Co-Authors**

The undersigned hereby certify that:
- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- that the candidate wrote all or the majority of the text.

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amir Hajirassouliha</td>
<td></td>
<td>2017/3/30</td>
</tr>
<tr>
<td>Andrew J. Taberner</td>
<td></td>
<td>April 3, 2017</td>
</tr>
<tr>
<td>Martyn P. Nash</td>
<td></td>
<td>April 5, 2017</td>
</tr>
<tr>
<td>Poul M. F. Nielsen</td>
<td></td>
<td>2017-03-30</td>
</tr>
</tbody>
</table>

Last updated: 19 October 2015
This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

Chapter 9: "A Model-based Technique for Calibration of Multiple Cameras"

<table>
<thead>
<tr>
<th>Nature of contribution by PhD candidate</th>
<th>Design of experiments, developing codes, data analysis, and manuscript preparation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extent of contribution by PhD candidate (%)</td>
<td>90</td>
</tr>
</tbody>
</table>

**CO-AUTHORS**

<table>
<thead>
<tr>
<th>Name</th>
<th>Nature of Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew J. Taberner</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
<tr>
<td>Martyn P. Nash</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
<tr>
<td>Poul M. F. Nielsen</td>
<td>Advice on concept, data analysis, and review of manuscript.</td>
</tr>
</tbody>
</table>

**Certification by Co.Authors**

The undersigned hereby certify that:
- the above statement correctly reflects the nature and extent of the PhD candidate’s contribution to this work, and the nature of the contribution of each of the co-authors; and
- that the candidate wrote all or the majority of the text.

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amir HajiRassouliha</td>
<td></td>
<td>2017/3/30</td>
</tr>
<tr>
<td>Andrew J. Taberner</td>
<td></td>
<td>April 3, 2017</td>
</tr>
<tr>
<td>Martyn P. Nash</td>
<td></td>
<td>April 5, 2017</td>
</tr>
<tr>
<td>Poul M. F. Nielsen</td>
<td></td>
<td>2017-03-30</td>
</tr>
</tbody>
</table>
Abstract.............................................................................................................................................i

Acknowledgments ...............................................................................................................................v

1 Introduction........................................................................................................................................1

1.1 Motivation .......................................................................................................................................1

1.2 Objectives ......................................................................................................................................4

1.3 Thesis overview and novel contributions .......................................................................................4

2 Subpixel image registration ..............................................................................................................9

2.1 Abstract .........................................................................................................................................9

2.2 Introduction ..................................................................................................................................10

2.3 Method .........................................................................................................................................15

1.1.1 Finding the integer pixel translation shift ..............................................................................15

1.1.2 Finding the translational subpixel shift ..................................................................................20

1.1.3 Ground truth data and comparison metrics ...........................................................................22

2.4 Results and discussion ..................................................................................................................27

1.1.4 Translational shifts ..................................................................................................................27

1.1.5 Robustness to noise and evaluation of the proposed algorithm with error metrics 29

1.1.6 Image rotation tests and evaluation of the proposed algorithm with error metrics 34

1.1.7 Computational complexity ......................................................................................................39

2.5 Conclusions ..................................................................................................................................40

3 The Effect of Camera Settings on the Image Noise Level and Subpixel Deformation Measurements ..........................................................................................................................45

3.1 Abstract .......................................................................................................................................45

3.2 Introduction ...................................................................................................................................46
6.3.2 Estimating subunit shifts .................................................................103
6.3.3 Generating synthetic 2D/3D images ..................................................105
6.3.4 Applying synthetic shifts .................................................................105
6.3.5 Adding Gaussian noise and measuring estimation errors .................105

6.4 Results and discussion ........................................................................107
6.4.1 Estimating 2D integer shifts and robustness to Gaussian noise ..........108
6.4.2 Estimating 3D integer shifts and robustness to Gaussian noise ..........111
6.4.3 Estimating 3D subvoxel shifts and robustness to Gaussian noise ........115

6.5 Summary .............................................................................................119

7 A Low-cost, Hand-held Stereoscopic Device for Measuring Deformations of Skin In Vivo 123

7.1 Abstract .............................................................................................123
7.2 Introduction .........................................................................................124
7.3 Camera and lens selection .....................................................................125
  7.3.1 Selecting the camera bus interface ..................................................125
  7.3.2 Selecting the camera and lens .........................................................129
7.4 Camera rig design ................................................................................129
  7.4.1 Determining the cameras positions ...............................................130
  7.4.2 Design of the camera rig ...............................................................131
  7.4.3 Mass ..............................................................................................134
7.5 Camera settings ..................................................................................135
  7.5.1 Depth of field (DOF) .....................................................................135
  7.5.2 Illumination ...................................................................................135
  7.5.3 Capturing image data and synchronising the cameras .................136
8 Robust and Accurate Multiple Cameras Stereographic Calibration

8.1 Abstract

8.2 Introduction

8.3 Theory and principles

8.3.1 The pinhole camera model

8.3.2 Lens distortion model

8.3.3 Calibration of intrinsic camera parameters

8.3.4 Calibration of extrinsic camera parameters

8.4 Method

8.4.1 Finding the initial values of the camera parameters

8.4.2 Optimising the intrinsic and extrinsic camera parameters

8.4.3 Validation of the results

8.4.4 Performance analysis

8.5 Experimental setup

8.6 Results and discussion

8.7 Summary

9 A Model-based Technique for Calibration of Multiple Cameras

9.1 Abstract

9.2 Introduction

9.3 Experimental setup and calibration target

9.4 Calibration method

9.4.1 Estimating initial values of the parameters of the multi-camera system
9.4.2 Refining the parameters and recalibrating the cameras ...........................................187
9.4.3 Experiments performed to demonstrate the accuracy of the estimated parameters ........................................................................................................................................194
9.5 Results and discussion ..............................................................................................................................196
  9.5.1 Estimating the initial values of the parameters of the multi-camera system 196
  9.5.2 Refining the parameters and recalibrating the cameras ......................................................197
  9.5.3 Experiments performed to demonstrate the accuracy of the estimated parameters ...........................................................................................................................................204
9.6 Summary ................................................................................................................................................209
10 Conclusions and future work ......................................................................................................................211
  10.1 Summary of the main findings and contributions ..............................................................................212
  10.2 Future work .........................................................................................................................................215
  10.3 Thesis outputs .....................................................................................................................................218
    10.3.1 Journal publications ......................................................................................................................218
    10.3.2 Conference papers and abstracts ..............................................................................................219
    10.3.3 Awards .........................................................................................................................................221
    10.3.4 Patents .........................................................................................................................................221
    10.3.5 Research grants ...........................................................................................................................221
    10.3.6 Invited presentations ....................................................................................................................222
Appendix A: Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms ..................................................................................................................223
  A.1 Abstract ...............................................................................................................................................223
  A.2 Introduction .........................................................................................................................................224
  A.3 Digital signal processors (DSPs) .......................................................................................................228
    A.3.1 Available development tools and utilities for DSPs ....................................................................230
A.3.2 Embedded operating systems .......................................................... 231
A.3.3 Development time ........................................................................... 231
A.3.4 Advantages of using DSPs ................................................................. 232
A.3.5 Disadvantages of using DSPs .............................................................. 233
A.3.6 Review of applications that use DSPs ............................................... 233
A.3.7 Summary and conclusion for DSPs .................................................... 236
A.4 Field-programmable gate arrays (FPGAs) ............................................ 236
  A.4.1 Xilinx FPGA families ..................................................................... 237
  A.4.2 Altera FPGA families .................................................................... 243
  A.4.3 Development time .......................................................................... 244
  A.4.4 Advantages of using FPGAs .............................................................. 245
  A.4.5 Disadvantages of using FPGAs ......................................................... 246
  A.4.6 Review of applications that use FPGAs ............................................ 246
  A.4.7 Summary and conclusion for FPGAs ............................................... 251
A.5 Graphics processing units (GPUs) ....................................................... 252
  A.5.1 NVidia GPU series ...................................................................... 252
  A.5.2 Tools and utilities for NVidia GPUs ................................................. 254
  A.5.3 Development time .......................................................................... 256
  A.5.4 Advantages of using GPUs .............................................................. 256
  A.5.5 Disadvantages of using GPUs .......................................................... 257
  A.5.6 Review of applications that use GPUs ............................................ 258
  A.5.7 Summary and conclusion for GPUs ............................................... 263
A.6 Comparison of FPGAs and GPU for implementing image processing, and
  computer vision algorithms ..................................................................... 265
A.7  Summary and conclusions.................................................................270

Appendix B:  FPGA Implementation of 2D Cross-Correlation.........................273

B.1  Abstract..........................................................................................273
B.2  Introduction .....................................................................................274
B.3  Method.............................................................................................275
  B.3.1  Loading and initialisation.............................................................278
  B.3.2  Synchronisation..........................................................................278
  B.3.3  Calculating the CC value..............................................................280
B.4  Results.............................................................................................283
B.5  Discussion and conclusion.................................................................286
Figure 2-1: The frequency response of a 7 point cubic first derivative SGD in comparison to first and second order central differences, and the ideal differentiator.

Figure 2-2: The power spectrum (20 log10(abs(FFT(I)))) (dB scale) of a sample image (LANDSAT image of Paris in Figure 2-5) (a) for the original image (b) with addition of a small amount of white Gaussian noise (variance of 0.001 of the normalised intensity values) to the image. The zero frequency (i.e. DC component) is shifted to the centre; hence higher frequencies are close to the peripheral parts. The white Gaussian noise has spread in higher frequencies in Figure 2-2(b), in comparison to fine details of the image in Figure 2-2(a).

Figure 2-3: The frequency response of Hamming, Hann, Blackman, and Tukey ($\alpha = 0.25$) window functions.

Figure 2-4: The diagram of the normalisation procedure.

Figure 2-5: Standard LANDSAT images used in this study. Images contrasts have been enhanced for presentation purposes, but all of the analyses in this study were performed on original images.

Figure 2-6: (a) The central points of subimages on the LANDSAT image of Paris. (b) A sample subimage (128 pixel × 128 pixel).

Figure 2-7: The displacement vectors measured at each subimage of the LANDSAT image of Paris (Figure 2-6) between the original image and synthetically shifted image using the proposed method.

Figure 2-8: The average registration errors (pixel) in estimating the shift [3.250, 4.750] in 400 subimages of the LANDSAT image of Paris (128 pixel × 128 pixel). The PSNR of the original and unshifted noisy images was from 49.46 dB to 33.00 dB, and the SSIM index was from 0.9949 to 0.8223 in this range of noise.

Figure 2-9: The average registration errors (pixel) in estimating the shift [3.250, 4.750] in 400 subimages of the LANDSAT image of Paris (128 pixel × 128 pixel). The PSNR of the original and unshifted noisy images was from 61.17 dB to 37.55 dB, and the SSIM index was from 0.9999 to 0.9853 in this range of noise.

Figure 2-10: The average registration errors (pixel) in estimating the shift [3.250, 4.750] in 400 subimages of the LANDSAT image of Paris (128 pixel × 128 pixel). The PSNR of the
original and unshifted noisy images was from 23.03 dB to 11.29 dB, and the SSIM index was from 0.3452 to 0.0330 in this range of noise.

Figure 2-11: The average registration errors (pixel) in estimating the shift [3.250, 4.750] in 400 subimages of the LANDSAT image of Paris (128 pixel × 128 pixel). The PSNR of the original and unshifted noisy images was from 27.91 dB to 14.09 dB, and the SSIM index was from 0.8763 to 0.1075 in this range of noise.

Figure 2-12: The LANDSAT image of Paris. (a) The original image (contrast has been enhanced in images for presentation purposes). (b) with Gaussian noise ($\sigma^2 = 0.120$) added to that (PSNR = 11.29 dB, SSIM = 0.0330) (c) with salt and pepper noise ($d = 0.120$) added to that (PSNR = 14.09 dB, SSIM = 0.1075). The average registration error of the proposed algorithm in estimating the [$x, y$] shift [3.250, 4.750], over 400 subimages of the noisy images that contained Gaussian noise (b) and salt and pepper noise (c) was 0.85 and 0.25 pixel, respectively.

Figure 2-13: (a) The average registration errors (pixel) in estimating the shift [3.250, 4.750] in 400 subimages of the LANDSAT image of Paris (128 pixel × 128 pixel) for the proposed method (b) The average number of points with normalised SGGC values (Equation 2.9) greater than 0.85 as the error metric of the integer part of the proposed algorithm. (c) The plane fitting error to $\mathbf{φp}$ data (Equation 2.11) as the error metric of the subpixel part of the proposed algorithm.

Figure 2-14: Displacement vectors determined using the proposed algorithm for the $1^\circ$ rotation about the centre of the LANDSAT image of Mississippi.

Figure 2-15: (a) The ideal rotation pattern of $1^\circ$ rotation for the same size image. Rotation patterns were found by using the proposed method b), Foroosh + HW (c), Forooosh et al. [49] (d), Guizar-Sicairos et al. [55] (e), Stone et al. [58] (f), Vandewalle et al. [62] (g), Feng et al. [61] (h), Brox et al. [60] (i), and Messerli et al. [53] (j). The average error in this image was 0.0834 pixel for the proposed method (b), 0.1322 pixel for Foroosh + HW (c), 0.3068 pixel for Foroosh et al. [49] (d), 0.2412 pixel for Guizar-Sicairos et al. [55] (e), 0.9269 pixel for Stone et al. [58] (f), 2.2078 pixel for Vandewalle et al. [62] (g), 0.2751 pixel for Feng et al. [61] (h), 0.8999 pixel for Brox et al. [60] (i), and 0.1060 pixel for Messerli et al. [53] (j).
Figure 3-1: The experimental setup of this study. The speckled sample was uniformly illuminated with four high power LEDs.

Figure 3-2: The distribution of subimage centres over the speckled surface of the test sample. The subimage size was selected to be 128 pixel × 128 pixel, and the distance between adjacent subimage centres was 20 pixel. A sample subimage is indicated on the surface of the test sample.

Figure 3-3: The noise variance of images taken with the lens cap on (experiment E1 described in Section 3.5).

Figure 3-4: The noise variance of images taken with the lens cap off (experiment E2). For gain values higher than 0 dB, the exposure time was increased until the image intensity became saturated in parts of the image.

Figure 3-5: The noise variance of images taken with the lens cap off, and with an exposure time of up to 16.3 ms, at a constant gain of 0 dB. The image noise level had small variations for exposure times above 6 ms.

Figure 3-6: The noise variance of images taken at a mean pixel intensity of approximately 110 (experiment E3), and exposure times ranging from 19 ms to 62 ms. 20 images were taken at each exposure time.

Figure 3-7: Registration error errors for different algorithms that estimate subpixel shifts for images taken at gain 0 dB, and a mean pixel intensity of approximately 110 (experiment E3).

Figure 3-8: The image noise level at combinations of exposure times and gain values that provided mean pixel intensity of approximately 110 in images (experiment E4). The exposure times and gain values were ranging from 2.57 ms to 18.80 ms, and 0 dB to 18 dB, respectively.

Figure 3-9: The registration error for subpixel shift measurements from images without noise, images with a noise variance of 0.92 (at 0 dB gain, and the exposure time of 18.80 ms), and images with a noise variance of 5.44 (at 18 dB gain, and the exposure time of 2.57 ms). The mean pixel intensity was approximately 110 in all of the images.
Figure 3-10: The registration error of subpixel shift measurements for combinations of exposure times and gain values that provided mean pixel intensity of approximately 110 in images (experiment E4). The exposure times and gain values were ranging from 2.57 ms to 18.80 ms, and 0 dB to 18 dB, respectively. The noise levels of these images were increased from 0.92 to 5.44 (see Figure 3-8).

Figure 4-1: The experimental setup used to evaluate the accuracy of P-SG-GC in measuring translational shifts. A photographic stand was used to hold the camera in a position perpendicular to the surface of a flat object attached to a linear translational stage.

Figure 4-2: The \( (Dn - MD) \) values measured using the P-SG-GC algorithm applied to the original and undistorted images. The values are colour-coded and are in units of pixel. The target was moved downward using the linear translational stage, and the shift magnitudes are indicated on the left.

Figure 4-3: The displacement map caused by the lens distortion model on the same area as the images in Figure 4-2. The displacement values were estimated by applying the P-SG-GC algorithm to one of the images and its undistorted version. Displacement magnitudes are colour-coded in a logarithmic scale and values are in units of pixel.

Figure 4-4: The linearity between the MD values estimated using the P-SG-GC algorithm, and the translational shifts of the target using the linear translational stage.

Figure 4-5: The standard deviation of displacements estimated for subimages of the original (distorted) and undistorted images using the P-SG-GC algorithm.

Figure 4-6: The images of a volunteer’s hand used in this study to test the capability of the P-SG-GC algorithm in finding the skin displacements using intrinsic features. The volunteer was asked to move their thumb in Figure 4-6(b) to deform the skin of the back of the hand.

Figure 4-7: The estimation of the skin deformation from the images of Figure 4-6 using the P-SG-GC algorithm and 64 pixel \( \times \) 64 pixel subimages. The magnitude of the displacements is colour-coded and the arrows show the directions of displacement.

Figure 5-1: The experimental setup used to test the accuracy of the P-SG-GC algorithm to correct motion artefacts.
Figure 5-2: The results of the translational shift in experiment E1. The final image for this experiment (5 mm shift of the target) is overlaid on the reference image without (a) and with (b) motion correction. In the overlaid images in (a) and (b), red colour was used for the reference image, green colour was used for the shifted image, and yellow colour was used for areas of similar intensity between the two images. The subtraction of the intensity values for the reference and final images without and with motion correction is shown in (c) and (d), respectively, and the values are colour-coded. The registration error is shown in (e). .................89

Figure 5-3: The results of the rotational tests in experiment E2 involving a 3° rotation of the target. See Figure 5-2 caption for explanation of the panels. .........................................................89

Figure 5-4: The results of the translational and rotational tests in experiment E3 involving a 5 mm translation and 1.25° rotation of the target. See Figure 5-2 caption for explanation of the panels. The P-SG-GC algorithm detected that the proportion of matched control points with an acceptable error was insufficient for correcting the motion in the final image of this experiment (i.e. 5 mm translation and 1.25° rotation of the target). For this reason, the result does not include the final image.........................90

Figure 5-5: The results of the camera shifts in experiment E4 involving a 10 mm shift of the camera. See Figure 5-2 caption for explanation of the panels.................................90

Figure 5-6: Results of the camera and translational shifts in experiment E5 involving a 5 mm shift of the camera and 2.5 mm shift of the target. See Figure 5-2 caption for explanation of the panels..........................................................91

Figure 6-1: The schematic of the integer part of the FCC algorithm. The squared \( \omega \)-domain N-D filter is the novel part of this algorithm that allows arbitrarily combining filters in the spatial and frequency domains (the box on the right). The squared operation in the frequency domain is equivalent to applying filters independently to each of the N-D subset datasets.102

Figure 6-2: The subunit part of the PGreg method.................................................................104

Figure 6-3: The Landsat image of Paris [116].................................................................107

Figure 6-4: Three samples of synthetically generated test 2D/3D images. The intensity values of the test images were uniformly distributed pseudorandom integers. The intensity
ranges were selected as 32, 64, and 128 intensity values centred around 127 to avoid intensity saturations as Gaussian noise was added.

Figure 6-5: The failure rate (Equation 6.2 and Equation 6.3) of the integer parts of the SGD-GC algorithm [35] and two options for the FCC algorithm in 2D images (i.e. one 2D Savitzky-Golay filter and two 1D Savitzky-Golay filters). On the left are the tests results for the standard Landsat image of Paris and on the right are the tests results for the synthetically generated I1s (refer to Figure 6-4 for the description of an I1). The top, middle and bottom rows show the results for subimage sizes of (16 \times 16) \text{pixel}^2, (32 \times 32) \text{pixel}^2, and (64 \times 64) \text{pixel}^2, respectively.

Figure 6-6: Average failure rates of the SGD-GC algorithm [35] and two options for the FCC algorithm in 2D images (i.e. one 2D Savitzky-Golay filter and two 1D Savitzky-Golay filters) at Gaussian noises with standard deviations from 0 to 50 added to the Landsat image of Paris and synthetically generated I1s, I2s, and I3s.

Figure 6-7: The failure rates of three algorithms calculated at (16 \times 16 \times 16) \text{voxel}^3, (32 \times 32 \times 32) \text{voxel}^3, and (64 \times 64 \times 64) \text{voxel}^3 subimages of synthetically generated V1s (refer to Section 6.3.3 and Figure 6-4 for the description of a V1).

Figure 6-8: A summary of the average failure rates of the algorithms at Gaussian noises with standard deviations from 0 to 50 added to subimages of synthetically generated V1s, V2s, and V3s.

Figure 6-9: The failure rates of 3D-CC calculated in noise-free V1s, V2s, and V3s.

Figure 6-10: The average error of the PGreg algorithm with three 1D filters in estimating random shifts.

Figure 6-11: The mean and standard deviation of bias errors of the PGreg algorithm in estimating subvoxel shifts at subimages of V3s.

Figure 6-12: The effects of the addition of the phase unwrapping and the selection of the phase values with phase error within the range of \([-\pi/2, \pi/2]\) in the subvoxel part of the PGreg algorithm tested at (16 \times 16 \times 16) \text{voxel}^3 subimages of V1s.

Figure 1-1: The positions of the focal points of an opposing pair of FL3-U3-13Y3M-C cameras. This arrangement provided a 200 mm field of view with a 6 mm lens (angle of view...
of 44.54°). The axes of the cameras intersect at an angle of 90° at the centre of the field of view.

Figure 1-2: The design of the stereoscopic device. Four cameras were placed in a ring configuration in two opposite pairs. The heights and the distances between the cameras were chosen based on the calculations in Section 7.4.1.

Figure 1-3: The top view (a) and the bottom view (b) of the central plate of the design. The curved plates were fixed and glued to this central plate in an inter-locking manner. It was also possible to screw the curved plates to the central plate.

Figure 1-4: The side view (a) and bottom view (b) of the camera attachments to the curved plates of the camera rig.

Figure 1-5: The side view (a) and bottom view (b) of the fields of view of each of the four cameras. The overlapping FOVs of the cameras are approximately inside the polygon shown in the image (b).

Figure 1-6: The designed hand-held stereoscopic device for in vivo measurement of skin surface deformations.

Figure 1-7: Images of a hand from each of the four cameras of the designed hand-held stereoscopic device. The camera images spanned the whole surface of the hand from each of the four viewing angles.

Figure 1-8: A sample 2D displacement measurement from the surface of the skin using two sequential images of a hand from one of the cameras of the hand-held device.
Figure 7-6: The average fitting error for fitting lines to rows of checkerboard corners in a test dataset consisting of 50 calibration images. The errors were calculated for distorted images, and undistorted images using parameters estimated from 20 calibration images and 100 calibration images.

Figure 7-7: An epipolar line in one the cameras (right) and its corresponding point in the second camera (left).

Figure 7-8: A sample epipolar line (right) and its corresponding point (left) found using initial camera parameters.

Figure 7-9: A sample epipolar line (right) and its corresponding point (left) found using the estimated parameters from 20 calibration images.

Figure 7-10: A sample epipolar line (right) and its corresponding point (left) found using the estimated parameters from 100 calibration images.

Figure 7-11: The RMS reprojection errors, the average length error, and 3D RMS error for a dataset consisting of 100 images for the proposed method compared to the traditional method of calibrating multiple cameras using the LM algorithm.

Figure 7-12: The RMS reprojection errors, the average length error, and 3D RMS error obtained using the proposed method (orange plots) or the traditional LM method (blue plots) with a randomly perturbed set of initial camera parameters. Green plots show the errors for initial camera parameters without the perturbations.

Figure 7-13: The RMS reprojection errors, the average length errors, and 3D RMS errors estimated for the proposed method, case I (i.e. 3D error functions were excluded from the objective function), and case II (i.e. 3D error functions were included in the objective function, but the camera parameters and errors were unweighted).

Figure 7-14: The RMS length error calculated using the parameters estimated by the proposed method, case I, and case II for a test dataset of 50 calibration images (refer to the legend of Figure 8-13 for the descriptions of case I and case II).

Figure 8-1: The stereoscopic system used for multi-camera calibration.

Figure 8-2: The calibration template used in this study. The template was designed in SolidWorks and was converted to a scalable vector graphics (SVG) image.
Figure 8-3: A sample image of the concentric calibration template recorded using one of the cameras of the multi-camera system.

Figure 8-4: (a) An example of applying the Canny edge detection algorithm to a calibration image. (b) The background was removed using the first step of the method described in Section 9.4.2.1.

Figure 8-5: An example of using the second step of the method described in Section 9.4.2.1 to remove the remaining parts of the background.

Figure 8-6: A sample result of removing all parts of the background from calibration images. The concentric circles were segmented for use in the ellipse fitting process.

Figure 8-7: Three samples of the ellipse fitting results. The least-squares method described in the third step of Section 9.4.2.1 was used to find the best-fit ellipses to the segmented concentric circles. The centres of the ellipses are shown with red crosses.

Figure 8-8: A sample result of localising the control points of the calibration template.

Figure 8-9: A sample result of mapped control points in one calibration image of all of the four cameras of the stereoscopic system. The lines connect the control points of the calibration template.

Figure 8-10: A sample calibration image (top left) and the generated model of the calibration template using steps one to four of the method described in Section 9.4.2.2 (top right). Zoomed views of the concentric circles of the calibration template in the calibration image and the model are shown below each image.

Figure 8-11: Two samples of 128 pixel × 128 pixel subimages of the calibration image (left) and the reconstructed model of the calibration template (right). The subpixel shifts between the subimages were measured using the P-SG-GC algorithm.

Figure 8-12: A sample image of the flat object attached to a linear stage in the FOV of one of the cameras of the stereoscopic system.

Figure 8-13: A virtual grid of points was selected on the surface of the flat object in the reference camera (green dots in the top left image). The grid size was 24 points × 39 points (963 points) with a step size of 10 pixels. The corresponding points were found in the other
three cameras using the method described in Section 9.4.3 (red dots in the top right and two bottom images). The images were cropped and zoomed for visualisation purposes.

Figure 8-14: The measurement errors in estimating 3D shifts of the flat object using the proposed method, the method of HajiRassouliha et al. [38], and the conventional method, for 32 pixel × 32 pixel subimages (a) and 64 pixel × 64 pixel subimages (b). Error bars show the STD of the 3D measurements. Refer to Table 9-4 for the average magnitude of 2D displacements for each 3D shift.

Figure A-1: Search trends for GPU (in blue colour) and FPGA (in orange colour).

Figure B-2: The implemented CC algorithm for two images. Dashed lines show the subimage and solid lines show the initial image. The first position of the algorithm was where the top of the two images were aligned, and the last position was where they were aligned at the bottom.

Figure B-3: Calculation of the CC value for each relative position of the subset and the initial images.

Figure B-4: Concatenating the intensity values for 8 pixels to form a 64 bit digit.

Figure B-5: The flow diagram of the procedure for synchronisation between reading the image data of two subimages.

Figure B-6: The multiplication unit and the accumulator for calculating CC values.

Figure B-7: Using registers to synchronise the output data and the data valid signal.

Figure B-8: Two sample neighbour slices in the hierarchical approach.

Figure B-9: Part of the final result. The data and data valid signal were synchronised, and the CC value was shifted up for each new row.

Figure B-10: Utilisation report of the proposed algorithm in the Virtex-6 (LX240T).
## Abbreviation

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>Two-Dimensional</td>
</tr>
<tr>
<td>3D</td>
<td>Three-Dimensional</td>
</tr>
<tr>
<td>AOV</td>
<td>Angle of View</td>
</tr>
<tr>
<td>CC</td>
<td>Cross-Correlation</td>
</tr>
<tr>
<td>CCD</td>
<td>Charge-Coupled Devices</td>
</tr>
<tr>
<td>CMOS</td>
<td>Complementary Metal–Oxide Semiconductor</td>
</tr>
<tr>
<td>CT</td>
<td>Computed Tomography</td>
</tr>
<tr>
<td>DFT</td>
<td>Discrete Fourier Transform</td>
</tr>
<tr>
<td>DIC</td>
<td>Digital Image Correlation</td>
</tr>
<tr>
<td>DOF</td>
<td>Depth of Field</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processors</td>
</tr>
<tr>
<td>DVC</td>
<td>Digital Volume Correlation</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>FOV</td>
<td>Field of View</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field-Programmable Gate Array</td>
</tr>
<tr>
<td>GB</td>
<td>Gradient-Based</td>
</tr>
<tr>
<td>GC</td>
<td>Gradient-Correlation</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>LM</td>
<td>Levenberg-Marquardt</td>
</tr>
<tr>
<td>MR</td>
<td>Magnetic Resonance</td>
</tr>
<tr>
<td>NCC</td>
<td>Normalised Cross-Correlation</td>
</tr>
<tr>
<td>NGC</td>
<td>Normalised GC</td>
</tr>
<tr>
<td>NR</td>
<td>Newton-Raphson</td>
</tr>
<tr>
<td>OC</td>
<td>Orientation Correlation</td>
</tr>
<tr>
<td>OCT</td>
<td>Optical Coherence Tomography</td>
</tr>
<tr>
<td>PC</td>
<td>Phase-Correlation</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td>PGreg</td>
<td>Phase Gradient Registration Algorithm</td>
</tr>
<tr>
<td>P-SG-GC</td>
<td>Phase-Based Savitzky-Golay Gradient-Correlation</td>
</tr>
<tr>
<td>PSNR</td>
<td>Peak Signal to Noise Ratio</td>
</tr>
<tr>
<td>RANSAC</td>
<td>Random Sample Consensus</td>
</tr>
<tr>
<td>RMS</td>
<td>Root Mean Squared</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>SGD</td>
<td>Savitzky-Golay Differentiator</td>
</tr>
<tr>
<td>SGGC</td>
<td>Savitzky-Golay Gradient-Correlation</td>
</tr>
<tr>
<td>SLAM</td>
<td>Simultaneous Localisation And Mapping</td>
</tr>
<tr>
<td>SSIM</td>
<td>Structural Similarity</td>
</tr>
<tr>
<td>STD</td>
<td>Standard Deviation</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular Value Decomposition</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Motivation

Noncontact measurement of the deformation and shape of objects has been extensively used in various fields of computer vision. For example, deformation measurement techniques are essential tools in experimental mechanics to determine or assess mechanical properties of objects and to understand or predict their behaviour [1–6]. Two-dimensional (2D) and three-dimensional (3D) digital image correlation (DIC) techniques are the most commonly used methods for measuring subpixel deformations from images [3, 7, 8]. Digital volume correlation (DVC) (also known as volumetric-DIC) is an extension of DIC for measuring 3D subvoxel deformations (or displacements) from volumes [1, 4, 9]. Most of the algorithms used in DIC and DVC techniques are based on the algorithms developed for general 3D reconstruction and image registration tasks [10], but DIC and DVC techniques required significantly higher precision and robustness than other computer vision applications. The information gathered about the mechanical properties of the objects using DIC and DVC techniques can also provide input data to inform or construct computational models [11]. Computational models are valuable tools to facilitate studying or predicting the behaviour of objects under certain conditions.

Bioengineering is one of the areas that can benefit greatly from measurements obtained using DIC and DVC techniques. One of the most important applications of DIC and DVC
Introduction

One of the most important applications of these techniques is to identify mechanical properties of human body tissues, such as bone, muscle, skin, heart, and liver [11]. The majority of the current methods for identifying the mechanical properties of tissues are based on measuring in vivo or in vitro deformation or behaviour of tissues during mechanical tests. For example, the deformations of skin and breast tissues can be captured and measured during biaxial, compression, suction, or indentation tests [12–17]. In addition, DVC algorithms have been used to quantify the volumetric deformations of biological tissues, such as the brain [18], and bones [4, 5], or non-biological volumes [19–22]. The volumetric deformations in these applications were typically captured using X-ray computed tomography (CT) devices [4, 5, 18–22].

The identified properties of tissues also provide parameters for computational models of human body tissues. Biomechanical models of tissues can assist clinicians to improve many of the current clinical methods and surgical procedures, by:

- modelling tissue movements to register pre-operative images to the surgical presentation of the tissue during image-guided surgeries [23];
- modelling breast deformation between diagnostic magnetic resonance (MR) images and pre-operative supine MR images to improve the diagnosis and treatment of breast cancer [16, 24];
- providing 3D surface images of the breast that can help to improve oncoplastic, reconstructive and aesthetic breast surgery [25, 26];
- modelling skin tissue to study the processes of wound healing and scar formation [27]; and
- modelling the biomechanics of stretched skin to rationalise tissue expansion methods used in plastic and reconstructive surgeries [28].

Human body tissues have a complex behaviour that varies significantly across different individuals depending on factors, such as age, gender, and ethnicity [16, 17]. For example, the skin is an anisotropic, nonlinear, and viscoelastic tissue [12, 29, 30], and liver and breast tissues have a heterogeneous distribution of their mechanical properties [31–33]. It is thus essential to identify and measure properties of tissues for each individual, and thereby develop patient-
specific models. Furthermore, the deformation measurements provided by DIC and DVC techniques ought to be sufficiently accurate and robust to be able to identify the complex behaviour of tissues. The accuracy of the measurements is thus a key factor in determining how well a computational model can describe the real behaviour of the tissue.

Despite the importance of the accuracy and robustness of DIC and DVC techniques, existing methods have some limitations, including:

1. inability to estimate large shifts [34, 35];
2. poor performance when processing poorly-textured images [1, 35];
3. requiring a good initial estimate of the parameters [36];
4. requiring accurate grey level interpolations and a suitable shape function [8];
5. being computationally complex and slow [8, 37];
6. providing insufficient precision [35]; and
7. inability to calibrate simultaneously cameras of a multi-camera system in 3D DIC applications [38].

The limitations and shortcomings of existing DIC and DVC algorithms have proven particularly problematic when measuring deformations of tissues. For example, in [39] 3D DIC required manual tracking of markers because it could not measure large deformations of the breast using intrinsic textures alone. Also, volume deformation measurements from optical coherence tomography (OCT) images have seen limited success because DVC cannot easily deal with low-quality images [40]). Reconstruction of living skin tissue was performed using multi-view stereo and manual tracking [41] at significantly lower accuracy compared to stereoscopic reconstruction because 3D DIC techniques often cannot simultaneously calibrate multiple cameras.

Significant improvements to existing DIC and DVC techniques are required to overcome the aforementioned challenges and limitations, and thus increase the usability of these techniques in many applications. However, this undertaking is a challenging and complicated
task since DIC and DVC techniques comprise a set of algorithms, and the overall performance of each technique is dependent on the performance of every step of the chain. Hence, their limitations need to be addressed separately at each step to increase the overall accuracy and robustness.

This thesis aims to develop a set of novel methods and tools that address many of the limitations and shortcomings of algorithms used in DIC and DVC techniques. Even though the developed methods and tools are applicable to many areas of computer vision, this thesis has a particular emphasis on DIC and DVC applications in bioengineering.

1.2 Objectives

The main objectives of the thesis were to:

1. address the limitations of existing 2D DIC techniques by developing a novel subpixel image registration method;
2. evaluate the performance of deformation measurements in practical applications and in the presence of noise;
3. track skin deformations using only intrinsic features;
4. perform motion correction in videos;
5. address the limitations of existing DVC techniques by developing a novel 3D subvoxel volume registration method;
6. develop a hand-held device for measuring skin deformations in vivo;
7. address the limitations of 3D DIC techniques by developing novel and accurate multi-camera calibration and 3D reconstruction methods; and
8. suggest solutions to decrease the computation time of DIC and DVC techniques.

1.3 Thesis overview and novel contributions

The outline and contributions of this thesis at each chapter are as follows:
Chapter 2 describes a novel method developed to perform 2D subpixel image registration. This chapter includes a review of the current state-of-the-art algorithms for 2D subpixel image registration, and their shortcomings. The method proposed in this chapter was developed to address the limitations of existing algorithms in accurately measuring small to large shifts, especially in low-textured or noisy images.

Chapter 3 investigates the effects of camera settings on the image noise level and the performance of 2D subpixel image registration in practical applications. The various types of camera noise are introduced in this chapter and a method is proposed to measure the camera image noise. The trade-off between increasing the camera gain or exposure time and the image noise level is quantified for the specific camera used in this work. The accuracy of 2D subpixel image registration methods, including the method proposed in Chapter 2, are evaluated in various camera settings for practical applications.

Chapter 4 evaluates and tests the 2D subpixel image registration proposed in Chapter 2 for measuring 2D skin deformation using only intrinsic features.

Chapter 5 evaluates and tests the 2D subpixel image registration proposed in Chapter 2 to correct motion artefacts in images.

Chapter 6 proposes a novel, accurate, and fast algorithm for subunit registration of arbitrary dimensional data. This chapter reviews the current DVC techniques and their
limitations. Even though the proposed method can be used to register arbitrary dimensional data, this chapter only focuses on 2D and 3D applications.

- **Chapter 7** describes the design of a hand-held stereoscopic device for measuring skin deformations in vivo. This device comprised four 150 frame/s cameras to capture images from the skin deformation in vivo with an overlapping field of view of 175 mm × 175 mm.

- **Chapter 8** discusses a novel method to simultaneously calibrate cameras of a stereoscopic system. This chapter reviews the current multi-camera calibration techniques and their limitations. The proposed method is tested using the hand-held device designed in Chapter 7.

- **Chapter 9** describes a novel model-based method to increase the accuracy of multi-camera calibration processes. The process of identifying control points of the calibration target is improved using a synthetically generated model of the calibration target, and the subpixel image registration method proposed in Chapter 2. In addition, a method is proposed to model lens distortions using Zernike polynomials. A 3D reconstruction method for stereoscopic systems is also proposed and tested in this chapter.

- **Chapter 10** draws conclusions from the topics covered in this thesis, and outlines possible future work that can be performed in this area.

- **Appendix A** is a comprehensive review on suitability of recent digital signal processors (DSPs), field-programmable gate arrays (FPGAs), and graphics processing units (GPUs) for computer vision and image processing algorithms. The purpose is to investigate and compare DSPs, FPGAs, and GPUs that are suitable for implementing DIC and DVC algorithms to increase their computation speed compared to their CPU
implementation. This appendix also reviews the speed-up ratios achieved in the literature for implementing image processing and computer vision algorithms using DSPs, FPGAs, and GPUs, compared to their CPU implementations.

- **Appendix B** describes a proposed FPGA implementation of 2D cross-correlation (CC) as an example of using hardware accelerators to speed up the computations. CC is the main part of the registration methods proposed in Chapter 2 and Chapter 6 and many other image registration methods.
2 Subpixel image registration

The content of this chapter is largely based on the following journal paper, which is in review for the journal of Computer Vision and Image Understanding:


2.1 Abstract

This chapter presents a new method for finding two-dimensional translational shifts with subpixel accuracy. The algorithm is a two-step method which is able to measure subpixel shifts, even in images with few features or noisy images where many existing algorithms fail. This method is the first to use Savitzky-Golay differentiators (SGDs) in gradient correlation to find the integer shift, followed by a modified phase-based method to find the subpixel shift. Using SGD to find the integer shifts improved the robustness of the algorithm to Gaussian and salt and pepper noise, and its ability to find shifts in low-textured images. The proposed modified phase-based method uses Hann windowing, 2D median filtering, and frequency-dependent selection of the phase data prior to 2D regression to find the phase gradient. These steps helped to increase the subpixel accuracy of the proposed method and its robustness to Gaussian and salt and pepper noise.
Comprehensive tests were conducted on 2400 subimages (128 pixel × 128 pixel) of the standard LANDSAT images subjected to both subpixel and large synthetic shifts, and synthetic image rotation. Two error metrics were used to quantify the algorithm output accuracy of the integer and subpixel parts of the proposed algorithm. The tests for translational shifts indicated that the accuracy of this method in finding shifts is of the order of a few ten-thousandths of a pixel, which is a substantial improvement over other state-of-the-art methods. In rotation tests, the method outperformed comparable techniques for finding the displacement in rotated images. Results showed that the proposed method generally provides better performance than existing methods when images have been degraded by Gaussian and salt and pepper noise.

2.2 Introduction

Rigid and non-rigid image registration techniques are widely used in various areas [42]. Fine registration algorithms for rigid transformations with subpixel accuracy are of particular interest in motion estimation and tracking, image alignment, medical image processing, super-resolution reconstruction, image stitching, satellite image analysis, and change detection [42]. Subpixel image registration techniques are also used for surface inspection, deformation measurement, and strain measurement in industrial and medical applications. Digital image correlation (DIC) is an image registration method used in experimental mechanics for small subimages where a speckle pattern is added to the surface and small displacements between two images are measured [43]. DIC has medical and non-medical applications, including 2D surface deformation measurement for anisotropic elastic membranes [44], finding strain fields in human tendon tissue [45], measuring 3D movements of rotary blades [46], and quantifying 3D deformation during metal sheet welding [47].

Two main approaches are available for subpixel image registration. In the first approach, integer and subpixel shifts are found separately using different methods (Table 2-1). The second approach finds integer and subpixel shifts in a single step using a coarse-to-fine methodology. Iterative optimisation processes and optical flow techniques are two examples of this class of methods.
Iterative methods treat the registration as a cost function, where the subpixel shift is found using an iterative error minimisation process based on methods such as Newton-Raphson (NR) or gradient-based (GB) algorithms [59]. Optical flow techniques measure the image motion based on local image gradients [60].

For finding the integer shift, cross-correlation (CC) is not sufficiently accurate, and normalised cross-correlation (NCC) is computationally intensive. Phase-correlation (PC) is another algorithm for finding integer shifts, based on the Fourier shift property and normalised cross power spectrum. PC is more robust to noise and changes in mean intensity across the image, and has been shown to perform better than CC [49]. Gradient-correlation (GC), normalised GC (NGC) [50][51], and orientation correlation (OC) [52][53] methods use central differences in the form of a complex term instead of the intensity values in the CC function.
For finding the subpixel shift between two images, template matching followed by interpolation in the spatial domain is one of the methods often used in the literature [48][53][54]. For example, Messerli et al. [53] used bicubic intensity interpolation, and local weighting of a neighbourhood of the NCC peak. Although Messerli et al. [53] suggested a search region to reduce the computation time, interpolation methods are typically computationally intensive and slow.

Guizar-Sicairos et al. [55] proposed a method to perform interpolation in the frequency domain (FFT-upsampling). More recently, Yousef et al. [56] reduced the computational complexity and memory requirement of this method without compromising its accuracy. However, the accuracy of FFT-upsampling methods is limited by the upsampling ratio as well as the interpolation method. Furthermore, these methods become very slow for large upsampling ratios.

Feng et al. [61] proposed a method to measure subpixel shifts from the centroid of the correlation surface using a modified moment method. Their method was faster and more accurate than the FFT-upsampling of Guizar-Sicairos et al. [55].

Curve fitting approaches numerically fit functions near the peak of the CC or GC function to find the subpixel shift. Examples of functions include Gaussian [57], quadratic [51], and modified Mexican hat wavelet [50]. The accuracy of these methods is highly dependent on the shape of the CC or GC function near the peak, and is therefore intensity-dependent. One drawback is that these methods usually contain a time-consuming iterative optimisation process to find the best fit.

Foroosh et al. [49] used PC and analytically demonstrated that subpixel shifts lead to a downsampled 2D Dirichlet kernel, which they approximated using a 2D sinc function to find the subpixel shift. They mentioned several sources of error that arose in their algorithm, such as the 2D sinc function approximation, non-overlapping regions in the two images, and aliasing [49].
Another technique for calculating the subpixel shift is to use a phase-based approach, employing the frequency shift theorem and the gradient of the frequency domain CC of the two images. In this method, phase unwrapping is necessary for shifts larger than one pixel. Vandewalle et al. [62] used the central part of the phase data, which introduces error when the shift is more than one pixel. The method used by Stone et al. [58] and Malcolm et al. [44] first determined the integer shift and registered the two images to within half a pixel to avoid requiring phase unwrapping, then used 2D regression (plane-fitting) on the phase difference data to find the subpixel shift. This method is sensitive to aliasing and, in order to offset this, Stone et al. [58] removed phase data with magnitudes less than a specified threshold. An important aspect of this method is the ability to register two images to within half a pixel at the first step, which is difficult to achieve using CC (e.g. in [44] and [58]), especially in images with few features [49].

The iterative NR and GB algorithms are two methods widely used in DIC and experimental mechanics for measuring subpixel displacements [59]. The NR method minimises the error between the original subset image deformed by shape functions, and the second image. In GB, subpixel displacements are modelled with a first or second order Taylor expansion, and the error between two images is minimised using the least squares method. The high computation costs of these two techniques make them slow. The criteria for minimising both NR and GB are based on the differences in intensity values, rendering these methods sensitive to intensity changes and making them suitable for small shifts only. However, in many applications in experimental mechanics, the displacement vectors may range in magnitude from subpixel size to large values. In addition, each subset deformation in the iterative NR method involves interpolation, which introduces additional systematic errors [63].

For applications involving images with few features, or where accuracy and near-real-time analysis are important, more robust subpixel registration algorithms capable of finding small to large translational shifts are required. To address these needs, this chapter presents a new two-step method for subpixel image registration and deformation measurement. The use of
Savitzky-Golay differentiators (SGDs) [64] is proposed for the first time as an extension of the GC method to increase the accuracy and noise tolerance. Furthermore, a new normalisation procedure is proposed in order to improve (over the use of NGC) the dynamic range and robustness to noise. For subpixel shift calculations, a phase-based method similar to [58] is proposed, but with: a different windowing function; normalisation and pre-filtering of the phase difference data; and modifications to decrease the effect of aliasing and increase the accuracy of the shift estimates. A detailed analysis is performed to motivate the selection of appropriate windowing functions for the integer and the subpixel parts of the algorithm.

For error analysis in image registration algorithms, two major methods are common. The first method is to apply known synthetic shifts to test images and to find the shift estimation error for the algorithms [8,10,12,14,15]. To make conditions similar to those of practical applications to which this method might be applied, white Gaussian noise is usually added to the images before the shift estimation [50, 65]. Even though this method provides a good estimation of the algorithm accuracy in comparison to other methods, the accuracy is dependent on the image dataset selected during testing. The other method of quantifying the accuracy of the algorithm is to measure shifts in the images of a rigid object undergoing a well-controlled pure translational displacement [44]. This method gives practical error analysis of the algorithm for the specific setup and specific object tested. However, the performance will vary between objects, even in the same setup. Furthermore, in some applications, such as medical image processing, it is very difficult and often impossible to test the algorithm using this method. In this chapter, the algorithm is tested based on the first approach (i.e. synthetic shift of images). Two error metrics were defined to evaluate the accuracy of the integer and subpixel parts of the algorithm. These metrics are introduced to address the limitations of the first and second methods of registration error analysis by providing estimates of the algorithm accuracy when it is applied to specific images, and where it is not possible to perform rigid object translational tests.

The proposed method was compared, for finding subpixel and large translational shifts, with the phase-based methods of Vandewalle et al. [62] and Stone et al. [58], the FFT-
upsampling method proposed by Guizar-Sicairos et al. \[55\], PC with 2D sinc function fitting (proposed by Foroosh et al. \[49\]), the method of Feng et al. \[61\], the optical flow motion estimation proposed by Brox et al. \[60\], and template matching using OC and bicubic interpolation proposed by Messerli et al. \[53\]. Note that the method proposed by Stone et al. \[58\] covers the phase-based method proposed independently by Malcolm et al. \[44\]. Also note that the method proposed by Guizar-Sicairos et al. \[55\] and the method proposed by Yousef et al. \[56\] have the same level of accuracy and noise-robustness \[56\]. The performance of the algorithm has been quantitatively compared with these methods for finding subpixel shifts in the presence of rotation.

2.3 Method

1.1.1 Finding the integer pixel translation shift

The intensity 2D-CC algorithm is a widely recognised method for finding the translational shift between two images. Let $I_1(x, y)$ and $I_2(x, y)$ denote the intensity values of two grey-scale images of size $N$ pixel $\times$ $N$ pixel at the point $(x, y)$, then the 2D-CC of two templates is defined as:

$$CC(k, j) = \sum_{x=1}^{N} \sum_{y=1}^{N} I_1(x, y) \times \bar{I}_2(x - k, y - j)$$

where $-N < k, j < N$, and the over-bar denotes complex conjugation. The negative indexes are usually replaced with zero (i.e. zero padding). 2D-CC can be computed in the frequency domain using the 2D FFT and its inverse (2D IFFT) \[66\](Equation 2.2).

$$CC = IFFT(FFT(I_1) \times FFT(I_2))$$

In Equation 2.2, $I_1$ and $I_2$ are intensity arrays of the images, and the 2D FFT of an image with size of $N \times N$ is given by:
Subpixel image registration

\[ CFFT(I_1(k, j)) = \sum_{x=1}^{N} \sum_{y=1}^{N} I_1(x, y)e^{-2\pi i \left(\frac{k x}{N} + \frac{k y}{N}\right)} \]  

where \( i = \sqrt{-1} \). In PC, instead of Equation 2.3, the normalised cross-power spectrum (Equation 2.4) was used to find the translational shift [49].

\[ NCPS = \frac{FFT(I_1) \times FFT(I_2)}{|FFT(I_1) \times FFT(I_2)|} \]  

In GC, image intensities were replaced with intensity gradients in the form of a complex term to find the translational shift with Equation 2.2. To calculate the real and the imaginary parts of this complex term, horizontal and vertical intensity gradients, respectively, are approximated using central differences in the \( x \) (\( CD_x \)) and \( y \) (\( CD_y \)) directions (Equation 2.5 and Equation 2.6).

\[ CD_x = I(x + 1, y) - I(x - 1, y) \]  
\[ CD_y = I(x, y + 1) - I(x, y - 1) \]

Central differences approximate an ideal differentiation at a specific pixel position (Figure 2-1). First order central differences were used in [51], whereas Tzimiropoulos et al. [50] improved the shift estimation accuracy by using second order central differences. However, central differences can be inaccurate, especially for noisy signals, since they involve interpolation of the neighbouring points that are already corrupted by noise. Instead, SGDs provide a more accurate approximation of the noise-free signal shape by smoothing the signal using running least-squares fitting to a polynomial. SGDs can reduce the noise, preserve the signal shape, and approach the ideal differentiator at low frequencies [64]. Figure 2-1 illustrates the frequency response of a 7 point cubic first derivative SGD in comparison to first and second order central differences, and the ideal differentiator.
Figure 2-1: The frequency response of a 7 point cubic first derivative SGD in comparison to first and second order central differences, and the ideal differentiator.

SGDs have the advantage that, for discrete signals, they can easily be implemented using convolution and a table of coefficients for each filter order, instead of using running least-squares fitting [67].

Based on the frequency response properties, and in order to use more information from the neighbouring points, a 7 point cubic first derivative SGD is used in the proposed method (details are described in [67]). The convolution kernel of this filter is:

$$SGD(Kernel) = \frac{1}{252}([22, -67, -58, 0, 58, 67, -22])$$  \hspace{1cm} 2.7

In this method, the SGD convolution kernel is applied to each row and column of the image to form SGDx and SGDy, respectively, and forms the complex term (Equation 2.8):

$$SGD(x,y) = SGDx(x,y) + SGDy(x,y) \times i$$  \hspace{1cm} 2.8

This term is used as the input to Equation 2.9 to form the 2D-Savitzky-Golay gradient-correlation (SGGC).
\[ SGGC = \text{IFFT}(\text{FFT}(SGD_1) \times \text{FFT}(\overline{SGD_2})) \]

where \( SGD_1 \) and \( SGD_2 \) are in the form of rectangular matrices. Applying a window function to this matrix is advantageous as it decreases boundary effects, aliasing, and noise [68]. Noise and fine details of the images (such as edges) are both present at high-frequency bands in the frequency domain. Selection of an appropriate window function is thus important to simultaneously preserve fine details (which are important to have an accurate matching between two images), and reduce the noise. In general, due to the limited resolution of imaging systems, rapid intensity changes are uncommon in images. Therefore, fine details of the image are usually in the lower range of high-frequency components compared to uncorrelated noise (Figure 2-2). For this method, the Hamming window was chosen to retain the image information in low frequencies and remove the noise in high frequencies (the frequency response of the Hamming window has a mild slope of attenuation (-6 dB/octave) in the first lobe, and high attenuation (-43 dB) in the second lobe (Figure 2-3) [68]). The noise robustness property of the proposed SGGC helps to extract fine details of the images, even in the presence of noise. The combination of the Hamming window with SGGC provides a good compromise between reducing the noise and maintaining fine details of the images at the higher frequencies, an advantage over other methods. The Blackman window used in [58], removes most of the high frequency information (i.e. fine details) in the images (Figure 2-3), and the Tukey window used in [50] and [62] does not have enough attenuation in its second lobe (Figure 2-3). The Tukey window also gives rise to phase distortions due to the ripples in its frequency response.
Figure 2-2: The power spectrum ($20\log_{10}(\text{abs}(\text{FFT}(I))))$ (dB scale) of a sample image (LANDSAT image of Paris in Figure 2-5) (a) for the original image (b) with addition of a small amount of white Gaussian noise (variance of 0.001 of the normalised intensity values) to the image. The zero frequency (i.e. DC component) is shifted to the centre; hence higher frequencies are close to the peripheral parts. The white Gaussian noise has spread in higher frequencies in Figure 2-2(b), in comparison to fine details of the image in Figure 2-2(a).

Figure 2-3: The frequency response of Hamming, Hann, Blackman, and Tukey ($\alpha = 0.25$) window functions.

Normalising the intensity values is a way to reduce the algorithm sensitivity to changes in mean intensity across the image, and to increase the dynamic range. Foroosh et al. [49] proposed normalised cross-power spectrum Equation 2.4 for normalisation, and
Tzimiropoulos et al. [50] used a similar formula for normalisation of the gradient-correlation (i.e. NGC). However, the individual terms of the normalised cross-power spectrum (Equation 2.4) and NGC include a DC component, which results in a reduction of the dynamic range. Instead, $SGD_1$ and $SGD_2$ (Equation 2.8) were normalised separately, based on the diagram in Figure 2-4, prior to substituting them in (Equation 2.9) to find $SGGC$.

![Diagram of the normalisation procedure](image)

**Figure 2-4:** The diagram of the normalisation procedure.

This approach performs better in decreasing intensity sensitivity compared with the normalisation technique of [49] and [50], since the real and imaginary parts of the complex term are normalised separately after the DC component was removed by subtracting the mean value (Figure 2-4).

### 1.1.2 Finding the translational subpixel shift

In the proposed method, the subpixel shift is found based on the frequency shift theorem, as described in [44] and [58] but with some modifications.

The Fourier transform of a signal ($F(\omega)$) includes separate real and imaginary parts (Equation 2.3), which can be represented in the form of Equation 2.10.

$$F(\omega) = \Re(F(\omega)) + i \times \Im(F(\omega)) \tag{2.10}$$

The phase spectrum ($\varphi$) for two signals is given by:

$$\varphi = \tan^{-1}\left(\frac{\Im(F_1(\omega)) \times \Im(F_2(\omega))}{\Re(F_1(\omega)) \times \Re(F_2(\omega))}\right) \tag{2.11}$$

The gradient of the $\varphi$ data of two 1D signals at the centre can be used to find the subpixel shift, when phase wrapping has not happened [44]. The same concept is extendable to 2D and the phase plane ($\varphi_p$) to find the subpixel translational shifts in images ([44] [58]).
For this modified phase-based method, the same normalisation procedure previously described for the integer part of the algorithm was used, and the Hann window was applied on images before computing the FFT of the intensity values in Equation 2.11. The Hann window was chosen, instead of the Hamming window (used in the integer part of algorithm in Section 1.1.1), or the Blackman window, because of its frequency domain properties. The Hann window frequency response has moderate attenuation of 31.5 dB in the second lobe ([24]), and sharp slope of attenuation (-18 dB/octave [24]) (Figure 2-3).

Note that, in the subpixel part of the algorithm, the images are previously registered to within less than 0.5 pixel in the integer part. Therefore, removal of the noise is the main consideration in this part. Blackman and Hann window functions are both capable of filtering noise in high-frequency data (Figure 2-3). Nevertheless, the Hann window also preserves low frequency \( \varphi_p \) data because it has less attenuation in its second lobe compared to the Blackman window (Figure 2-3).

Instead of using frequency masking to remove aliased frequency components, as proposed by Stone et al. [58], a 2D median filter was applied on the \( \varphi_p \) data to remove spurious frequencies caused by aliasing or noise (median filters can eliminate impulse noise [69]). A small neighbourhood size of 2D median filter was selected (i.e. 3 pixel \( \times \) 3 pixel) to help to smooth the \( \varphi_p \) data, without altering the \( \varphi_p \) slope at the centre. This 2D median filter has the effect of improving the 2D regression required for finding the subpixel shift from the slope of the \( \varphi_p \) data. However, the 2D regression becomes less accurate when the \( \varphi_p \) data are affected by noise. To address this issue, Stone et al. [58] proposed to limit the frequency range of the \( \varphi_p \) data near the origin. They chose a fixed value of 0.6 of the image size around the centre of the \( \varphi_p \) data as the number of samples (\( N_\varphi \)) to be used in 2D regression [58]. Nevertheless, they also mentioned that they empirically found that this constant factor can range from 0.5 to 0.7 without significantly affecting the algorithm [58]. However, the choice of an appropriate (\( N_\varphi \)) is dependent on the noise level of the image. As discussed in Section
1.1.1, both the fine details of images and image noise contribute to the high frequency components of the image. This means that by selecting a smaller $N_\varphi$ around the centre, noise and fine details are filtered from the $\varphi_p$ data at the same time. To preserve the fine details of the image in less noisy images, and remove the noise in noisy images, a different number of samples was chosen around the centre of $\varphi_p$ data, prior to 2D regression. The noise level of an image was indicated based on the 2D standard deviation (2DSD) of the $\varphi_p$ data. In the ideal case of noise- and aliasing-free $\varphi_p$ data, the $\varphi_p$ data is completely symmetric around the centre, which gives the 2DSD value of 0. Therefore, the function in Equation 2.12 was defined to select $N_\varphi$ dependent on the image noise level (i.e. $2\text{DSD}(\varphi_p)$).

$$p = \begin{cases} 
(0.95 - 2 \times 2\text{DSD}(\varphi_p)), & \text{if } 2\text{DSD}(\varphi_p) \leq 0.1 \\
0.75, & \text{if } 2\text{DSD}(\varphi_p) > 0.1
\end{cases}$$

$$N_\varphi = p \times \text{Image size}$$

The maximum value of $N_\varphi$ in Equation 2.12 is 0.95 of the image size to avoid border effects, and the minimum value is 0.75 of the image size to preserve the image details. The function (Equation 2.12) helps to maintain the 2D regression accuracy depending on the image noise level, as opposed to the fixed value of 0.6, which was proposed by Stone et al. [58].

2D regression is computationally faster than principal component analysis (PCA) and singular value decomposition (SVD) for finding the gradient of the $\varphi_p$. In addition, tests showed 2D regression was more accurate than the other two methods for finding the slope at the centre of the $\varphi_p$ data.

1.1.3 Ground truth data and comparison metrics

1.1.3.1 Integer and subpixel image shifts

The subpixel image registration algorithms were evaluated by applying synthetic shifts on the 6 standard LANDSAT images used in [50], shown in Figure 2-5, and then estimating the shift value resulting from each algorithm. A variety of subpixel or integer shifts, in two groups consisting of values smaller or greater than 1 pixel, were applied to the whole image using
Subpixel image registration

FFT-shift [70] (since it resembles real-world experiments [70]). Images were cropped into 128 pixel × 128 pixel subimages, the image size which was used in [49], [50], and [58]. The central points of subimages were selected across the entire image, with a step increment of 20 pixel, and the minimum distance of 64 pixel to the periphery (Figure 2-6(a)). This resulted in 400 subimages (Figure 2-6(b)) for each image, and 2400 subimages in total for 6 LANDSAT images of Figure 2-5. A sample image of displacement vectors is shown in Figure 2-7 for the LANDSAT image of Paris.

Figure 2-5: Standard LANDSAT images used in this study. Images contrasts have been enhanced for presentation purposes, but all of the analyses in this study were performed on original images.
Subpixel image registration

Figure 2-6: (a) The central points of subimages on the LANDSAT image of Paris. (b) A sample subimage (128 pixel × 128 pixel)

Figure 2-7: The displacement vectors measured at each subimage of the LANDSAT image of Paris (Figure 2-6) between the original image and synthetically shifted image using the proposed method.

The average registration error in finding the translational shift ($RE_T$) was computed based on:

$$RE_T = \frac{1}{2400} \sum_{m=1}^{6} \sum_{n=1}^{400} \sqrt{(x(m, n) - \hat{x}(m, n))^2 + (y(m, n) - \hat{y}(m, n))^2}$$
where \([x, y]\) is the pixel position, a hat indicates the estimated pixel position values, \(m\) is the LANDSAT image number and \(n\) is its subimage number. \(RE_T\) values were used to evaluate the performance of algorithms in finding translational shifts. However, as discussed in Sections 1.1.1 and 1.1.2, an appropriate window function is important in increasing the performance of the image registration algorithms. Therefore, for the algorithm proposed by Foroosh et al. [10], a Hann window was added to their algorithm to improve its performance, and make a fair comparison with this method. This modified version is called “Foroosh + HW” in this chapter.

1.1.3.2 Image rotation

To evaluate the translational shift error in the presence of rotation, rotation values of 0.5°, 1°, 2°, and 3° were applied to the whole image in 6 LANDSAT images with the centre of rotation at the middle of the image. To estimate the algorithm error in finding translational shifts in the presence of rotation, these steps were followed:

1. The images were cropped to subimages (as described in Section 1.1.3.1, and shown in Figure 2-6) before and after applying the rotation;
2. The translational shifts in the \(x\) and \(y\) directions \((T_x, T_y)\) were found between each of the 400 subimages of the original LANDSAT images and the associated subimage of the rotated image;
3. The magnitudes of the translational shift for each subimage were calculated using:

\[
M = \sqrt{(T_x)^2 + (T_y)^2}
\]

This gave 400 discrete values uniformly distributed over the whole image;
4. These 400 discrete \(M\) values were interpolated using cubic spline interpolation for all the integer pixel locations to form a continuous rotation pattern on the whole image;
5. The magnitude of the shift resulting from the ideal rotation \((M_r)\) was modelled mathematically for each pixel location \(([x, y])\) using Equations 2.15 and 2.16.
\[
\begin{aligned}
[x_r, y_r] &= [x_0, y_0] + \left( [x - x_0, y - y_0] \times \begin{bmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{bmatrix} \right) \\
M_r &= \sqrt{(x_r - x)^2 + (y_r - y)^2}
\end{aligned}
\]

where \( \theta \) is the rotation angle in radian, \([x_0, y_0]\) is the centre of the image, and \([x_r, y_r]\) is the pixel position of the point \([x, y]\) in the rotated image;

6. The average registration error of the algorithm for estimating translational shift in the presence of rotation (\(RE_R\)) was calculated using:

\[
RE_R = \left( \sum_{m=1}^{6} \sum_{n=1}^{400} \sqrt{(M_r(m, n) - M(m, n))^2} \right) / 2400
\]

where \( m \) is the LANDSAT image number, and \( n \) is its subimage number.

1.1.3.3 Error metrics

Two error metrics were used to analyse of the integer and subpixel parts of the algorithm. The purpose of these metrics is to give an estimation of the algorithm accuracy in distinct images with different features and noise levels. The distinctiveness of the dominant peak in the output of 2D-CC or GC is often considered to be indicative of the ability of the algorithm to determine the integer shift [49][65][71]. For the integer part of the algorithm, the number of points in which the normalised SGGC value in Equation 2.9 was greater than 0.85 was used to indicate the error in finding the integer shift in the algorithm. The closer this metric value is to 1, the less shift estimation error will be achieved by the algorithm in the integer part.

The error metric of the subpixel part of the algorithm was defined to be the plane fitting error for the plane fitted to \( \varphi_p \). A small fitting error indicates good accuracy of the algorithm in estimating the subpixel shifts. The pre-filtering of the phase data, using a 2D-median filter with a small neighbourhood, helps to remove outliers from the phase data, resulting in robust plane fitting, and thus a reliable subpixel error metric.
2.4 Results and discussion

1.1.4 Translational shifts

Table 2-2 and Table 2-3 show the $R_{E_T}$ (Equation 2.13) for shifts in the $x$ and $y$ directions ([x, y]) for each algorithm. The average $R_{E_T}$ of the proposed method is approximately four times smaller than the next best method (i.e. Stone et al. [58]), for shifts of less than one pixel (Table 2-2). This demonstrates that, although a similar procedure was chosen for subpixel displacement measurement, the modifications have significantly improved the accuracy.

Table 2-2: The average registration errors (in pixel) for all 2400 LANDSAT subimages (128 pixel × 128 pixel) ($R_{E_T}$) for the subpixel shifts indicated in the first row. The final column provides the relative magnitude of the average registration errors quantified with respect to that of the proposed method.

<table>
<thead>
<tr>
<th>Method</th>
<th>Registration errors at shifts [x, y] (pixel)</th>
<th>Average (pixel)</th>
<th>Relative difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[0.125, 0.875] [0.250, 0.750] [0.375, 0.625] [0.500, 0.500]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vandewalle et al. [62]</td>
<td>0.0358 0.0310 0.0281 0.0271</td>
<td>0.0305</td>
<td>115.85</td>
</tr>
<tr>
<td>Stone et al. [58]</td>
<td>0.0005 0.0009 0.0012 0.0014</td>
<td>0.0010</td>
<td>3.85</td>
</tr>
<tr>
<td>Guizar-Sicairos et al. [55]</td>
<td>0.0679 0.0717 0.0723 0.0734</td>
<td>0.0713</td>
<td>270.93</td>
</tr>
<tr>
<td>Foroosh et al. [49]</td>
<td>0.0528 0.0757 0.1033 0.1428</td>
<td>0.0936</td>
<td>355.71</td>
</tr>
<tr>
<td>Foroosh + HW</td>
<td>0.0017 0.0040 0.0072 0.0121</td>
<td>0.0062</td>
<td>23.67</td>
</tr>
<tr>
<td>Feng et al. [61]</td>
<td>0.1086 0.1113 0.1089 0.0729</td>
<td>0.1004</td>
<td>381.84</td>
</tr>
<tr>
<td>Brox et al. [60]</td>
<td>0.1574 0.3264 0.4778 0.1167</td>
<td>0.2696</td>
<td>1025.00</td>
</tr>
<tr>
<td>Messerli et al. [53]</td>
<td>0.0341 0.0465 0.0495 0.0104</td>
<td>0.0351</td>
<td>133.56</td>
</tr>
<tr>
<td>The proposed method</td>
<td>0.00010 0.00021 0.00032 0.00042</td>
<td>0.00026</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2-3: The average registration errors (in pixel) for all 2400 LANDSAT subimages (128 pixel × 128 pixel) ($R_{E_T}$) for shifts greater than 1 pixel (indicated in the first row). The final column provides the relative magnitude of the average registration errors quantified with respect to that of the proposed method.

<table>
<thead>
<tr>
<th>Method</th>
<th>Registration errors at shifts [x, y] (pixel)</th>
<th>Average (pixel)</th>
<th>Relative difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[1.125, 2.875] [3.250, 4.750] [5.375, 6.625] [7.500, 8.500]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vandewalle et al. [62]</td>
<td>2.9343 5.7021 8.4883 11.2950</td>
<td>7.1049</td>
<td>27118.06</td>
</tr>
<tr>
<td>Stone et al. [58]</td>
<td>0.4062 3.3074 7.0283 10.0330</td>
<td>5.1937</td>
<td>19823.34</td>
</tr>
<tr>
<td>Guizar-Sicairos et al. [55]</td>
<td>0.1576 0.2251 0.3179 0.4250</td>
<td>0.2814</td>
<td>1074.00</td>
</tr>
<tr>
<td>Foroosh et al. [49]</td>
<td>0.0218 0.0438 0.0588 0.1064</td>
<td>0.0577</td>
<td>220.27</td>
</tr>
<tr>
<td>Foroosh + HW</td>
<td>0.0027 0.0069 0.0126 0.0387</td>
<td>0.0152</td>
<td>58.15</td>
</tr>
<tr>
<td>Feng et al. [61]</td>
<td>0.1114 0.1207 0.1227 0.0896</td>
<td>0.1111</td>
<td>424.05</td>
</tr>
<tr>
<td>Brox et al. [60]</td>
<td>0.1839 0.3784 0.5730 0.2551</td>
<td>0.3476</td>
<td>1326.72</td>
</tr>
<tr>
<td>Messerli et al. [53]</td>
<td>0.0358 0.0473 0.0498 0.0096</td>
<td>0.0356</td>
<td>135.97</td>
</tr>
<tr>
<td>The proposed method</td>
<td>0.00010 0.00021 0.00032 0.00042</td>
<td>0.00026</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 2-3 quantifies the ability of the proposed algorithm to find integer and subpixel shifts. The method of Vandewalle et al. [15] is not capable of finding subpixel shifts of more than one pixel, because of phase wrapping. The result using the method of Stone et al. [58] indicates the difficulty that CC has with finding integer shifts in poorly textured subimages. A similar issue arose (albeit to a lesser degree) when using the method of Guizar-Sicairos et al. [9], which failed to find the correct integer shifts in some cases. The method of Foroosh et al. [10] was reasonably reliable for finding subpixel and large shifts, because of the advantages of PC over CC. However, these results highlight the issue of the increase in error for non-overlapping regions, as discussed in Section 2.2.

The method proposed by Feng et al. [61] had a similar performance for both large and subpixel shifts, and had a higher accuracy compared to the FFT-upsampling of Guizar-Sicairos et al. [9] in large shifts (Table 2-3). However, it was observed that this method performed the worst when using the LANDSAT images that had fewer image features. The optical flow method of Brox et al. [60] could handle large shifts, but was unable to provide good accuracy in measuring the shifts. The OC and the interpolation method of Messerli et al. [53] performed relatively well across the LANDSAT images for large and subpixel shifts. Nevertheless, this method was slower than methods of Feng et al. [61], Foroosh et al. [8], and the proposed method. Moreover, the subpixel error of the method of Messerli et al. [53] was dependent on the subpixel shift (Table 2-2 and Table 2-3).

Of the previously published methods, the modified version of the method of Foroosh et al. [49] (Foroosh + HW) was the most accurate method. Nevertheless, the proposed method provided an average registration error \( RE_T \) 58 times smaller than Foroosh + HW (Table 2-3) and performed consistently well for calculating subpixel and large shifts (Table 2-2 and Table 2-3, respectively). The comparisons of the accuracy between Foroosh and Foroosh + HW in Table 2-2 and Table 2-3 emphasise the role of an appropriate window function in increasing the accuracy.
1.1.5 Robustness to noise and evaluation of the proposed algorithm with error metrics

To investigate the robustness of the proposed algorithm to noise, two types of noise (Gaussian white noise and salt and pepper noise) were added to the LANDSAT image of Paris and the registration error ($RE_T$) was calculated and averaged for estimating the $[x, y]$ shift of [3.250, 4.750] over all 400 subimages. Gaussian noise is the dominant type of noise in most imaging systems, and salt and pepper noise is uncorrelated instances of pixels set to minimum or maximum intensity values. For comparison, a similar test was performed for methods of Foroosh + HW, Feng et al. [61], and Messerli et al. [53] (Table 2-2 and Table 2-3).

Two ranges were chosen for the Gaussian noise variance ($\sigma^2$) and salt and pepper noise density ($d$) to characterise the robustness of the algorithms to noise. One range was chosen to represent a low-level of noise (i.e. from $10^{-5}$ to $5 \times 10^{-4}$ of the normalised values - intensities limited to [0, 1]) and one range was chosen to represent a high-level of noise (i.e. from $5 \times 10^{-3}$ to 0.120 of the normalised values). The peak signal to noise ratio (PSNR) and the structural similarity (SSIM) index was measured between the original image and the unshifted noisy images in both ranges and for both types of noise.

Figure 2-8 and Figure 2-9 show the average registration errors for the low-level noise range. These figures indicate that the proposed method has considerably lower estimation error compared to Foroosh + HW, and performed better than the methods of Feng et al. [61] and Messerli et al. [53] for the applied Gaussian noise in this range. The average registration error of the proposed method was 0.027 pixel at $\sigma^2 = 5 \times 10^{-4}$ in Figure 2-8, and 0.013 pixel at $d = 5 \times 10^{-4}$ in Figure 2-9. Even though the method of Foroosh + HW had the next best performance (after the proposed method) in estimating the translational shifts (Table 2-2 and Table 2-3), it was the most sensitive to noise. The methods of Feng et al. [61] and Messerli et al. [53] had similar performances for both types of noise and were not affected by noise in this range.
Figure 2-8: The average registration errors (pixel) in estimating the shift [3.250, 4.750] in 400 subimages of the LANDSAT image of Paris (128 pixel × 128 pixel). The PSNR of the original and unshifted noisy images was from 49.46 dB to 33.00 dB, and the SSIM index was from 0.9949 to 0.8223 in this range of noise.

Figure 2-9: The average registration errors (pixel) in estimating the shift [3.250, 4.750] in 400 subimages of the LANDSAT image of Paris (128 pixel × 128 pixel). The PSNR of the original and unshifted noisy images was from 61.17 dB to 37.55 dB, and the SSIM index was from 0.9999 to 0.9853 in this range of noise.
Figure 2-10 and Figure 2-11 show the average registration errors for the high-level noise range. The result in Figure 2-10 illustrates that the proposed method and the method proposed by Feng et al. [61] performed the best at high levels of Gaussian noise. Note that the method of Feng et al. [61] was specifically developed for low PSNR images. The registration error of the proposed algorithm was 0.85 pixel, even with a large Gaussian noise \( \sigma^2 = 0.120 \) (Figure 2-10). The proposed method consistently outperformed Foroosh + HW while, in similar tests undertaken by Tzimiropoulos et al. [50], their method failed for \( \sigma^2 > 0.045 \), whereas the method of Foroosh et al. [49] was still able to detect subpixel shifts with less than 0.35 pixel error [12].

Figure 2-10: The average registration errors (pixel) in estimating the shift [3.250, 4.750] in 400 subimages of the LANDSAT image of Paris (128 pixel × 128 pixel). The PSNR of the original and unshifted noisy images was from 23.03 dB to 11.29 dB, and the SSIM index was from 0.3452 to 0.0330 in this range of noise.
Figure 2-11: The average registration errors (pixel) in estimating the shift [3.250, 4.750] in 400 subimages of the LANDSAT image of Paris (128 pixel × 128 pixel). The PSNR of the original and unshifted noisy images was from 27.91 dB to 14.09 dB, and the SSIM index was from 0.8763 to 0.1075 in this range of noise.

Figure 2-11 shows that the registration error of the proposed method was less than the Foroosh + HW method, and was similar to the method proposed by Feng et al. [61], whereas the method proposed by Messerli et al. [53] was better than the other methods in dealing with the high-level of salt and pepper noise (Figure 2-11). Figure 2-10 and Figure 2-11 illustrate that, in addition to the PSNR and the SSIM index of the images, the type of noise also affects the accuracy of the registration algorithms.

Figure 2-12 shows examples of images with Gaussian noise (Figure 2-12(b)) and salt and pepper noise (Figure 2-12(c)). The average registration errors of the proposed algorithm in estimating the [x, y] shift of [3.250, 4.750] pixel, over 400 subimages of the noisy images that contained Gaussian noise (Figure 2-12(b)) and salt and pepper noise (Figure 2-12(c)) was 0.85 pixel and 0.25 pixel, respectively.
Figure 2-12: The LANDSAT image of Paris. (a) The original image (contrast has been enhanced in images for presentation purposes). (b) with Gaussian noise ($\sigma^2 = 0.120$) added to that (PSNR = 11.29 dB, SSIM = 0.0330) (c) with salt and pepper noise ($d = 0.120$) added to that (PSNR = 14.09 dB, SSIM = 0.1075). The average registration error of the proposed algorithm in estimating the $[x, y]$ shift [3.250, 4.750], over 400 subimages of the noisy images that contained Gaussian noise (b) and salt and pepper noise (c) was 0.85 and 0.25 pixel, respectively.

To evaluate the integer and subpixel error metrics (defined in Section 1.1.3.3) and thus assess the algorithm accuracy, these two error metrics were calculated at each Gaussian noise level for the proposed algorithm (Figure 2-13(b)-(c)). Figure 2-13(a) and Figure 2-13(b) demonstrate the direct relation between the average registration error of the proposed algorithm and the integer error metric value at the same level of Gaussian noise. The mild slope of increase in the registration error in Figure 2-13(a) resulted in a gradual increase in the integer error metric from 1 to approximately 2. In addition, Figure 2-13(b) shows that the integer part of the proposed algorithm performed well in finding the correct integer shift between the two subimages, even in the presence of significant noise, by taking advantage of the inherent noise-robustness of SGDs. The average number of points with normalised SGGC values greater than 0.85 was less than two points for all subimages, even with appreciable Gaussian noise ($\sigma^2 = 0.120$) (Figure 2-13).

Nevertheless, for the subpixel error metric (Figure 2-13(c)), the values increased rapidly for low levels of Gaussian noise ($\sigma^2 < 0.01$), for which the subpixel accuracy was decreased from 0.0001 pixel to 0.2 pixel. For larger magnitudes of Gaussian noise ($\sigma^2 > 0.01$), the changes
in the subpixel error metric became smaller as the rate of decrease in subpixel accuracy was also reduced. This trend showed that this metric is sensitive enough to be an indication of the algorithm accuracy in estimation of subpixel shifts.

![Graphs showing subpixel image registration results](image_url)

Figure 2-13: (a) The average registration errors (pixel) in estimating the shift [3.250, 4.750] in 400 subimages of the LANDSAT image of Paris (128 pixel × 128 pixel) for the proposed method. (b) The average number of points with normalised SGGC values (Equation 2.9) greater than 0.85 as the error metric of the integer part of the proposed algorithm. (c) The plane fitting error to $\varphi_p$ data (Equation 2.11) as the error metric of the subpixel part of the proposed algorithm.

1.1.6 Image rotation tests and evaluation of the proposed algorithm with error metrics

In several fine registration algorithms, such as [62],[72], and [73], the registration algorithm is able to estimate the rotation using the pseudo-polar FFT. But the rotation estimation provided by these algorithms is not as accurate as the translational shift estimation [72], and is usually used to find the rotation when the rotation centre lies at the centre of the image [73]. However, in most practical applications, rotation is part of a non-rigid image transformation.
This type of rotation decreases the accuracy of rigid image registration algorithms, and pseudo-polar FFT is unable to estimate that correctly. The robustness of the algorithms was thus evaluated for estimating translational shift in the presence of rotations with the procedure which is described in Section 1.1.3.2.

Figure 2-14 shows an example of displacement vectors estimated by the proposed algorithm before and after applying 1° rotation to the LANDSAT image of Mississippi. Figure 2-15(a) shows the ideal rotation calculated mathematically (Equations 2.15 and 2.16) for the same image size. Figure 2-15(b)-(i) show the rotation pattern estimated by the proposed algorithm, Foroosh + HW, Foroosh et al. [49], Guizar-Sicairos et al. [55], Stone et al. [58], Vandewalle et al. [62], Feng et al. [61], Brox et al. [60], and Messerli et al. [53], respectively. As is clear from Figure 2-15(b)-(i), the rotation pattern estimated by the proposed algorithm is the closest pattern to the ideal rotation in Figure 2-15(a). Even though the method of Brox et al. [60] in Figure 2-15(i) had a pattern similar to the ideal pattern in Figure 2-15(a), the values were misplaced and inaccurate. The average registration error in this image was 0.0834 pixel for the proposed method, 0.1322 pixel for Foroosh + HW, 0.3068 pixel for Foroosh et al. [49], 0.2412 pixel for Guizar-Sicairos et al. [55], 0.9269 pixel for Stone et al. [58], 2.2078 pixel for Vandewalle et al. [62], 0.2751 pixel for Feng et al. [61], 0.8999 pixel for Brox et al. [60], and 0.1060 pixel for Messerli et al. [53].
Figure 2-14: Displacement vectors determined using the proposed algorithm for the $1^\circ$ rotation about the centre of the LANDSAT image of Mississippi.

Figure 2-15: (a) The ideal rotation pattern of $1^\circ$ rotation for the same size image. Rotation patterns were found by using the proposed method b), Foroosh + HW (c), Foroosh et al. [49] (d), Guizar-Sicairos et al. [55] (e), Stone et al. [58] (f), Vandewalle et al. [62] (g), Feng et al. [61] (h), Brox et al. [60] (i), and Messerli et al. [53] (j). The average error in this image was 0.0834 pixel for the proposed method (b), 0.1322 pixel for Foroosh + HW (c), 0.3068 pixel for Foroosh et al. [49] (d), 0.2412 pixel for Guizar-Sicairos et al. [55] (e), 0.9269 pixel for Stone et al. [58] (f), 2.2078 pixel for Vandewalle et al. [62] (g), 0.2751 pixel for Feng et al. [61] (h), 0.8999 pixel for Brox et al. [60] (i), and 0.1060 pixel for Messerli et al. [53] (j).
Table 2-4 summarises the $RMSE_R$ (Equation 2.17) of each algorithm for rotation values of 0.5°, 1°, 2°, and 3°. The proposed algorithm has the smallest average $RMSE_R$ for all rotations. Similar to translational shifts larger than 1 pixel in Table 2-2, Vandewalle et al. [62] and Stone et al. [58] methods failed to find the displacements for rotations larger than 1°. Guizar-Sicairos et al. [55] and Foroosh et al. [49] methods could perform relatively well up to 3° rotation, but failed to find the displacement beyond this angle. The optical flow method of Brox et al. [60] could not achieve good accuracy in measuring the displacements. The proposed method and methods of Foroosh + HW, Feng et al. [61], and Messerli et al. [53] achieved less than 1 pixel error in all the tested rotation values. Nevertheless, the average $RE_R$ of the proposed algorithm was 1.77, 2.31, and 1.95 times less than Foroosh + HW, Feng et al. [61], and Messerli et al. [53], respectively.

<table>
<thead>
<tr>
<th>Method</th>
<th>Registration errors (pixel) at rotations [θ degree]</th>
<th>Average (pixel)</th>
<th>Relative difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.5°</td>
<td>1°</td>
<td>2°</td>
</tr>
<tr>
<td>Vandewalle et al. [62]</td>
<td>0.4067</td>
<td>2.1874</td>
<td>4.9871</td>
</tr>
<tr>
<td>Stone et al. [58]</td>
<td>0.0313</td>
<td>0.4846</td>
<td>3.0053</td>
</tr>
<tr>
<td>Guizar-Sicairos et al. [55]</td>
<td>0.0968</td>
<td>0.1888</td>
<td>0.5842</td>
</tr>
<tr>
<td>Foroosh et al. [49]</td>
<td>0.1404</td>
<td>0.3447</td>
<td>0.9348</td>
</tr>
<tr>
<td>Foroosh + HW</td>
<td>0.0749</td>
<td>0.1276</td>
<td>0.3006</td>
</tr>
<tr>
<td>Feng et al. [61]</td>
<td>0.1063</td>
<td>0.1730</td>
<td>0.3548</td>
</tr>
<tr>
<td>Brox et al. [60]</td>
<td>0.4600</td>
<td>0.9086</td>
<td>1.8548</td>
</tr>
<tr>
<td>Messerli et al. [53]</td>
<td>0.0428</td>
<td>0.0856</td>
<td>0.2450</td>
</tr>
<tr>
<td>The proposed method</td>
<td>0.0574</td>
<td>0.0628</td>
<td>0.1456</td>
</tr>
</tbody>
</table>

Table 2-4: The average registration errors (in pixel) for all 2400 LANDSAT subimages (128 pixel × 128 pixel) ($RE_R$) for rotation. The final column provides the relative difference in the average registration errors quantified with respect to that of the proposed method.

Table 2-4 shows that the only cases in which algorithms had lower errors than the proposed method were for Stone’s and Messerli’s methods in the case of 0.5° rotation. The main reason for this arises from the difference between the SGGC in the integer part of the proposed algorithm and CC and OC algorithms in the Stone et al. [58], and Messerli et al. [53] methods, respectively. CC had difficulty with displacements larger than 1 pixel, as was also observed in the 1° rotation for the LANDSAT image of Mississippi, as shown in Figure 2-15(f). Nevertheless, since the average magnitude of the shift was 1.2821 pixel in 0.5° rotation, CC
performed well in the integer part of the Stone’s method [58]. In contrast, the dependency of
SGGC on image features in the $x$ and $y$ directions (Equations 2.8 and 2.9), means that this
method is sensitive to rotational changes of the image. However, for rotations larger than 0.5°,
the proposed method showed a clear advantage over Stone’s method [58], for which the $RE_R$
averaged over all rotations was 17.60 times larger than that for the proposed method. The
proposed method also outperformed the method of Messerli et al. [53] for rotations larger
than 0.5°, where the $RE_R$ of the proposed method averaged over all rotations was 1.95 times
smaller than that for Messerli’s method (Table 2-4).

As with the translational shift tests in the presence of Gaussian noise in Section 1.1.6, the
integer and subpixel error metrics (defined in Section 1.1.3.3) were evaluated in image rotation
tests. For this purpose, two LANDSAT images with different levels of image features (i.e.
Paris (Figure 2-5(a)) and Mississippi (Figure 2-5(b)) were selected. The LANDSAT image of
Paris has more features compared to the LANDSAT image of Mississippi. A rotation of
between 0.5° to 3° was applied to each image, and the algorithm displacement registration
error and the two error metrics were calculated at each rotation angle. The results are
summarised in Table 2-5 for the LANDSAT image of Paris and Table 2-6 for the LANDSAT
image of Mississippi. Table 2-5 and Table 2-6 indicate that, as expected from the rotation tests
in Figure 2-15 and Table 2-4, the integer part of the algorithm is more sensitive to rotation
than to Gaussian noise (Figure 2-13). However, the error metrics for the integer and the
subpixel part of the algorithm are still able to provide an indication of the algorithm
registration error in two different images with different textures. As an example, the error
metrics have similar values for rotation of 2° in Table 2-5 and rotation of 1° in Table 2-6 where
the algorithm registration error has also similar values (i.e. 0.0863 and 0.0834 in Table 2-5 and
Table 2-6, respectively). The same observation is valid for the example of 3° rotation in Table
2-5 and 2° rotation in Table 2-6.
Table 2-5: The error metrics for integer and subpixel parts of the proposed algorithm for the LANDSAT image of Paris

<table>
<thead>
<tr>
<th>Rotation (degree)</th>
<th>Error metric (integer part)</th>
<th>Error metric (subpixel part)</th>
<th>Displacement error (pixel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>1.7225</td>
<td>0.13766</td>
<td>0.0555</td>
</tr>
<tr>
<td>1</td>
<td>1.7250</td>
<td>0.18762</td>
<td>0.0379</td>
</tr>
<tr>
<td>2</td>
<td>1.9450</td>
<td>0.32536</td>
<td>0.0863</td>
</tr>
<tr>
<td>3</td>
<td>2.3350</td>
<td>0.51955</td>
<td>0.1934</td>
</tr>
</tbody>
</table>

Table 2-6: The error metrics for integer and subpixel parts of the proposed algorithm for the LANDSAT image of Mississippi

<table>
<thead>
<tr>
<th>Rotation (degree)</th>
<th>Error metric (integer part)</th>
<th>Error metric (subpixel part)</th>
<th>Displacement error (pixel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>2.0875</td>
<td>0.13777</td>
<td>0.0560</td>
</tr>
<tr>
<td>1</td>
<td>2.0350</td>
<td>0.19238</td>
<td>0.0834</td>
</tr>
<tr>
<td>2</td>
<td>2.6525</td>
<td>0.34667</td>
<td>0.2065</td>
</tr>
<tr>
<td>3</td>
<td>3.5250</td>
<td>0.54151</td>
<td>0.4031</td>
</tr>
</tbody>
</table>

However, the comparison of error metrics values in Figure 2-13 (for Gaussian noise tests) with Table 2-5 and Table 2-4 (for rotation tests) shows that these metrics should be used in combination to provide sufficient information about the algorithm accuracy. Although it is not possible to find the exact registration error of the algorithm based solely on these metrics, they can provide useful information about the algorithm accuracy in estimating the translational shift.

1.1.7 Computational complexity

The computational cost of the proposed method is considerably lower than methods using iterative optimisation processes (NR, GC), or upsampling methods (such as [55]). The proposed method is less computationally demanding for large image sizes compared to methods that involve intensity interpolation, such as the subpixel part of the method proposed by Messerli et al. [53] that uses bicubic interpolation, and local weighing of the NCC peak. Furthermore, the proposed method requires fewer computations than Tzimiropoulos’s method, which uses SVD and an optimisation algorithm [50]. The computation complexity of SVD and 2D FFT are $O(N^3)$ and $O(N^2 \log N)$, respectively. This implies 60.74 faster calculation of 2D FFT in comparison to SVD for $N = 128$. However, the proposed modified phase-base method requires one more 2D FFT than Stone et al. [58], but this does not significantly
increase the computational time since 2D FFT is fast, and this small additional effort is justified given that the proposed method provides considerably better performance (see Table 2-2, Table 2-3, and Table 2-4).

2.5 Conclusions

A new two-step method is presented for finding the integer and subpixel translational shifts between two images for image registration and deformation measurement. For finding the integer shift, SGGC (Equation 2.9) performed considerably better than other algorithms (Table 2-2, Table 2-3, and Table 2-4), even in the presence of high levels of Gaussian and salt and pepper noise (Figure 2-8 to Figure 2-11). The importance of choosing an appropriate windowing function was discussed in detail, and analysis in the frequency domain was performed to choose an appropriate window function for the integer and subpixel parts of the proposed algorithm.

For the subpixel translational shifts, the proposed phase-based method provides a quarter of the error compared to other phase-based methods [3, 14] (Table 2-2). This was due to the selected window function, pre-filtering, normalisation procedure, and the two-stage selection of the phase data.

Tests using 2400 subimages, for a range of subpixel to large translational pixel shifts, demonstrated that the proposed method significantly outperforms the state-of-the-art algorithms in this field (Table 2-2 and Table 2-3). In addition to the methods in Table 2-2 and Table 2-3, Tzimiropoulos et al. [50] proposed and tested a method, using the same images used in this study, and compared their results against those generated using the methods by Vandewalle et al. [62] and Foroosh et al. [49]. Tzimiropoulos et al. [50] reported an error of a few thousandths of a pixel, which was up to a 1/7 of Vandewalle’s method, and 1/21 of Foroosh’s method for subpixel shifts. In comparison, the average registration error for the proposed method was a few ten-thousandths of a pixel for both subpixel and large shifts (Table 2-2 and Table 2-3), which is on average 1/116 of Vandewalle’s method, and 1/356 of Foroosh’s method for subpixel shifts (Table 2-2). This implies a factor of over 16 times better
performance of the proposed method compared to that of Tzimiropoulos et al. [50] when is compared against the same methods applied to the same images of the same size (i.e. 128 pixel × 128 pixel).

The noise robustness of the proposed algorithm to Gaussian noise and salt and pepper noise was tested and was compared to other algorithms (Figure 2-8 to Figure 2-11). The proposed algorithm showed higher accuracies for both types of noise at low-level ranges of noise (i.e. PSNR 49.46 dB to 33.00 dB for Gaussian noise, and 61.17 dB to 37.55 dB for salt and pepper noise) (Figure 2-8 and Figure 2-9). This range of noise is typical for many existing camera systems. Even though phase-based methods are usually more sensitive to noise compared to template matching methods (such as the methods proposed by Feng et al. [61] and Messerli et al. [53]), the proposed algorithm could perform well in presence of high levels of noise (Figure 2-10 and Figure 2-11). The proposed algorithm and method of Feng et al. [61] had the lowest registration errors in presence of high-level of Gaussian noise (Figure 2-10), and it could perform well for high-level of salt and pepper noise (Figure 2-11). The method proposed by Messerli et al. [53] could only have a lower registration error for high-levels of salt and pepper noise (Figure 2-11).

A method was also proposed for evaluating the subpixel image registration algorithms to measure displacement in the presence of rotation. It was shown that the proposed algorithm outperformed existing state-of-the-art methods in finding the rotation pattern and displacements (Figure 2-15, Table 2-4).

The translation and rotation tests on 6 LANDSAT images with different textures demonstrated that the proposed method is able to find the correct subpixel shift in images with few features, for which other methods, such as [44][55][58], fail (Sections 1.1.4 and 1.1.6). The tests with noisy images in Section 1.1.5 indicated that the proposed algorithm has an accuracy of 0.85 pixel where many algorithms, such as [50] and [49], are unable to estimate the
correct shift. Furthermore, this method can find large shifts (Table 2-3), where algorithms such as NR and GB fail to reliably estimate the shift.

In addition, two error metrics were defined for the integer and subpixel part of the algorithm, and tests were performed to relate the algorithm accuracy in the presence of Gaussian noise (Figure 2-13) and image rotation (Table 2-5 and Table 2-6). It was shown that, although these error metrics cannot be used to find the absolute error of the algorithm, together they could provide beneficial information about the algorithm accuracy in estimating the translational shift.

The proposed method belongs to the category of area-based methods (i.e. correlation-like registrations) [42]. These methods use the intensity information of grey-scale images to find the shift and register the images. However, colour images could be registered using these techniques either by converting the images to grey-scale images, or by applying the algorithm on the each colour channel of the image. Although the proposed method is designed for measuring subpixel translational shifts, because of its high accuracy, low-complexity, and robustness to noise, this method can be used for subpixel image registration or deformation measurement in applications where both accuracy and computational efficiency are important. This method can also be incorporated into coarse-to-fine image registration techniques for non-rigid transformations.

In this chapter, the proposed algorithm was tested and compared against other methods using synthetic translational and rotational shifts. Further analysis on the performance of the algorithm on physical shifts and combinations of translation, rotation, and scaling is performed in [74] (Chapter 4) and [75] (Chapter 5).

Finally, the relatively small computational cost of this method and its parallelisable nature makes it suitable for hardware implementation in digital signal processors (DSPs), field-programmable gate arrays (FPGAs), and graphics processing units (GPUs) for near real-time applications. Appendix A reviews the suitability of recent DSPs, FPGAs, and GPUs for computer vision and image processing algorithms, and Appendix B describes the FPGA
Subpixel image registration

implementation of CC, which is a major part of the computation of this subpixel image registration algorithm.
Subpixel image registration
3 The Effect of Camera Settings on the Image Noise Level and Subpixel Deformation Measurements

The content of this chapter is based largely on the following journal paper, which is in review for the Optics and Lasers in Engineering journal:


3.1 Abstract

Measurements of deformations under controlled loads are one of the primary methods to assess the mechanical properties of materials. Surface deformations are often measured in stereoscopic systems using a technique called digital image correlation (DIC). Estimating the accuracy of deformation measurements in practical experiments is important, yet challenging in many applications. The measurement accuracy of DIC systems is influenced by several factors, in particular camera noise. However, camera noise is dependent on camera settings, such as the exposure time and gain. Moreover, the uncertainty of measurement is not directly proportional to the image noise. Hence, the accuracy of measurements ought to be estimated for the specific camera settings of the experiment and the algorithm used for measuring subpixel deformations. Nevertheless, the camera settings cannot be selected independently
because they influence properties of a DIC system, such as the depth of field, and the maximum frame rate, which limits the camera exposure time in high-speed DIC systems. Therefore, to simultaneously satisfy the DIC system requirements and maintain the accuracy of measurements, the trade-off between the camera noise and the system specifications ought to be considered.

In this chapter, four experiments were designed to estimate the effect of changing the camera settings on the image noise level, and the accuracy of subpixel deformation measurements. The results showed that the camera parameters directly influenced the image noise level, but the accuracy of algorithms in measuring deformations was not proportional to the image noise level. The result of this study can be used to quantitatively determine the relationship between camera settings and the error of a DIC system.

3.2 Introduction

Measuring surface deformations with a stereoscopic imaging system is the most commonly used method in non-destructive tests for assessing material properties in experimental mechanics [1]. In-plane projection of surface deformations are typically measured using a single camera, and multiple camera systems are used to measure the deformation data in three dimensions (3D). Digital image correlation (DIC) techniques are the most popular methods for finding surface deformations. For DIC, images are divided into a collection of small subimages, and the deformations are found by matching features of the subimages to those derived from a target image, under, for example, a different set of loading conditions. DIC techniques can achieve subpixel accuracy in deformation measurements [7]. Depending on the frame rates of the cameras, a DIC system can measure low-speed and quasi-static deformations, or high-speed dynamic deformations. Low-speed DIC systems have been used, for instance, to measure the thermal expansion of film specimens [76], to find deformation properties of asphalt mixtures [77], to identify defects in glass structures [78]. Examples of high-speed DIC include skin deformation measurements at 150 frames per second (fps) [79], rotary blade tests at 400 fps [46], dynamic punch tests at 100,000 fps [80], and the fracture behaviour of composites under a dynamic loading at 96,000 fps [81]. Several algorithms are
available for estimating subpixel deformations by comparing a reference image to a target image (refer to [43] for details). A random speckle pattern is often applied to the surfaces of the objects being tested to enable reliable intensity matching between the initial and target images.

DIC systems are often required to perform accurate measurements; hence, it is essential to quantify the accuracy of such systems. The measurement accuracy can be estimated by investigating the sources and relative contributions of errors. Wang et al. [82] quantified the effects of errors related to the intensity pattern noise and the subset size in the accuracy of DIC systems. In a more comprehensive study, Bornert et al. [83] assessed systematic errors associated with the speckle pattern, speckle size, subset size, grey level interpolation, shape functions, and optimisation algorithms. In addition to the errors related to the DIC algorithms, the errors associated with the camera system need to be analysed to estimate the system accuracy. The noise of a camera system is dependent on various parameters, such as sensor resolution, sensor quality, sensor temperature, lens quality, lighting conditions, and camera settings. Reu et al. [84] evaluated the effects of camera system resolution on the accuracy of a DIC system. Pan et al. [85] assessed the effect of camera temperature variations in the DIC measurement, and Grediac and Sur [86] addressed the effect of camera sensor noise on displacement measurements. In addition to DIC measurements, camera noise was characterised and modelled for some other applications, such as image denoising [87], high-dynamic-range imaging [87], and digital forensics [88].

The camera sensor noise can be reduced by lowering the temperature of the sensor, providing adequate illumination, or selecting appropriate camera settings. Among the camera settings, the $f$-number of the lens (the ratio of the lens focal length to the diameter of its pupil), the camera exposure time, and the camera gain are the major parameters that can be controlled to reduce the camera noise. However, the camera parameters are correlated, and often it is not possible to change them independently to achieve the desired condition. For instance, small $f$-numbers (i.e. large aperture sizes) increase the light intensity at the sensor, but also limit the
camera depth of field (refer to [79] for details). The desirable depth of field of a DIC system is determined by the working volume and the field of view (FOV), where wide FOVs require large $f$-numbers. The camera frame rate limits the maximum duration that the camera shutter remains open (i.e. exposure time). The relations between the camera parameters thus imply that, in high frame rate cameras, a large depth of field (or equivalently a large FOV) is only possible with high illumination. However, in many applications, it is not practical to provide intense illumination. Furthermore, the camera parameters are limited by the system constraints, such as the achievable ranges of depth of field, FOV, or frame rate. High-speed DIC systems often require a large depth of field and FOV, but short camera exposure times.

In a high-speed camera system, sufficient exposure can be achieved by controlling illumination intensity, lens aperture, exposure time, and sensor gain. Increasing sensor gain amplifies the camera sensor output. Note that sensor gain is analogous to ISO settings (i.e. the sensitivity of the camera sensor to the light) in conventional photography. However, it may not be advantageous to increase the exposure time and sensor gain. Long exposure times increase the image noise level by increasing the photon shot noise of the sensor, and cause blurring of moving objects. Increasing the sensor gain simultaneously scales the sensor output and noise, which results in brighter but noisier images compared to low sensor gains. Therefore, the image noise level is directly influenced by camera parameters, such as the exposure time and gain. It is known that increasing the sensor gain and exposure time increases the image noise, but their contribution to the image noise level, and their ultimate effects on the accuracy of subpixel deformation measurements, are poorly characterised in the literature. In this work, this issue was addressed and the influence of camera parameters on the image noise level, and on the accuracy of subpixel deformation measurements was investigated. The main purpose of the present study is to identify an appropriate set of camera parameters to optimise the accuracy of deformation measurement in a DIC system.

This chapter is organised as follows. The sources of noise in digital cameras are discussed in Section 3.3. The experimental setup and the camera used in this study are introduced in Section 3.4. Section 3.5 outlines the set of experiments designed to estimate image noise levels
The Effect of Camera Settings on the Image Noise Level and Subpixel Deformation Measurements

when exposure time, sensor gain, illumination intensity, and frame rate are varied. In Section 3.6, algorithms used for measuring subpixel deformations are introduced, and a method is proposed to measure the accuracy of these algorithms in estimating the deformations in camera images. The results are presented in Section 3.7, while Section 3.8 is dedicated to the discussion and conclusion.

3.3 Noise in Digital Cameras

Current camera sensors are typically based on two main technologies: Charge-coupled devices (CCD), and complementary metal–oxide semiconductor (CMOS) technology. CCD sensors have a long history of use in digital cameras, whereas CMOS is a more recent and cost-effective technology [89]. CCD and CMOS technologies use photodiodes to convert incident photons to electronic charge, but use different processes to convert the electronic charge to a digital signal. CCD sensors have a limited number of amplifiers to convert the charge to voltage (usually only one) while in CMOS sensors, each pixel has its own amplifier (refer to [89] for a detailed description). The quality and performance of modern CMOS sensors have improved considerably, such that the noise-level of recent CMOS sensors is now comparable with CCD sensors. Even though CCD and CMOS sensors use different technologies, they have similar sources of error: thermal noise, photon shot noise, and fixed pattern noise, described below:

- Thermal noise arises from the stochastic behaviour of electrons freed by thermal energy in the sensor. The freed electrons cause a fluctuation in the electric charge (known as the dark current), which randomly increases the image intensity values, independent of incoming light photons. Thermal noise has a Gaussian distribution [87, 90].

- Photon shot noise results from the stochastic nature of photons arriving at a given pixel. The number of incoming photons hit the camera sensor with a Poisson
distribution, which causes a random variation (noise) in the values. Photon shot noise is thus directly related to the time duration that sensor is exposed to the light [91].

- Fixed pattern noise is a time-invariant spatial noise that is non-uniformly distributed over the pixels. Fixed pattern noise is usually caused by spatial variations in sensitivity of sensor photo detectors [92], dust, or defective pixels.

Among these sources of noise, thermal noise is typically the dominant source. Camera noise can thus be approximated by a Gaussian distribution, particularly for short exposure times.

3.4 Experimental setup

A monochrome CMOS USB 3 camera (Flea3 FL3-U3-13Y3M-C, Point Grey, Canada) [17], with maximum image resolution of 1280 pixel × 1024 pixel, and a maximum frame-rate of 150 fps at its full image size, was used in all experiments. The camera was a part of a stereoscopic DIC system designed for measuring skin deformations (Figure 3-1) [79]. A rigid speckled disc was used as a sample object for DIC tests, and four high power (750 lm) current-driven LEDs were placed at four sides of the disc to provide a uniform constant illumination on the surface of the test sample.

The cameras provide sensor gain values in the range of 0 dB to 18 dB. The sensor gain refers to the scale factor by which, the electric charge of each pixel is converted to a voltage through its on-board analogue amplifier. The output voltage approximately doubles with each 3 dB increase in sensor gain.
3.5 Measuring the camera noise

The performance of subpixel deformation estimation methods can be evaluated under various noise conditions. A simple method for estimating the temporal noise of pixels is provided by measuring the variation about the average intensity values of many images taken under identical conditions [87]. However, this method cannot estimate non-uniformly distributed noise nor constant components of noise (such as fixed pattern noise), and changes in the illumination will often be seen as noise. Furthermore, it is difficult to replicate noise conditions for use in new experiments. These limitations can be avoided by blind noise estimation and measuring the spatial noise level of individual images. Typical blind noise estimation algorithms estimate the variation of intensity values in small image regions (patches) as the image noise, assuming that the image intensity values are constant in small image patches. Yet this assumption does not hold for rich-textured images, such as speckled surfaces in DIC systems, and will lead to an overestimation of the noise level. Liu et al. [93] proposed a blind noise estimation method that addresses this issue by estimating the noise from low...
textured patches. This method can perform well in the presence of speckles, and was thus chosen for image noise estimation in the current study.

The following four experiments were designed to determine the effect of changing the camera parameters on the image noise level:

E1) With the camera lens cap attached, images were recorded as the camera gain was changed from 0 dB to 18 dB in 2 dB increments. At each gain value, the camera exposure time was changed from 0.1 ms to 16 ms in increments of 0.1 ms.

E2) With the lens cap removed, and during uniform illumination, the camera gain and exposure time were changed in a similar way to the first experiment. For high gain values, the exposure time was increased until the image intensity started to become saturated in some parts of the image.

E3) With the lens cap removed, the gain was kept constant while the exposure time and illumination were changed to reach to a mean pixel intensity of approximately 110 (out of 255). The selected mean pixel intensity value provided sufficient contrast in the images of the experimental setup to perform DIC.

E4) With the lens cap removed, the illumination was kept constant, the camera gain was changed from 0 dB to 18 dB in 2 dB increments, and the exposure time was changed to achieve a mean pixel intensity of approximately 110.

In the first and second experiments (E1 and E2), one image was taken at each set of camera gain and exposure time values, whereas in the third and fourth experiments (E3 and E4), 20 images were taken at each set of values to measure the variations of the noise level. In all these four experiments, the image noise level was measured in single images using the method of Liu et al. [93].

Experiment E1 was designed to determine the camera sensor thermal noise (also known as dark current noise), and fixed pattern noise. In experiment E2, the lens cap was removed, thus including photon shot noise into the image noise. Experiment E3 was designed to measure
The Effect of Camera Settings on the Image Noise Level and Subpixel Deformation Measurements

independently the effect of exposure time and illumination on the image noise level at a constant gain. The mean pixel intensity of approximately 110 was chosen to provide sufficient contrast at the region of interest of the test sample for measuring deformations. Finally, experiment E4 revealed the effects of changing both the camera gain and exposure time on the accuracy of algorithms that measure subpixel deformations.

3.6 Subpixel deformation measurements

Several methods have been used to measure subpixel shifts [43]. The effect of camera noise on the measurement accuracy was evaluated using three algorithms developed by Guizar-Sicairos et al. [55], Foroosh et al. [49], and HajiRassouliha et al. [35] (described in chapter 2) were selected. These algorithms demonstrated an average error of 0.281 pixel, 0.057 pixel, and 0.00026 pixel, respectively, in a previous study examining subpixel translational shifts [35]. As suggested in [35], a Hann window (HW) was added to the method of Foroosh et al. [49] to improve its accuracy (Foroosh + HW). Foroosh + HW showed an improved average error of 0.015 pixel for finding subpixel translational shifts compared with the 0.057 pixel of the original algorithm of Foroosh et al. [49].

It is challenging to use real-world deformations to investigate the effects of real camera noise on the accuracy of subpixel deformation measurements. A translational shift is the simplest form of deformation that can be applied to an object using a translational stage. However, the accuracy of measurements for a single camera is dependent on several factors, including the resolution of the translational stage, the estimation of the relative angle between the translational stage and the camera sensor, and the camera lens distortion. To account for these factors in the estimation of the accuracy, the FFT-shift method was chosen to apply synthetic translational shifts on the images. This choice was based on a previous study, which showed that the FFT-shift can closely resemble real-world experiments, and real physical translational shifts [70]. The camera noise was estimated from several images taken of a rigid, non-moving test sample under the same conditions, where one image was synthetically shifted,
and one image from the remaining set was used as the reference image. In the tests, the shifted image had a translational shift of \([x, y] = [3.250, 4.750]\) pixel and the algorithms for measuring subpixel shift were used to estimate the displacement between the shifted image and the reference image. The subimage size for finding the shift was 128 pixel × 128 pixel in all algorithms, and the centres of subimages were distributed over the speckled part of the test sample in the images with a step increment of 20 pixel (Figure 3-2).

![A sample subimage](image)

**Figure 3-2:** The distribution of subimage centres over the speckled surface of the test sample. The subimage size was selected to be 128 pixel × 128 pixel, and the distance between adjacent subimage centres was 20 pixel. A sample subimage is indicated on the surface of the test sample.

The error of a given algorithm in estimating the shift of a subimage was defined to be the Euclidean distance between the translation estimated using that algorithm \([\hat{x}, \hat{y}]\), and the specified applied synthetic translation \([x, y]\) (Equation 3.1).

\[
\text{Registration error} = \sqrt{(x - \hat{x})^2 + (y - \hat{y})^2}
\]  \hspace{1cm} 3.1

The registration errors for the subimages were averaged to quantify the error for each algorithm in estimating the shift in that image. The median value of the registration errors over 20 image pairs, taken in experiment E3 and experiment E4, was selected as the registration error for the algorithm.
The Effect of Camera Settings on the Image Noise Level and Subpixel Deformation Measurements

The registration errors for the noise-free images were assessed in a similar manner, by applying a synthetic translational shift to one of the images, and using each algorithm to estimate the translation between the synthetically shifted image and the original image.

3.7 Results

Figure 3-3 shows the noise variance of the images taken in experiment E1. In the absence of light, the major source of noise is from thermal effects. Figure 3-3 indicates that the increase in exposure time had little effect on the image noise level at gains lower than 10 dB. Above 10 dB, the camera exhibited an increase in noise with longer exposure time. This may have been caused by the addition of a digital gain alongside the analogue gain at high gain values. The digital gain multiplies the output voltage by a constant value to increase the image brightness, but this comes at the cost of an increase in the noise.

Figure 3-3: The noise variance of images taken with the lens cap on (experiment E1 described in Section 3.5).

To investigate the effect of light on the image noise level, experiment E2 was performed with the lens cap off. Figure 3-4 shows the noise variance of the images taken during this
experiment. For gain values higher than 0 dB, the exposure time was increased until the image intensity values started to saturate. Figure 3-4 illustrates that the increase in the image noise level for longer exposure times was less than the increase caused by increasing the gain. By comparing plots in Figure 3-3 and Figure 3-4, it can be seen that the noise variances for the lens cap on and off were similar at gain values lower than 8 dB, indicating that the camera did not have significant light-dependent noise at those lower gains. However, longer exposure times increased the image noise level, which may have been due to an increase in photon shot noise that is dependent on the light intensity. The gradient of the noise variance with respect to exposure time increased as the gain was increased.

Figure 3-5 shows an expanded view of the changes in the image noise variance at 0 dB gain from Figure 3-4. The image noise increased with increasing exposure to 6 ms, and levels off during longer exposures (a similar trend was present for gain values higher than 0 dB). This suggests that, in this experiment with exposure times shorter than 6 ms (i.e. the frame rate up to 167 fps), the intensity values of some of the noisy pixels saturated to zero (the lower band of the intensity values in the noisy image is limited to zero). However, the intensity value was not constant for the different exposure times in this experiment, while previous studies have reported that most of the algorithms for measuring subpixel deformations are sensitive to the changes in illumination [43, 93]. To address this, experiment E3 was performed to take images at a same mean value of intensity to explore the effects of exposure time on the image noise level and the registration errors of each algorithm. Figure 3-6 shows the image noise variance for 20 images taken with a mean brightness of approximately 110 and exposure times ranging from 19 ms to 62 ms. As illustrated, the image noise level had increased at each exposure time, but with a relatively consistent variation (i.e. 10 % increase from its lowest value at an exposure time of 19 ms to its highest value at 62 ms).
Figure 3-4: The noise variance of images taken with the lens cap off (experiment E2). For gain values higher than 0 dB, the exposure time was increased until the image intensity became saturated in parts of the image.

Figure 3-5: The noise variance of images taken with the lens cap off, and with an exposure time of up to 16.3 ms, at a constant gain of 0 dB. The image noise level had small variations for exposure times above 6 ms.
Figure 3-6: The noise variance of images taken at a mean pixel intensity of approximately 110 (experiment E3), and exposure times ranging from 19 ms to 62 ms. 20 images were taken at each exposure time.

The 20 images taken for experiment E3 were designed to have the same brightness level at various exposure times. Figure 3-7 shows the effect of changing the camera exposure time on the registration error for the four algorithms described in Section 3.6. The algorithms of Foroosh et al. [49], Foroosh + HW, and HajiRassouliha et al. [26] resulted in a small increase in error with increasing exposure time. The errors for these three algorithms were increased by approximately 10 % from their lowest values at 19 ms to their highest values at 62 ms, consistent with the increasing image noise level with exposure (Figure 3-6). However, the error for the algorithm of Guizar-Sicairos et al. [55] did not have a noticeable change across the range of exposure times.
To extend experiment E3 to a wider range of camera settings, the effect of camera gain and exposure time on the accuracy of subpixel shift measurements was investigated using experiment E4. Figure 3-8 shows the noise variance for the images with the same mean pixel intensity of approximately 110 taken at various gains and exposure times. The exposure time was 2.57 ms at 18 dB (i.e. maximum 389 fps) and 18.8 ms at 0 dB (i.e. maximum 53 fps). The image noise variance increased nonlinearly from an average of 0.92 at 0 dB gain to an average of 5.44 at a gain of 18 dB. Thus, the consequence of increasing the camera frame rate from 53 fps to 389 fps was a five-fold increase in the image noise. Figure 3-9 shows the registration error for the four algorithms described in Section 3.6 for measuring subpixel shifts on images without noise, and images at the minimum and maximum camera noise for a mean pixel intensity of approximately 110 shown in Figure 3-8. This test showed that the accuracy of the algorithms for measuring subpixel shifts was not proportional to the image noise level (Figure 3-9). The accuracy of the algorithms of HajiRassouliha et al. [35] and Foroosh + HW was affected to a greater extent by the image noise in comparison to the other two algorithms. Nonetheless, the algorithm of HajiRassouliha et al. [35] had a significantly lower error.
compared to the other three algorithms. Despite the better performance of Foroosh + HW over the original algorithm of Foroosh et al. [49] under noise-free conditions, no significant improvement was demonstrated when compared using the images taken at minimum camera noise level. The algorithm of Guizar-Sicairos et al. [55] suffered the least noise-related reduction in accuracy. While the algorithm of Guizar-Sicairos et al. [55] was less accurate than Foroosh et al. [49] and Foroosh + HW in noise-free images, it performed better in the presence of camera noise.

Figure 3-8: The image noise level at combinations of exposure times and gain values that provided mean pixel intensity of approximately 110 in images (experiment E4). The exposure times and gain values were ranging from 2.57 ms to 18.80 ms, and 0 dB to 18 dB, respectively.
The Effect of Camera Settings on the Image Noise Level and Subpixel Deformation Measurements

![Error Graph](Image)

Figure 3-9: The registration error for subpixel shift measurements from images without noise, images with a noise variance of 0.92 (at 0 dB gain, and the exposure time of 18.80 ms), and images with a noise variance of 5.44 (at 18 dB gain, and the exposure time of 2.57 ms). The mean pixel intensity was approximately 110 in all of the images.

The effect of changing the camera parameters on the accuracy of the algorithms described in Section 3.6 was further investigated for the images in Figure 3-8 at combinations of gain values and exposure times that provided images with mean pixel intensity of approximately 110. The exposure times and gain values were ranging from 2.57 ms to 18.80 ms, and 0 dB to 18 dB, respectively. Figure 3-10 shows the registration error for these algorithms. The results of Figure 3-10 illustrated that the algorithm of HajiRassouliha et al. [35] had the best performance across all of the image noise levels and camera parameters. The algorithm of Guizar-Sicairos et al. [55] had the least change in error and performed better than the method of Forooosh et al. [49], and Forooosh + HW. While adding a Hann window to the algorithm of Forooosh et al. [49] (i.e. Forooosh + HW) improved the accuracy of this algorithm, the improvement was insignificant for high noise levels.

The algorithm of Guizar-Sicairos et al. [55] was the least sensitive to noise. This is due to its method for finding the subpixel shift, where images are upsampled in the frequency domain.
Although upsampling is relatively insensitive to noise, it limits accuracy (Figure 3-10), and is computationally demanding, particularly for upsampling values larger than 20.

Figure 3-10: The registration error of subpixel shift measurements for combinations of exposure times and gain values that provided mean pixel intensity of approximately 110 in images (experiment E4). The exposure times and gain values were ranging from 2.57 ms to 18.80 ms, and 0 dB to 18 dB, respectively. The noise levels of these images were increased from 0.92 to 5.44 (see Figure 3-8).

The exposure times in Figure 3-10 could be converted to their equivalent frame rates to find the relation between the frame rates and the errors of the algorithms. The maximum frame rate of this experiment was 389 fps at 18 dB gain, and the minimum frame rate was 53 fps at 0 dB gain. The result in Figure 3-10 illustrated that, by increasing the frame rate from its minimum (53 fps) to its maximum (389 fps), the errors were increased by 2.2 times in the algorithm of HajiRassouliha et al. [35], 2.5 times in Foroosh + HW, 1.85 times in the algorithm of Foroosh et al. [49], and did not change noticeably for the algorithm of Guizar-Sicairos et al. [55]. Note that the image noise level increased approximately six times over this range (Figure 3-8).

3.8 Discussion and conclusion

The accuracy of algorithms for measuring surface deformations is dependent on the camera settings, image noise level, and the method that the algorithms use for finding the subpixel
The Effect of Camera Settings on the Image Noise Level and Subpixel Deformation Measurements
deflections. The first experiment of this study (E1) showed that increasing the camera gain
substantially affected the image noise level (Figure 3-3). In this case, the thermal noise was the
dominant image noise source.

The second experiment (E2) showed that an increase in exposure time increased the image
noise level (Figure 3-4, Figure 3-5), but this increase was not as great as the effect of increasing
the gain in Figure 3-3. However, the image intensity level may have affected the estimation of
the image noise level, and the accuracy of subpixel deformation measurements.

The result of the third experiment (E3) showed that the image noise level increased slightly
with increasing exposure time (Figure 3-6). It was further shown that increasing the exposure
time from 19 ms to 62 ms increased the measurement error by only approximately 10 % for
the algorithms proposed by Foroosh et al. [49], Foroosh + HW, and Hajirassouliha et al. [35],
and did not noticeably change the error for the algorithm of Guizar-Sicairos et al. [55] (Figure
3-7).

The fourth experiment (E4) investigated a range of exposure times and gains and their
effects on the camera image noise, and the accuracy of subpixel shift measurements. Results
in Figure 3-8 showed that the image noise level at a constant mean intensity of approximately
110 increased by approximately six times where the gain was increased from 0 dB to 18 dB,
and the exposure time was decreased from 18.8 ms to 2.57 ms. However, the increase in image
noise was not proportional to the accuracy of measuring subpixel shifts for the algorithms
tested. The increase in image noise influenced the methods of Hajirassouliha et al. [35] and
Foroosh + HW more than the algorithms proposed by Foroosh et al. [49] and Guizar-Sicairos
et al. [55] (Figure 3-9). For example, even the minimum noise of the camera used in this study
(noise variance of 0.92) significantly increased the error of the methods of Hajirassouliha et
al. [35] and Foroosh + HW compared with the case using noise-free images (Figure 3-9).
However, these two algorithms had a significantly more accurate measurements in noise-free
images compared to the algorithms proposed by Foroosh et al. [49] and Guizar-Sicairos et al.
Discussion and conclusion

[55] (Figure 3-9). This observation motivates the use of intrinsically low-noise cameras, such as cameras with high-quality sensors or cooled-sensor cameras, for the methods of HajiRassouliha et al. [26] and Foroosh + HW to improve the accuracy of measurements.

This study showed that the proposed method for evaluating the performance of deformation measurement algorithms in the presence of camera noise can provide useful information for selecting appropriate camera settings. The camera settings are often forced by the system requirements. For example, the camera f-number is often restricted by depth of field requirements, and it is sometimes practically infeasible to provide more than a certain illumination. In these cases, the camera gain and exposure time should be selected to both satisfy the system constraints and to minimise the measurement errors. Figure 3-10 shows how the accuracy of displacement measurements was affected by frame rate changes from 53 fps to 389 fps at a mean pixel intensity of approximately 110. The results suggest that the effect of camera noise on the measurement accuracy is dependent on the algorithm that is used for measuring deformations. For this reason, the advantage of low-noise cameras may be negated by the selection of certain deformation measurement algorithms, such as algorithms proposed by Guizar-Sicairos et al. [55] and Foroosh et al. [49]. By contrast, the use of low-noise cameras is justifiable for accurate algorithms, such as the subpixel image registration method of HajiRassouliha et al. [35].

The selection of the mean pixel intensity of approximately 110 in experiments (E3 and E4) was based on the contrast of the test sample in the camera images of the experimental setup. However, as results of experiment (E2) in Figure 3-4 and Figure 3-5 showed, changes in the image contrast (caused by changing the exposure time in this experiment), did not have a great impact in the noise level. Therefore, the results of experiments (E3 and E4) will be similar to Figure 3-8 to Figure 3-10 at other mean pixel intensity values, as far as the image intensities were not saturated or under exposed.

The noise measurements performed in this study could be used to define the parameters of an appropriate camera noise model, such as the noise models used in [88, 94, 95]. The camera noise model would allow analytical determination of the optimal camera settings.
The Effect of Camera Settings on the Image Noise Level and Subpixel Deformation Measurements

The results of this chapter illustrate that using Flea3 CMOS cameras, the subpixel image registration method of HajiRassouliha et al. [35] (described in chapter 2) can achieve a maximum accuracy of approximately 0.0065 pixel at 0 dB gain, and a minimum accuracy of approximately 0.0143 pixel at 18 dB gain (Figure 3-10) in 128 pixel × 128 pixel subimages.
4 Subpixel Measurement of Living Skin Deformation Using Intrinsic Features

The content of this chapter is based largely on the following conference paper:


4.1 Abstract

Accurate measurement of skin deformation is essential to study and understand its behaviour under mechanical load. Digital image correlation (DIC) techniques are commonly used for in vivo subpixel measurements of deformation using camera-based devices. However, most of the existing DIC methods have modest accuracy and require the addition of feature rich textures in order to measure the deformations. These limitations have made it challenging to measure skin deformations using DIC, especially where the skin does not have a rich texture, or when high measurement accuracies are required. In this chapter, the accuracy and applicability of the phase-based Savitzky-Golay gradient-correlation (P-SG-GC) algorithm described in chapter 2 was tested for measuring subpixel deformations of living skin. As discussed in chapter 2, this algorithm addressed many of the limitations of existing subpixel
image registration algorithms, and its advantages could lead to new advances in measuring skin deformation.

Experiments were performed using a camera, and a flat object attached to a linear translational stage. A series of translational shifts were applied to the object using the linear stage and were measured using P-SG-GC. The result showed that P-SG-GC could successfully estimate translational shifts ranging from 0.05 pixel to larger than 20 pixel (physical shifts from 5 µm to 2000 µm) in a 64 pixel × 64 pixel subimage. The standard deviations of the measurements for translational shifts ranged from 0.008 pixel to a maximum of 0.045 pixel in the camera images (i.e. 0.8 µm to 4.5 µm). The P-SG-GC algorithm was used to measure skin deformation over an approximately 100 mm × 100 mm field-of-view. Results showed that P-SG-GC was capable of measuring skin deformations ranging from subpixel values to more than 19 pixel using only the intrinsic features of skin. The results illustrate that P-SG-GC is a robust, efficient, and accurate algorithm that can significantly improve the methods of measuring deformation distributions of living skin.

4.2 Introduction

Characterising the behaviour of skin is important in a number of applications. For example, quantification of skin properties can provide a better understanding of the wound healing process [27], the effects of ageing [96], the process of wrinkle formation [97], as well as improved methods for surgical planning [98]. Skin is a complex tissue that is anisotropic, heterogeneous, nonlinear, and viscoelastic [17]. Several methods have been developed to characterise the complex mechanical behaviour of skin in vivo or in vitro, such as during biaxial [12], compression [13], suction [14], or indentation [15] tests on tissues. Measuring the skin surface deformation is an essential part of all of these methods. To address this, various devices have been designed and built for measuring skin deformations in vivo. Some use sensor technologies, such as three-degree-of-freedom tactile devices [99], microrobots [100], motion capture systems [101], while others use image processing techniques for measuring deformations, such as single cameras [102, 103], multi-view stereo [41], and handheld stereoscopic devices [79].
Digital image correlation (DIC) is the most common image processing method used for measuring skin deformation with subpixel accuracy. DIC is a technique in which images are divided into small overlapping subimages, and the displacement is found for each subimage separately [43]. The DIC technique typically includes two main steps for first finding the integer shift, and then the subpixel shift between two subimages. The integer part of most of the existing DIC algorithms uses cross-correlation (CC) to find the shift between two subimages. However, CC frequently fails on images that are poorly textured. Therefore, speckled patterns or markers are often added to the surface of the skin to enable the shift to be estimated using CC [41, 102, 103]. Furthermore, the accuracy and efficiency of the subpixel part of many existing algorithms are limited. To address these limitations, a new method for subpixel image registration was recently proposed. This method is based on a phase-based algorithm that uses Savitzky-Golay differentiators in gradient correlation (P-SG-GC) [35] (described in chapter 2). It has been shown that the P-SG-GC algorithm can achieve accuracies better than 0.0002 pixel (60 times better than the state-of-the-art algorithms) in finding synthetically applied shifts to 2400 standard 128 pixel × 128 pixel subimages [35]. Furthermore, P-SG-GC is computationally efficient and performs well in low-textured images, where most of the existing algorithms fail to find the displacement [35].

This chapter investigates the performance of P-SG-GC for finding the deformation of living skin. First, an experimental setup was arranged to measure the accuracy of P-SG-GC in finding translational shifts of a target ranging from subpixel values to large shifts. Then, P-SG-GC was used to measure the deformation of skin using only intrinsic features.

4.3 Method

The accuracy of P-SG-GC in measuring translational shifts was evaluated by the experimental setup shown in Figure 4-1. A photographic stand was used to hold a single monochrome CMOS USB 3 camera (Flea3 FL3-U3-13Y3M-C, Point Grey, Canada) in a position perpendicular to the surface of a flat disk (target). A speckle pattern was applied to
the surface of the flat disk using an airbrush. The camera was equipped with a 12.5 mm lens (Fujinon, Japan) and the target was attached to a ball bearing manual linear stage with an accuracy of 2 µm over a 25 mm travel range (M-423 [104], Newport, U.S.A.).

![Figure 4-1: The experimental setup used to evaluate the accuracy of P-SG-GC in measuring translational shifts. A photographic stand was used to hold the camera in a position perpendicular to the surface of a flat object attached to a linear translational stage.](image)

A series of translational shifts (approximately 5 µm, 10 µm, 100 µm, 500 µm, 1000 µm, 1500 µm, and 2000 µm) were applied to the target using the translational stage. The camera captured images from the initial and shifted positions of the target. These were divided into approximately 2700 64 pixel × 64 pixel subimages, which were distributed uniformly across the surface of the target with a step increment of 10 pixel. The P-SG-GC algorithm was used to measure displacements between all the subimages of the initial and shifted images in the $x$ and $y$ directions ($d_x$ and $d_y$). The total displacement for each subimage was calculated using Equation 4.1:

$$D(n) = \sqrt{d_x^2 + d_y^2} \text{ (pixel)} \quad (n = \{1, 2, 3, \ldots, N\})$$

4.1
where, \( n \) is the subimage number, and \( N \) is the total number of subimages. The mean of the \( D(n) \) values (MD) (Equation 4.2) was chosen as the estimated value of the P-SG-GC algorithm for the applied translational shift.

\[
MD = \frac{\sum_{n=1}^{N} D(n)}{N} \text{ (pixel)} \tag{4.2}
\]

If we assume an ideal condition at each applied translational shift, including a perfect perpendicular position of the camera and a completely flat target, all the \( D(n) \) values should be equal. Therefore, the standard deviation of the \( D(n) \) values at each translational shift was considered to be an estimate of the accuracy of the P-SG-GC algorithm in measuring the shift. In addition, the linearity between the set of applied physical shifts (in \( \mu m \)) and the estimated \( MD \) shifts was considered to be an indication of the linearity of shift measurements.

The estimation of physical shifts using a camera system is subject to various sources of error. Error sources that can be compensated for include the effects of optical distortions of the camera lens, especially the radial distortion (also known as barrel distortion) [105]. The lens radial distortion causes straight lines to become curved in the camera images (similar to being mapped around a sphere), which affects the estimates of displacements from the images. To minimise this effect, lens distortion was compensated in the images and the computations were repeated. Lens distortion removal (i.e. undistortion of the images) was performed using radial and tangential lens distortion coefficients estimated through the camera calibration process explained in Chapter 8.

After the P-SG-GC algorithm was tested, this algorithm was used to measure deformations of living skin. For this purpose, two images were taken of the skin on the back of a volunteer’s hand at two different deformation states due to the movement of the volunteer’s thumb. To provide a similar condition to that used in the evaluation step, the volunteer’s hand was positioned at the same location as the target used for the validation experiments.
4.4 Results and discussion

Figure 4-2 shows the difference between the $D(n)$ values (Equation 4.1) and the MD values (Equation 4.2) estimated using the P-SG-GC algorithm applied to original (distorted) image in the left column and undistorted images in the right column for shifts of 500 µm, 1000 µm, 1500 µm, and 2000 µm. The difference values are colour-coded and are in units of pixel. As can be seen in Figure 4-2, the MD values were all larger than 1 pixel for all these shifts and a distinct pattern is evident for the original (distorted) images in the left column. This pattern is indicative of lens radial distortion that compresses the periphery of the image towards the optical centre (i.e. the principal point) [105]. Because of this effect, the measured shifts around the edges of the original (distorted) images were smaller than their actual values. The shift independent effect of lens distortion is shown in Figure 4-3, where the P-SG-GC algorithm was applied to determine the displacement field between one of the images and its undistorted version.

The comparison between the left and right columns of Figure 4-2 illustrates the improvement in the displacement estimates by correcting for lens distortion. Not only were the spatial distributions more uniform in the images of the right column compared with the left column, but the degree of underestimation of the displacement magnitudes was also reduced. The values in undistorted images revealed a distribution of displacements that is likely to be related to a relative tilt between the camera sensor and the target surface. This is an issue because the displacement measurements were performed in 2D by assuming that the camera axis was perpendicular to the surface of the flat object, the accuracy of which was difficult to determine using the experimental setup. Furthermore, the camera calibration error in finding the lens distortion coefficients was not considered in these measurements.
Subpixel Measurement of Living Skin Deformation Using Intrinsic Features

![Original (distorted) images](image1)

- **shift = 500 µm**
  - (a) MD = 5.11 pixel
  - (b) MD = 5.14 pixel

![Undistorted images](image2)

- **shift = 1000 µm**
  - (c) MD = 10.21 pixel
  - (d) MD = 10.26 pixel

- **shift = 1500 µm**
  - (e) MD = 15.29 pixel
  - (f) MD = 15.37 pixel

- **shift = 2000 µm**
  - (g) MD = 20.403 pixel
  - (h) MD = 20.5 pixel

![Color scale](image3)
Results and discussion

Figure 4-2: The \((D(n) – MD)\) values measured using the P-SG-GC algorithm applied to the original and undistorted images. The values are colour-coded and are in units of pixel. The target was moved downward using the linear translational stage, and the shift magnitudes are indicated on the left.

Figure 4-3: The displacement map caused by the lens distortion model on the same area as the images in Figure 4-2. The displacement values were estimated by applying the P-SG-GC algorithm to one of the images and its undistorted version. Displacement magnitudes are colour-coded in a logarithmic scale and values are in units of pixel.

Figure 4-4 shows the linearity of the relation between translational shift (5 µm, 10 µm, 100 µm, 500 µm, 1000 µm, 1500 µm, and 2000 µm) and the \(MD\) values for the original (distorted) and undistorted images. As illustrated in Figure 4-4, the estimates from the P-SG-GC algorithm show a linear correlation with the physical shift for both the original (distorted) and undistorted images \(((1 – R^2) = 5.2 \times 10^{-4}\) for the original images and \((1 – R^2) = 5.6 \times 10^{-4}\) for undistorted images).

Figure 4-5 shows the standard deviation of the estimated \(D(n)\) values (Equation 4.1) at the different translational shifts estimated from the original images and undistorted images. The standard deviation of the estimates based on the undistorted images was approximately half of that estimated using the original (distorted) images. This illustrates that the removal of the lens distortion helped to increase the measurement accuracy.
Figure 4-4: The linearity between the $MD$ values estimated using the P-SG-GC algorithm, and the translational shifts of the target using the linear translational stage.

Figure 4-5: The standard deviation of displacements estimated for subimages of the original (distorted) and undistorted images using the P-SG-GC algorithm.

The standard deviations of the measurements in Figure 4-5 were less than 0.01 pixel for shifts less than 1 pixel (at 5 µm, 10 µm, and 100 µm shifts it was 0.008 pixel, 0.008 pixel, and 0.009 pixel, respectively). The standard deviation increased for displacements larger than 1 pixel, but the values were small compared with the displacement. For example, the standard deviation of the P-SG-GC measurements for the estimation of the 2000 µm shift was 0.0735 pixel for the original (distorted) images and 0.0456 pixel for undistorted images (0.36 %).
and 0.22% of the \( MD \) values, respectively). The low standard deviation of the measurements (Figure 4-5), and the high degree of linearity between the estimated values and actual physical shifts (Figure 4-4), demonstrate that the P-SG-GC algorithm could accurately measure both small and large displacements.

The P-SG-GC algorithm was used to measure skin deformation on a volunteer’s hand. As shown in Figure 4-6, the volunteer’s hand was placed at a similar location to that of the target, and an image was taken of the initial state of the skin of the back of the hand (Figure 4-6(a)). The volunteer then moved their thumb to slightly deform the skin of the back of the hand (Figure 4-6(b)). As evident in Figure 4-6, the skin in this test did not have obvious intrinsic features, and no extra pattern was applied to the skin. The P-SG-GC algorithm was used to estimate the skin deformations from Figure 4-6(a) and Figure 4-6(b). The result of the measurements for 64 pixel × 64 pixel subimages is shown in Figure 4-7. The magnitude of the displacements is colour-coded, and the arrows indicate the localised direction of skin movement. The skin displacements ranged from subpixel values (0.01 pixel) to values larger than 18 pixel. As can be seen in Figure 4-7, the displacement fields show a continuous gradient over the surface of the skin. The result illustrates that the P-SG-GC algorithm could estimate the skin deformation using only intrinsic features. These measurements cannot be validated directly with currently available methods, since most of them fail to measure skin deformation using only intrinsic features.
Figure 4-6: The images of a volunteer’s hand used in this study to test the capability of the P-SG-GC algorithm in finding the skin displacements using intrinsic features. The volunteer was asked to move their thumb in Figure 4-6(b) to deform the skin of the back of the hand.

Figure 4-7: The estimation of the skin deformation from the images of Figure 4-6 using the P-SG-GC algorithm and 64 pixel × 64 pixel subimages. The magnitude of the displacements is colour-coded and the arrows show the directions of displacement.
4.5 Summary

Measuring skin deformation is an important step for developing biomechanical models of skin. Existing methods cannot provide sufficient accuracy [99–101], or require the addition of textured patterns to the skin [41, 102, 103]. In this chapter, these limitations have been addressed using a novel subpixel image registration algorithm (P-SG-GC) [35]. An experimental setup was created (Figure 4-1) for validating the P-SG-GC algorithm with a flat target and a linear translational stage. A series of translational shifts (5 µm, 10 µm, 100 µm, 500 µm, 1000 µm, 1500 µm, and 2000 µm) were applied to the target, which were then measured by comparing the original and undistorted initial and shifted images using P-SG-GC with 64 pixel × 64 pixel subimages. The lens distortion effects (Figure 4-3) were corrected in the images, which resulted in more accurate displacement measurements (Figure 4-2).

The results showed a high degree of linearity between the physical and estimated shifts (Figure 4-4). Furthermore, the measurements had small standard deviation values compared to the applied displacements (Figure 4-5). For instance, the standard deviation of measuring 5 µm and 10 µm shifts in 64 pixel × 64 pixel subimages was 0.008 pixel. These results indicate the high accuracy of the P-SG-GC algorithm. Increasing the subimage size from 64 pixel × 64 pixel to 128 pixel × 128 pixel would further decrease this error, although this would also decrease the spatial resolution of the displacement estimates.

The measurement errors estimated in this chapter (i.e. 0.008 pixel standard deviation) are close to the estimated range of the P-SG-GC algorithm error for noisy camera images using Flea 3 cameras in chapter 3 (i.e. 0.0065 pixel at 0 dB gain), despite that the subimage size of the measurements in this chapter was smaller than the subimage size used in chapter 2 (the accuracy of measurements decrease in small subimages).

The P-SG-GC algorithm was used to find the deformation of the skin of a volunteer’s hand (Figure 4-6). The result showed that P-SG-GC could measure the skin displacements from subpixel values (0.01 pixel) to values larger than 18 pixels (Figure 4-7) using 64 pixel × 64 pixel
subimages. Even though the skin in this example did not have obvious image features, P-SG-GC could estimate the displacements using only intrinsic textures.
5 Motion Correction Using Subpixel Image Registration

The content of this chapter is largely based on the following conference paper:


5.1 Abstract

Several methods have been proposed to correct motion in medical and non-medical applications, such as optical flow measurements, particle filter tracking, and image registration. In this chapter, experiments were designed to test the accuracy and robustness of the phase-based Savitzky-Golay gradient-correlation (P-SG-GC) algorithm proposed in chapter 2. In this case, the algorithm was used to correct the relative motion of the object and camera in pairs of images.

Experiments were performed using a camera, a flat object, a manual translational stage, and a manual rotational stage. The P-SG-GC algorithm was used to detect the relative motion of the object between reference and shifted images using a set of control points on the surface of the object, which were automatically matched in subimages of 128 pixel × 128 pixel. A least-squares method was used to estimate the image transformation matrix that best registered the shifted image and the reference image.
The results demonstrated that the P-SG-GC algorithm can accurately correct for the relative motion of the object and camera for a large range of applied shifts with a registration error of less than 1 pixel. Furthermore, the P-SG-GC algorithm could detect the images in which the motion could not be corrected due to poorly matched control points between the reference and shifted images. It was concluded that the P-SG-GC algorithm is an accurate and reliable algorithm that can be used to correct for object or camera motion.

5.2 Introduction

Motion artefacts between pairs of images can arise due to unwanted movements of either the imaging device or the imaging target. Motion artefacts are problematic in many applications. In medical images, breathing and movement of patients can cause distortions and artefacts that can confound diagnosis. Motion artefacts are also an issue in camera-based systems, especially where it is necessary to have a stabilised recording. For example, it is challenging to use a hand-held stereoscopic device (such as that described in [79]) to record surface deformations of living skin, due to movement of the subject’s limbs or the stereoscopic device. Correcting for such relative motion can improve the analysis of medical images and can increase the accuracy of measurements. For example, human knee cartilage mapping was improved after motion correction [106]. Motion correction has been performed using several methods, such as optical flow [107], particle filter tracking [108], adaptive block motion vectors filtering [109], and, most commonly, image registration [110–115]. However, most of the existing methods for subpixel image registration lack accuracy or robustness to large shifts. In this chapter, the accuracy and robustness of the P-SG-GC algorithm to correct motion artefacts was tested. The P-SG-GC algorithm is computationally efficient and performs well in poorly textured images [35]. It is thus a suitable algorithm for motion correction in real-time applications.

To test the P-SG-GC algorithm, a variety of manual translational and rotational shifts were applied to a flat object imaged using a camera. The relative shifts between pairs of images were estimated using the P-SG-GC algorithm with a set of control points (centres of subimages of size 128 pixel × 128 pixel) in the reference and shifted images. The displacements of these
control points were used to estimate affine and projective image transformation matrices using a linear least squares method. Each shifted image was registered to the reference image, and the registration error was calculated to indicate the accuracy of the P-SG-GC algorithm in correcting the motion. In addition, affine and projective transformations were compared to determine the degree of perspective distortions in the images, and whether adding the projection vector to the image transformation matrices can improve the registration accuracy.

5.3 Method

Figure 5-1 shows the experimental setup used to test the ability of the P-SG-GC algorithm in measuring rigid motion. A single monochrome CMOS USB 3 camera (Flea3 FL3-U3-13Y3M-C, Point Grey, Canada) equipped with a 12.5 mm lens was attached to a photographic stand in a position perpendicular to the surface of a flat target object. The target was attached to a manual linear translational stage, which was itself attached to a manual rotational stage. The combination of the two stages enabled the application of rigid in-plane shifts to the target, and the camera stand enabled moving the camera in the direction perpendicular to the surface of the object (Figure 5-1) to provide a scaling effect in the camera images.
Method

Figure 5-1: The experimental setup used to test the accuracy of the P-SG-GC algorithm to correct motion artefacts.

The following five experiments (E1 to E5) were designed and performed to test the ability of the P-SG-GC algorithm in detecting and measuring the relative motion of the target and camera:

E1. Translational shifts of the target: the target was shifted in 1 mm increments to a maximum of 5 mm (i.e. five translational shift values) using the translational stage.

E2. Rotational shifts of the target: The target was rotated in 0.5° increments to a maximum of 3° (i.e. six rotational shift values) using the rotational stage.

E3. Translational and rotational shifts of the target: the target was shifted in 1 mm increments, and at each increment was rotated by 0.25° to a maximum of 5 mm and 1.25° (i.e. five combinations of rotational and translational shift values) using the translational and rotational stages.

E4. Translational shifts of the camera: the camera was shifted in the direction perpendicular to the surface of the target in 2 mm increments to a maximum of 10 mm (i.e. to provide five scaling effects) using the photographic stand handle (Figure 5-1).
E5. Translational shifts of the camera and the target: the camera was shifted as described in experiment E4, but in 1 mm increments, and at each increment 0.5 mm translational shifts were applied to the target using the translational stage (i.e. to provide five combinations of scaling and translation).

The images of experiments E1 to E5 were divided into subimages of size 128 pixel × 128 pixel to measure localised motion of the object. These subimages were distributed uniformly across the surface of the target with a step increment of 20 pixel. The subimage size limits the maximum shift that the algorithm could identify. The number of steps and the maximum shift values in experiments E1 to E5 were chosen according to the pixel size, the subimage size, and the ability of the P-SG-GC algorithm to estimate that type of motion. Note that this algorithm was developed for registration of images with translational shifts, but here it was tested in a wider set of test conditions.

In the first step, the P-SG-GC algorithm was used to measure shifts between all subimages of the reference and shifted images in the x and y directions \(d_x\) and \(d_y\) at each step of experiments E1 to E5. The centres of the subimages in the reference image \(C_i\) were considered as control points. Thus, the corresponding control points (i.e. centres of subimages) in the shifted image \(C_s\) are given by:

\[
C_s(n)[x] = C_i(n)[x] + [d_x(n)]
\]

where, \(n\) is the subimage number, and \(N\) is the total number of subimages. The accuracy of the P-SG-GC algorithm in estimating \(d_x\) and \(d_y\) values for subimages of an image (and consequently \(C_s(n)\)) depends on several factors, including the subimage texture level, the magnitude of subimage shift, and the nature of the shift between the subimages. The integer error metric of the P-SG-GC algorithm was used as an indication of the level of confidence in estimating the shift between subimages (the integer error was defined the number of points around the peak of the correlation function. Details are in [35]). The threshold value of the
integer error was set to 4 points to provide an acceptable accuracy, and the control points with an error less than this threshold were considered as control points with a precise match (i.e. \( \check{C}_s(n) \) and \( \check{C}_i(n) \)).

The next step for correcting the motion between the reference and the shifted images is to find a geometric image transformation \((T)\) that registers the images. Equation 5.2 shows the relation between the control points in the reference and the shifted image that can be used for registration [66].

\[
\check{C}_s(n) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = T \times \check{C}_i(n) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (n \in \{1, 2, \ldots, M\})
\]

where, \(T\) is a \(3 \times 3\) matrix given in Equation 5.3 [66], and \(M\) is the number of subimages that had an acceptable match (i.e. integer errors less than 4 points).

\[
T = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & 1 \end{bmatrix}
\]

In the transformation matrix, \(T, [a_3, a_6]\) is the translation vector, \([a_7, a_8]\) is the projection vector, and \([a_1, a_2, a_4, a_5]\) defines rotation, scaling and shearing [66]. The values of these elements identify the type of transformation, i.e. rigid, similarity, affine, projective, or some combination of these transformation types. Considering the applied shifts in experiments E1 to E5, affine and projective image transformations are suitable for the registration of the images in this study. Affine transformations are able to describe translational, rotational, scaling, and shearing differences between the images. However, the projection vector is zero for affine transformations, which thus preserve the parallelism between lines and cannot correct perspective effects. This issue was solved in projective transformations by adding the projection vector to the transformation matrix of affine transformations. In this way, projective transformations are able to correct perspective effects, at the cost of introducing two additional parameters to the projection vector (i.e. \([a_7, a_8]\)). The elements of \(T\) \(([a_1, \ldots, a_8])\) were estimated using a least-squares method based on control points from the
Motion Correction Using Subpixel Image Registration

reference and shifted images. The least-squares method was only used for the images in which the percentage of the control points with an acceptable match ($\hat{C}_s(n)$ and $\hat{C}_i(n)$ in Equation 5.2) was more than 80% of the total number of control points ($C_s(n)$ and $C_i(n)$ in Equation 5.1). Otherwise, this method detects that the number of matched control points are insufficient for finding an accurate transformation. This will help to avoid an incorrect registration of the images. The 80% ratio was selected based on the output of displacement measurements in the test images. However, it is an approximate value that could be selected within a range without significantly changing the results.

Two methods were used to assess the accuracy of registration between the reference and shifted images. The first method was a qualitative assessment performed by overlaying the registered and the reference images and subtracting their intensity values. The second method was a quantitative method for estimating the registration error. The 2D subpixel displacements ($D$) between the control points of the registered and the reference images ($\hat{C}_s(n)$ and $\hat{C}_i(n)$ in Equation 5.2) were estimated using Equation 5.4, and were averaged in all the subimages to find the registration error (Equation 5.5) ($M$ is the number of subimages that had an acceptable match).

$$D(n) = \sqrt{d_x^2 + d_y^2} \text{ (pixel)} \quad (n \in \{1,2,\ldots,M\}) \quad 5.4$$

$$RE = \frac{\sum_{n=1}^{M} D(n)}{M} \text{ (pixel)} \quad 5.5$$

The accuracy of motion correction in a camera system is affected by optical distortions of the camera lens, especially the radial distortion [105]. Lens distortion causes non-uniform pixel displacements over the image. Thus, the appearance of an object changes when the object is positioned in different locations of a camera image. This effect is evident in the estimation of registration error. This effect was reduced by undistorting the images using radial and tangential lens distortion coefficients estimated through a camera calibration process [105].
5.4 Results and discussion

The results of experiments E1 to E5 are presented in Figure 5-2 to Figure 5-6. In each figure, the reference image and the final image for that particular experiment were overlaid following and prior to correction to provide a qualitative indication of the accuracy of the P-SG-GC algorithm to correct the relative motion of the target and camera. The intensity values of the reference and the final images for each experiment were subtracted and were colour-coded with and without motion correction to illustrate the effect. The registration error (Equation 5.5) was found at each step of experiments E1 to E5, and is provided as a quantitative estimate of the accuracy of the P-SG-GC algorithm. The vertical axis of the graphs in Figure 5-2(e) to Figure 5-6(e) shows the average displacement of all subimages for that particular applied shift. In the final image of experiment E3, the control points with an integer error less than 4 points ($\hat{C}_s(n)$ and $\hat{C}_t(n)$ in Equation 5.2) were less than 80 % of the total number of control points. The P-SG-GC algorithm thus detected that the number of control points was insufficient for correcting the motion in this case. For this reason, the result of the final image of this experiment is not presented in Figure 5-4.
Figure 5-2: The results of the translational shift in experiment E1. The final image for this experiment (5 mm shift of the target) is overlaid on the reference image without (a) and with (b) motion correction. In the overlaid images in (a) and (b), red colour was used for the reference image, green colour was used for the shifted image, and yellow colour was used for areas of similar intensity between the two images. The subtraction of the intensity values for the reference and final images without and with motion correction is shown in (c) and (d), respectively, and the values are colour-coded. The registration error is shown in (e).

Figure 5-3: The results of the rotational tests in experiment E2 involving a 3° rotation of the target. See Figure 5-2 caption for explanation of the panels.
Results and discussion

Figure 5-4: The results of the translational and rotational tests in experiment E3 involving a 5 mm translation and $1.25^\circ$ rotation of the target. See Figure 5-2 caption for explanation of the panels. The P-SG-GC algorithm detected that the proportion of matched control points with an acceptable error was insufficient for correcting the motion in the final image of this experiment (i.e. 5 mm translation and $1.25^\circ$ rotation of the target). For this reason, the result does not include the final image.

Figure 5-5: The results of the camera shifts in experiment E4 involving a 10 mm shift of the camera. See Figure 5-2 caption for explanation of the panels.
Figure 5-6: Results of the camera and translational shifts in experiment E5 involving a 5 mm shift of the camera and 2.5 mm shift of the target. See Figure 5-2 caption for explanation of the panels.

The comparison of Figure 5-2(e) to Figure 5-6(e) illustrates that the affine and projective transformations had very close registration errors, indicating that the perspective effects of the images were not large in experiments E1 to E5. This was also observed in the projective transformation matrices, in which the elements of the projection vectors had very small values.

Overall, the registration errors were less than 1 pixel for all motion correction tests in Figure 5-2(e) to Figure 5-6(e), except for the fourth image of experiment E3 (Figure 5-4), for which the registration error was 1.52 pixel in an average displacement of 45.2 pixel (the relative error was 0.034).

Comparison of Figure 5-2(e) to Figure 5-6(e) shows that the P-SG-GC algorithm performed best when correcting translational shifts in experiment E1 (i.e. 0.04 pixel error for 55.3 pixel average shift (relative error = 0.00072)). The largest registration error was for the combination of object translation and rotation in experiment E3 (i.e. 1.52 pixel for 45.2 average shift (relative error = 0.034). This finding was expected, since the P-SG-GC algorithm
was developed for subpixel registration of images with translational shifts. Despite this, the performance of the algorithm for rotational shifts and scaling of the images would be acceptable in many applications.

The experiments of this study (E1 to E5) were designed to cover typical scenarios in motion correction problems. Even though the shifts were applied using translational and rotational stages, it is expected that the P-SG-GC algorithm would perform similarly in correcting motion in practical applications.

5.5 Conclusion

Image registration is the most common technique for correcting for motion in applications that require an accurate registration, such as medical images, and image and video stabilisation [110–115]. In this chapter, the capability of the P-SG-GC algorithm described in Chapter 2 was tested using a series of experiments with a camera, a flat object, a translational stage, and a rotational stage. The P-SG-GC algorithm was used to estimate the motion in a series of control points in subimages of size 128 pixel × 128 pixel. Affine and projective transformations were determined using control points from subimages and a linear least-squares method, and were applied to the shifted images to register them to the reference images. It was found that the perspective effects were not large in these types of motions and the addition of projection vectors to the image transformations cannot increase the registration accuracy.
6 An Accurate and Fast Algorithm for Subunit Registration of Arbitrary Dimensional Data

6.1 Abstract

Registration of arbitrary dimensional data has a wide range of applications. For example, digital image correlation (DIC) and digital volume correlation (DVC) are well-known subpixel and subvoxel techniques for measuring surface and volume deformations. Accuracy of measurement and computation speed are two important measures of image registration algorithms. Several methods have been proposed to increase the accuracy or speed of registration, particularly for two-dimensional (2D) registration. However, three-dimensional (3D) registration presents more challenges than 2D registration. For instance, the computation costs of 3D registration are substantially higher than 2D registration, low-contrast signals are common and their contrast cannot be easily increased, and most algorithms cannot handle large shifts or need to have a good initial estimate of the shift. To overcome these limitations, an accurate and fast algorithm for subunit registration of arbitrary dimensional data was proposed. The multidimensional phase gradient registration algorithm (PGreg) is a non-iterative two-step method that estimates integer and subunit shifts in two separate parts. A method named filtered cross-correlation (FCC) is proposed for the integer part of the algorithm, and subunit shifts are found using a phase-based method. The PGreg algorithm is applicable to arbitrary dimensional data, and has input parameters that can be tuned or selected.
for specific applications. Despite its generality, only 2D and 3D versions of the PGreg algorithm with fixed input parameters were tested here.

Three sets of 2D/3D images, with various contrast ranges, were synthetically generated and used to test the PGreg algorithm to compare against state-of-the-art algorithms. The 2D version of the PGreg algorithm was also tested on a standard image (Landsat image of Paris, a sample image in the Matlab image processing toolbox [116]). Results showed that the PGreg algorithm was able to find large shifts and shifts of low-contrast images. Furthermore, the PGreg algorithm could achieve substantially smaller registration errors than existing methods. The average registration errors of the PGreg algorithm using noise-free (16 voxel)$^3$, (32 voxel)$^3$, and (64 voxel)$^3$ test subimages were $0.0112 \pm 0.0033$ voxel, $0.00311 \pm 0.0011$ voxel, and $0.0015 \pm 0.00040$ voxel, respectively. The PGreg algorithm is able to provide significantly higher accuracies at lower computation costs than the competing state-of-the-art algorithms.

6.2 Introduction

Several methods have been proposed to register two-dimensional (2D) or three-dimensional (3D) images [42, 117, 118]. Subpixel image registration and subpixel deformation measurement are of particular interest in applications where accurate registration of the images, or precise measurement of small deformations, is required. Many of the methods that register images to subpixel accuracy are based on a two-step process, where the integer and the subpixel/subvoxel shifts are found in two separate steps. The most common method of finding the integer shift is to find the peak in the cross-correlation (CC) or normalised CC (NCC) of the two images. The subpixel/subvoxel shifts are found by various methods, such as upsampling in the spatial domain or the frequency domain [55, 56], fitting to a function [50], using phase-based methods [35], and using iterative methods and shape functions [59].

Digital image correlation (DIC) is the best known iterative method for finding subpixel deformations in images [119]. Digital volume correlation (DVC) (also known as volumetric-DIC) is an extension of DIC for measuring 3D subvoxel deformations (or displacements) in
An Accurate and Fast Algorithm for Subunit Registration of Arbitrary Dimensional Data

volumes [1]. DVC has been used to quantify displacements of human soft tissues, such as the brain, to inform computational models [18]. Furthermore, DVC is an essential part of a widely used technique to assess the structural and mechanical behaviour of materials, in which the test material is deformed under an applied force while a device records images before and during the deformations. X-ray computed tomography (CT) is the most common imaging modality used to capture volumetric deformations. For example, the DVC technique has been applied to X-ray CT or micro-CT images to assess the mechanical behaviour of foams [19], composites [20], bones [4, 5], scaffold implants [21], polymer bonded sugars [22], and bread crumbs [120]. The DVC technique was also used to study crack initiation and propagation in X-ray CT and synchrotron radiation laminography images [9, 121, 122], and to measure thermal properties in X-ray CT images [123]. However, the use of DVC in low-quality 3D imaging technologies is restricted because of the limitations of current DVC techniques. For instance, DVC was used for the first time in 2013 to measure elastic stiffness from optical coherence tomography (OCT) images [40].

Sutton and Hild [8] summarised some of the challenges of current DVC techniques in a recent paper, including: performing accurate grey level interpolations; selecting a suitable shape function; and processing the enormous amount of data in a short time. Furthermore, DVC algorithms require a good initial estimate of the parameters [36]. These challenges arise from the inherent limitations of the existing techniques used for DVC. Most of the existing DVC techniques use the 3D-CC of volumes to find the integer shift and an iterative nonlinear optimisation to find the subvoxel shifts [124]. The computational costs of DVC algorithms are $G \times S$ times greater than 2D-DIC algorithms for grid point spacing of $G$ pixel (voxel) and a subset size of $S$ pixel$^2$ (voxel$^3$) over the image (inside the volume) [36], and are thus slow. For example, Gates et al. [37] used parallel computing with 8 processors and could only reduce the computation time from 45.7 hours to 5.7 hours to compute a DVC of size 41 voxel $\times$ 41 voxel $\times$ 41 voxel in a grid consisting of (39 $\times$ 39 $\times$ 39) points. This limitation becomes more problematic for high-resolution images and for large amounts of data.
Furthermore, the CC used in the DVC algorithms is sensitive to noise and changes in the image illumination, and fails with images that have poor texture or have undergone large deformations [34, 35]. Thus, the use of DVC was limited to measuring small deformations in rich-textured volumes [34]. To improve registration, CC was replaced with NCC in some methods [4, 5, 40, 124]. However, although NCC has some advantages over CC in dealing with changes in image illumination, it does not address several other limitations of CC. For instance, NCC is substantially more computationally demanding than CC, and it performs poorly with large deformations. Pan et al. [125] addressed the latter limitation by proposing an incremental DIC method to update the reference image to measure large deformations, but their method considerably increased the computation time.

The limitations of CC and NCC have been addressed by some 2D methods, such as gradient-correlation (GC) proposed by Argyriou et al. [51], and phase-correlation (PC) proposed by Foroosh et al. [49]. GC combines the central differences of intensity values in the two coordinate directions to form a single complex image by multiplying one real subimage by \( i \) and adding it to the other real subimage. This approach allows the information in two real values to be encoded as a single complex value. PC uses the normalised cross-power spectrum of the intensities of two images to find the shift between them. Since PC and GC do not directly use the intensity values of the images, they are both more robust than CC at finding shifts in images with poor texture [49, 51]. GC was later used in 2D subpixel registration algorithms proposed by Tzimiropoulos et al. [50, 73]. HajiRassouliha et al. [35] proposed using Savitzky-Golay differentiators instead of the central differences in the GC algorithm to improve its noise-robustness. The phase-based Savitzky-Golay gradient-correlation (P-SG-GC) method of HajiRassouliha et al. [35] was used to measure subpixel shifts. Previous tests on standard images showed that the registration errors of PC and GC subpixel image registration algorithms in estimating translational shifts of subimages of size 128 pixel \( \times \) 128 pixel ranged from 0.01 pixel for the PC algorithm to 0.0002 pixel for the P-SG-GC algorithm [35]. Even though the P-SG-GC algorithm substantially improved the accuracy of image registration compared to the other competing methods, it was limited to
An Accurate and Fast Algorithm for Subunit Registration of Arbitrary Dimensional Data

2D image registration. This limitation arises from the structure of the GC algorithm that only allows the combination of two dimensions into a single complex image.

Some approaches were proposed to address the shortcomings of CC and NCC in DVC applications where the test volume had large deformations [34, 126]. Franck et al. [34] proposed a method called stretch-correlation to address the limitations of DVC in measuring large deformations. Stretch-correlation was implemented in the Fourier-domain using the fast Fourier transform (FFT) of the volumes. Stretch-correlation takes the stretch of subimages into consideration in a polar coordinate system by decomposing the deformation gradient tensor into the orthogonal rotation and the symmetric right-stretch tensors, assuming small rotations and shears. However, the stretch-correlation method [34] can only improve the registration performance in large deformations where the volume is stretched, not when subimages are displaced or shifted. Recently, Bar-Kochba et al. [126] proposed an iterative DVC approach and a weighting function for CC coefficients to measure large deformations. The method of Bar-Kochba et al. [126] used an approach similar to that proposed by Pan et al. [125] for 2D applications, which was to start with a large subimage, measure the deformations, warp the two volumes, measure the error between the volumes, and decide based on an error threshold whether to continue to another iteration with a smaller subimage or to stop the process. The purpose of using a weighting function in the method of Bar-Kochba et al. [126] was to increase the resolution of displacement fields by weighting the high frequencies. The weighting function of the method of Bar-Kochba et al. [126] was adapted from the method of Nogueira et al. [127], which was proposed for particle image velocimetry. Bar-Kochba et al. [126] identified and removed the outliers from the CC output at each iteration, and found subvoxel shifts in their method by fitting a bivariate Gaussian function to the peak of the final cross-correlation output. Although using the weighting function in the iterative method of Bar-Kochba et al. [126] improves the performance of CC at large shifts, it cannot eliminate the low-pass behaviour of the CC. Furthermore, the approach of Bar-Kochba et al. [126] is a computationally intensive iterative process.
Wein et al. [128] proposed a volume gradient correlation method to overcome the shortcomings of CC when performing 2D-3D registrations between low-contrast images and 2D projected volumes. Gradient correlation uses the mean of the NCC values of two coordinates of the gradient images of 2D projected volumes [128]. Volume gradient correlation was shown to perform better than NCC for 2D-3D registration in medical images [129], and was able to register clinical 3D CT data to 2D X-ray images where CC failed [130]. Even though volume gradient correlation performs better than CC and NCC in low-contrast applications, it is only applicable in 2D cases, and cannot be used for DVC. However, dealing with low-contrast volumes is a big challenge in DVC applications, since it is difficult to increase the contrast by adding a random speckle pattern, as is performed in 2D cases [1].

To address the limitations of registration algorithms in dealing with large deformations or low-contrast data, a method named filtered cross-correlation (FCC) was proposed to register arbitrary dimensional data. This method was extended to subunit registration based on a previously developed phase-based method for subpixel image registration [35]. The 3D version of the PGreg algorithm was tested on synthetically generated 3D images (i.e. volumes) with various contrast ranges, and the 2D version of the PGreg algorithm on an example of a standard image (Landsat image of Paris [116]), as well as synthetically generated 2D images with various contrast ranges. The 2D version of the PGreg algorithm was compared to the P-SG-GC algorithm [35] that has been shown to outperform competing algorithms in performing 2D registration [35]. The 3D version of the PGreg algorithm was compared to the 3D-CC, which is the most widely used method for performing DVC.

6.3 Method

The proposed registration method is a two-step process. In the first stage, the relative integer displacement of two N-dimensional (N-D) datasets is found to within one unit along each dimension, and is used to shift back the deformed N-D data to the initial state. In the second step, the relative displacement of the two N-D datasets, which were now displaced to within less than one unit of each other, is found to within a fraction of a unit along each dimension.
The integer and the subunit parts of the PGreg algorithm are described in sections 6.3.1 and 6.3.2, respectively.

### 6.3.1 Estimating integer shifts

The CC of the intensity values of an initial image with its shifted version can be regarded as a moving-average operator [126, 131]. This is due to the properties of the convolution operator, which is integrated into both CC and moving-average filters [131]. Moving-average filters are low-pass filters that have smoothing properties and attenuate frequencies higher than around 0.2 of the sampling frequency [132]. However, much of the information in an image is stored in its fine details, which are represented by high frequency components. This becomes more important in low-contrast images in which low frequencies are mainly influenced by the DC component of the signal. The main reason that a weighting function is used in the method of Bar-Kochba et al. [126], and that the gradient operator is used in the integer part of the algorithms in [35, 50, 73], is that they emphasise high frequencies. In the integer part of the GC algorithm for a 2D image with orthogonal coordinate directions $x$ and $y$, the image gradient in the $x$ direction ($G_x$) and the image gradient in the $y$ direction ($G_y$) are combined to form a single complex image ($CI$) by multiplying $G_y$ by $i$ and adding it to $G_x$ (Equation 6.1).

$$CI(x, y) = G_x(x, y) + G_y(x, y) \times i$$  \hspace{1cm} 6.1$$

This approach allows the information in two real values to be encoded into a single complex value ($CI$). A complex discrete FFT is then applied to the resulting $CI$. The advantage of combining two real images into a complex image is that a single complex FFT can be computed more efficiently than two real FFTs. Disadvantages with this approach include: it can only be applied to two dimensional images (i.e. it cannot be used for one-, three-, and higher-dimensional images); it assumes that the registration objective function is formed as the CC of the sum of the first derivatives of subimages with respect to the image axes; and the gradient operation is applied to the images in the spatial domain. In the P-SG-GC algorithm
Method

[35], the Savitzky-Golay differentiator is used to estimate $G_x$ and $G_y$. The benefit of using the Savitzky-Golay differentiator is that it provides a balance of reducing noise through smoothing, approximating the first derivative through differencing, while retaining a relatively easy to compute, integer-valued, small filter kernel. However, since the computational cost of the convolution operation in the spatial domain is directly proportional to the kernel size, it is desirable to use a small kernel size in the P-SG-GC algorithm.

The FCC algorithm (Figure 6-1) can overcome the shortcomings of both the GC and CC algorithms. The left diagram in Figure 6-1 is the procedure for performing CCs in the frequency domain using the FFT. This involves taking N-D FFT and multiplying the discrete Fourier transform (DFT) of subsets of the initial data by the DFT of the conjugate subset displaced data. The box on the right side of Figure 6-1 is the novel part of the FCC algorithm. In this section, any number of N-D filter kernels can be designed either in the spatial domain or the frequency domain (or both). The designed filters are then squared and summed by taking advantage of the linearity of the FFT prior to being applied to the output of the CC of the images in the left diagram in Figure 6-1. This is equivalent to applying filters independently to each of the N-D subset datasets. An advantage of the FCC algorithm over the GC methods (such as [35, 50, 73, 128, 130]) is that, because the filter kernels are represented in both the spatial and frequency domains and are multiplied by the outputs in the frequency domain, there is much greater freedom to tailor the characteristics of the filters with little or no impact on computational complexity. For instance, removal of the DC component of the subset N-D data, which is typically performed in the spatial domain, could be transformed to the frequency domain where the zero frequency element of the squared frequency domain filter can be set to zero. Furthermore, the filters can be precalculated or recalled from a library, as they do not directly depend on the data. Decoupling the calculation of the kernels from the convolution pipeline can provide significant savings in computational cost compared to the convolution-based methods such as [35, 50, 73, 128, 130], leading to a simpler and faster algorithm that eliminates the need for any expensive convolution operations. Furthermore, this method enables the use of a combination of the frequency domain and/or the spatial
domain filters. The frequency domain representations of the filters are squared in the precalculation stage before applying them to the DFT outputs (Figure 6-1). This squaring operation mimics the effect of convolving filter kernels (such as the gradient operator) with images in the methods based on GC ([35, 50, 73, 128, 130]). While in the GC methods in [35, 50, 73] $G_x$ and $G_y$ were combined in a complex number (Equation 6.1), and in the GC methods in [128, 130] $G_x$ and $G_y$ were averaged, better performance can be achieved when the registration objective can more effectively combine information about the image. The relative shift between the two datasets is estimated by finding the peak of FCC in N-dimensions.
Figure 6-1: The schematic of the integer part of the FCC algorithm. The squared \( \omega \)-domain N-D filter is the novel part of this algorithm that allows arbitrarily combining filters in the spatial and frequency domains (the box on the right). The squared operation in the frequency domain is equivalent to applying filters independently to each of the N-D subset datasets.

Even though the PGreg algorithm provides full freedom to use any number of spatial or frequency domain filters on N-D data, only two of the possible options on 2D and 3D images were tested as sample demonstrations. The first demonstration used one 2D Savitzky-Golay filter (one 3D Savitzky-Golay filter for the 3D case), and the second demonstration used two 1D Savitzky-Golay filters (three 1D Savitzky-Golay filter for the 3D case) in all dimensions of the images \((x, y, z)\) for 3D images). The Savitzky-Golay filter was used because of its noise-robustness properties [35].
6.3.2 Estimating subunit shifts

Figure 6-2 shows the schematic of the subunit part of the PGreg method. The subunit algorithm is based on the phase-based subpixel part of the P-SG-GC algorithm [35] (refer to Chapter 2 for the details). Phase-based methods have been used to estimate subpixel shifts in some other methods, such as [44] and [58]. Here, three novel parts were added to the phase-based method. Firstly, similar to the integer part described in Section 6.3.1 and Figure 6-1, a squared frequency domain N-D filter can be applied to the DFT CC. Secondly, a method was proposed to identify and unwrap the phase values that were wrapped in the output of the DFT CC due to noise. In the unwrapping step, the phase error (the difference between the phase value estimated and the phase calculated) was calculated and was used to wrap the phase data to the interval (-π, π] (by subtracting π from, or adding π to, the phase data). This process was continued iteratively until there was no phase data out of the range of (-π, π], or until three iterations were performed (to save the computation time on noisy images). Thirdly, it was proposed to only use the phase values of the unwrapped phase data that their phase errors lie in the interval of [-π/2, π/2] to estimate the phase gradient. This part was added because, after the registration of the data in the integer part of the algorithm, the estimated subunit shifts should be in the interval [-0.5, 0.5] unit. Therefore, the phase data with the phase error values out of the interval of [-π/2, π/2] are affected by noise (absolute errors larger 0.5 unit), and were thus removed from the estimation of the phase gradient. The proposed second and third steps of the subunit part of the PGreg method eliminated the need to use a median filter to remove the spurious phase data, as it was done in [35].

The 3D version of the proposed subunit algorithm was tested on 3D images with various contrast ranges, reported in Section 6.4.3. The frequency domain N-D filter used was set weight of zero for the first three and the last three samples of the DFT CC output at every dimension, and weight of one for the rest of the data (the zero-frequency component was shifted to the centre of spectrum). The reason of using this filter was to remove the noisy phase data observed in few samples close to the beginning and end of the DFT CC phase.
data. The zero frequency element of the filter was also set to zero to remove the DC component of the data.

Figure 6-2: The subunit part of the PGreg method.
6.3.3 Generating synthetic 2D/3D images

Synthetic real valued 2D/3D images were generated using uniformly distributed pseudorandom integers. A 2D/3D Gaussian smoothing filter with a kernel size of 3 pixel²/voxel³ was applied to the generated 2D/3D images. Pseudorandom integers were selected in three ranges of 32, 64, and 128 intensity values centred around 127 (i.e. [111 to 143], [95 to 159], and [63 to 191]) to generate a range of contrasts in the images. Intensity values were selected around 127 (not 0) to avoid over- or under-saturation when Gaussian noise was added to the images to test the robustness of the algorithms to noise. Similar approaches were taken to generate synthetic 3D images in [36, 124, 126].

6.3.4 Applying synthetic shifts

Synthetic subunit shifts were applied to the images using the FFT-shift technique as it was used to test DVC algorithms in [124]. The FFT-shift technique has been shown to closely resemble actual physical shifts [70].

To test the algorithms, 50 uniformly distributed random subunit shifts in the range of $[-w/4$ to $+w/4]$ pixel/voxel ($w$ is the correlation window size) were applied to all dimensions of 50 randomly generated test images at each Gaussian noise level. The maximum displacement of $\sqrt{2} \times (w)/4$ pixel and $\sqrt{3} \times (w)/4$ voxel were thus applied to the test 2D images and 3D images, respectively. The synthetic images were regenerated before applying the random shifts (the intensities of the synthetic images were uniformly distributed pseudorandom integers). The applied shifts were then measured at 20 uniformly distributed points in the images, which resulted in $50 \times 20 = 1000$ measurements at each level of Gaussian noise, at each correlation window size, and for each algorithm.

6.3.5 Adding Gaussian noise and measuring estimation errors

The accuracy of the algorithms in estimating the applied shifts (Section 6.3.4) was tested on samples of generated images and a Landsat image of Paris [116]. Correlation windows of size
16 pixel, 32 pixel, and 64 pixel in each dimension were used in these tests. The robustness of the algorithms to noise was tested by adding Gaussian noise with standard deviations from 0 to 50 to the test images.

To illustrate the advantages of the PGreg method, compared to the other methods, the integer parts of the algorithms were tested separately. A failure rate was defined to assess the integer part of the algorithms. The algorithm was considered to have failed if the absolute estimation error of the integer part ($IE$) in any of the $N$ dimensions of the test image was larger than 0.5 pixel/voxel (Equation 6.2).

\[
\text{failure} = \begin{cases} 
1 & \text{if } |IE(n)| > 0.5 \\
0 & \text{if } |IE(n)| \leq 0.5 
\end{cases} \quad n \in \{1, 2, 3, \ldots, N\} \text{ and } n \leq N 
\]  \tag{6.2}

The shifts were applied to all dimensions of the test images to provide a fair comparison to assess the algorithms. The threshold of 0.5 for failure in Equation 6.2 means that the algorithm could not find the closest integer number to the applied shift in that dimension. The failure rate was then defined as the ratio of the failed estimations to the total number of estimations at each Gaussian noise level ($M$).

\[
\text{failure rate} = \frac{1}{M} \sum_{m=1}^{M} \text{failure}_m 
\]  \tag{6.3}

The error of the integer and subunit parts of the PGreg algorithm in estimating applied shifts at all dimensions (Section 6.3.4) was estimated by measuring the Euclidian distances between the estimated points and the actual shifted points ($D$). The error was averaged for all the measurements performed at each Gaussian noise level (Equation 6.4).

\[
\text{average error} = \frac{1}{M} \sum_{m=1}^{M} D_m 
\]  \tag{6.4}

The mean bias error (or systematic error) of the subunit part of the 3D PGreg algorithm was also calculated. Subunit shifts in the range of [0 to 1] with a step size of 0.1 voxel were applied to the $\zeta$ direction of 50 synthetically generated test 3D images, and the algorithm was used to estimate the subunit shifts at 20 uniformly distributed points in those 3D images. The average
An Accurate and Fast Algorithm for Subunit Registration of Arbitrary Dimensional Data

distance between all the measured shifts \(S_{me}\) and applied shifts \(S_{ap}\) was defined as the mean bias error (Equation 6.5).

\[
\text{mean bias error} = \frac{1}{M} \sum_{m=1}^{M} (S_{me}(m) - S_{ap}(m))
\]  

6.4 Results and discussion

Figure 6-3 shows the Landsat image of Paris used for 2D tests, and Figure 6-4 shows three samples of synthetically generated test 2D/3D images, which were generated using uniformly distributed pseudorandom integers (Section 6.3.3).

Figure 6-3: The Landsat image of Paris [116].
Results and discussion

Figure 6-4: Three samples of synthetically generated test 2D/3D images. The intensity values of the test images were uniformly distributed pseudorandom integers. The intensity ranges were selected as 32, 64, and 128 intensity values centred around 127 to avoid intensity saturations as Gaussian noise was added.

6.4.1 Estimating 2D integer shifts and robustness to Gaussian noise

Uniformly distributed random subunit shifts were applied to the test images (Section 6.3.5) that contained Gaussian noise with standard deviations from 0 to 50. The applied shifts were estimated in the test images using the 2D registration algorithms. Figure 6-5 shows the calculated failure rate of the integer parts of the SGD-GC algorithm [35] and two options for the FCC algorithm in 2D images (i.e., one 2D Savitzky-Golay filter and the product of two 1D Savitzky-Golay filters). The SGD-GC algorithm has been shown to outperform the other state-of-the-art 2D algorithms [35]. The failure rate plots in the left and right columns of Figure 6-5 were calculated for the Landsat image of Paris, and all of the synthetically generated low-contrast images (I1s), respectively (refer to Figure 6-4 for the description of I1, and Section 6.3.5 for the definition of the failure rate). The failure rate plots in the rows of Figure 6-5 were calculated for subimage sizes of (16 × 16) pixel$^2$, (32 × 32) pixel$^2$, and 64 pixel × 64 pixel.
Comparison of plots in the left column with the right column of Figure 6-5 shows that the failure rates had a similar trend in the standard Landsat image of Paris and the synthetically generated I1s. This indicates that the synthetically generated images in this study resemble the characteristics of real standard images. The failure rate plots in Figure 6-5 illustrate that the FCC algorithm with the product of two 1D Savitzky-Golay filters had the highest accuracy in estimating integer shifts compared to the other two algorithms, except for high levels of noise in I1s where the SGD-GC algorithm [35] had a smaller failure rate. The difference between the FCC algorithm with two 1D filters and the other two methods became more obvious at subimage sizes of (32 × 32) pixel$^2$ and (64 × 64) pixel$^2$.

The failure rates of the algorithms had a similar trend in I1s and I2s (refer to Section 6.3.3 and Figure 6-4 for the description of an I1 and an I2). The table and plots in Figure 6-6 summarise the average failure rate of the algorithms at Gaussian noises with standard deviations from 0 to 50 added to the test images (1000 measurements were performed at each Gaussian noise level, as described in 6.3.5). The FCC algorithm with two 1D filters had the smallest average failure rates for all the test images and at all subimage sizes (Figure 6-6). The failure rates of all three algorithms were lower in I2s and I3s because of their higher contrasts. In particular, the average failure rates of the algorithms were very small in I3s at subimage sizes of 32 pixel$^2$ and 64 pixel$^2$, even at high levels of Gaussian noise.
Figure 6-5: The failure rate (Equation 6.2 and Equation 6.3) of the integer parts of the SGD-GC algorithm [35] and two options for the FCC algorithm in 2D images (i.e. one 2D Savitzky-Golay filter and two 1D Savitzky-Golay filters). On the left are the tests results for the standard Landsat image of Paris and on the right are the tests results for the synthetically generated I1s (refer to Figure 6-4 for the description of an I1). The top, middle and bottom rows show the results for subimage sizes of (16 \times 16) \text{ pixel}^2, (32 \times 32) \text{ pixel}^2, and (64 \times 64) \text{ pixel}^2, respectively.
6.4.2 Estimating 3D integer shifts and robustness to Gaussian noise

Uniformly distributed random subvoxel shifts were applied to the test 3D images (Section 6.3.5) that contained intensity Gaussian noise with standard deviations from 0 to 50. The applied shifts were estimated in the test 3D images using 3D-CC and two options for the FCC algorithm in 3D images (i.e. one 3D Savitzky-Golay filter and three 1D Savitzky-Golay filters). The failure rate of the integer part of the FCC algorithm was calculated and compared to the 3D-CC, which is the traditional method of estimating integer shifts in DVC algorithms [19, 21, 124, 133]. Figure 6-7 shows the failure rates of 3D-CC, FCC with one 3D Savitzky-Golay filter, and FCC with three 1D Savitzky-Golay filters, calculated at (16 × 16 × 16) voxel$^3$, (32 × 32 × 32) voxel$^3$, and (64 × 64 × 64) voxel$^3$ subimages of synthetically generated V1s. While the failure rate of 3D-CC was close to 1 in all the subimage sizes, the average failure rate of FCC was less than 0.5 in the majority of cases. Similar to the 2D tests, FCC with three 1D Savitzky-Golay filters had the best performance. Figure 6-8 summarises the average failure
rate of the algorithms at Gaussian noise levels from 0 to 50 standard deviations added to $(16 \times 16 \times 16)$ voxel$^3$, $(32 \times 32 \times 32)$ voxel$^3$, and $(64 \times 64 \times 64)$ voxel$^3$ subimages of the synthetically generated 3D images (1000 measurements were performed at each Gaussian noise level, as described in Section 6.3.4). The average failure rates of FCC with three 1D filters were smaller than 0.002 for all of the subimage sizes of V1s, V2s, and V3s, except at $(16 \times 16 \times 16)$ voxel$^3$ subimages of V1s, which was 0.0413 (Figure 6-8). While the average failure rate of FCC was significantly less than 3D-CC in all cases, it has only a few more computations compared to 3D-CC (i.e. an extra frequency domain multiplication to a filter).

The failure rate of 3D-CC was close to 1 in almost all the cases. The high failure rate of 3D-CC is due to three main limitations: firstly, the inability of CC to estimate large shifts; secondly, the inability of CC to estimate shifts in low-contrast data; and thirdly, the sensitivity of CC to noise. The independent effects of the first two limitations were investigated in another test where synthetic shifts from 0 to $(\text{subimage size})/4$ voxel at a step size of 0.1 voxel were applied to only one dimension of noise-free V1s, V2s, and V3s. 3D-CC was then used to estimate the applied shifts, and the failure rates were calculated. Figure 6-9 shows the failure rate of 3D-CC at $(16 \times 16 \times 16)$ voxel$^3$, $(32 \times 32 \times 32)$ voxel$^3$, and $(64 \times 64 \times 64)$ voxel$^3$ subimages of synthetically generated V1s, V2s, and V3s. The plots in Figure 6-9 illustrates that 3D-CC was unable to estimate large shifts and shifts in low-contrast 3D images. For instance, even at noise-free $(64 \times 64 \times 64)$ voxel$^3$ subimages, 3D-CC was unable to estimate shifts larger than 2 voxel in low-contrast 3D images (V1s), and shifts larger than 7.5 voxel in high-contrast 3D images (V3s) (Figure 6-9).
An Accurate and Fast Algorithm for Subunit Registration of Arbitrary Dimensional Data

Figure 6-7: The failure rates of three algorithms calculated at (16 × 16 × 16) voxel³, (32 × 32 × 32) voxel³, and (64 × 64 × 64) voxel³ subimages of synthetically generated V1s (refer to Section 6.3.3 and Figure 6-4 for the description of a V1).
Results and discussion

Figure 6-8: A summary of the average failure rates of the algorithms at Gaussian noises with standard deviations from 0 to 50 added to subimages of synthetically generated V1s, V2s, and V3s.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>V1 (16 x 16 x 16 voxels)</th>
<th>V2 (32 x 32 x 32 voxels)</th>
<th>V3 (64 x 64 x 64 voxels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D cross-correlation</td>
<td>0.9914</td>
<td>0.9984</td>
<td>0.9992</td>
</tr>
<tr>
<td>FCC (one 3D filter)</td>
<td>0.3477</td>
<td>0.0563</td>
<td>8.5E-03</td>
</tr>
<tr>
<td>FCC (three 3D filters)</td>
<td>0.0413</td>
<td>0.0016</td>
<td>1.5E-05</td>
</tr>
</tbody>
</table>

Figure 6-9: The failure rates of 3D-CC calculated in noise-free V1s, V2s, and V3s.
6.4.3 Estimating 3D subvoxel shifts and robustness to Gaussian noise

The accuracy of the 3D version of the PGreg algorithm in estimating subvoxel shifts was tested at (16 × 16 × 16) voxel\(^3\), (32 × 32 × 32) voxel\(^3\), and (64 × 64 × 64) voxel\(^3\) subimages of synthetically generated 3D images. Uniformly distributed random shifts were applied to test 3D images (Section 6.3.5) that contained intensity Gaussian noise with standard deviations from 0 to 50. Filtered cross-correlation with three 1D filters was used for the integer part of the algorithm, and the 3D version of the phase-based method described in Section 6.3.2 was used for the subvoxel part. Figure 6-10 shows the average error of the PGreg algorithm with three 1D filters in estimating random shifts at (16 × 16 × 16) voxel\(^3\), (32 × 32 × 32) voxel\(^3\), and (64 × 64 × 64) voxel\(^3\) subimages of synthetically generated V1s, V2s, and V3s. As Figure 6-10 illustrates, the accuracy of the PGreg algorithm was greater when using subimage of size (32 × 32 × 32) voxel\(^3\), and (64 × 64 × 64) voxel\(^3\) compared to (16 × 16 × 16) voxel\(^3\) subimages. The lowest average estimation errors were achieved at subimages of V3s, which had the highest contrast. Even though the errors increased when the standard deviation of Gaussian noise was increased, errors were less than 0.2 voxel, even at high Gaussian noise levels, in all cases except (16 × 16 × 16) voxel\(^3\) subimages of V1s at Gaussian noise with a standard deviation higher than 25. Considering that the intensity ranges of V1s, V2s, and V3s, were respectively 32, 64, and 128 intensity units, the average errors in Figure 6-10 indicate the robustness of the PGreg method to Gaussian noise. Figure 6-10 also shows that the average errors were always less than 1 voxel, except at (16 × 16 × 16) voxel\(^3\) subimages of V1s at Gaussian noise with standard deviations higher than 40. This indicates that the integer part of the algorithm was very robust to Gaussian noise, and had only a few
failures at \((16 \times 16 \times 16)\) voxel\(^3\) subimages of V1s at Gaussian noise with standard deviations higher than 40 (this is in accordance with the results in Section 6.4.2).
Figure 6-10: The average error of the PGreg algorithm with three 1D filters in estimating random shifts.

The average and the standard deviation of the estimation errors calculated for the PGreg algorithm at \((16 \times 16 \times 16)\) voxel\(^3\), \((32 \times 32 \times 32)\) voxel\(^3\), and \((64 \times 64 \times 64)\) voxel\(^3\) noise-free subimages of V1s, V2s, and V3s (Figure 6-10) were \(0.0112 \pm 0.0033\) voxel, \(0.00311 \pm 0.0011\) voxel, and \(0.0015 \pm 0.00040\) voxel, respectively.

The mean bias error of the subvoxel part of the PGreg algorithm was tested at \((17 \times 17 \times 17)\) voxel\(^3\), \((25 \times 25 \times 25)\) voxel\(^3\), and \((33 \times 33 \times 33)\) voxel\(^3\) subimages of noise-free V3s. The subimage sizes selected were the same as those used by Wang et al. [124] to enable performance comparisons. Figure 6-11 shows the mean and standard deviation of bias errors for the PGreg algorithm. The maximum absolute error of the PGreg algorithm was less than \(0.00011\) voxel at \((17 \times 17 \times 17)\) voxel\(^3\) subimages, which is significantly smaller than the \(0.003\) voxel achieved in the method of Wang et al. [124]. The difference between the PGreg algorithm and the method of Wang et al. [124] was more significant at larger subimages. While the maximum absolute error of the method of Wang et al. [124] remained around \(0.003\) voxel, the maximum absolute error of the PGreg algorithm at \((25 \times 25 \times 25)\) voxel\(^3\), and \((33 \times 33 \times 33)\) voxel\(^3\) was \(1.2 \times 10^{-5}\) and \(3.6 \times 10^{-6}\), respectively. This illustrates that the error of the proposed phase-based method could be decreased by increasing the correlation window size. This is an important advantage over area-based interpolation methods of finding subvoxel shifts, in which the error does not reduce as correlation window size is increased [36, 124].

Another main difference between the proposed phase-based method of finding subvoxel shifts and area-based interpolation methods is in the shape of their mean bias error plot at subvoxel shifts from 0 to 1 voxel. A sinusoidal shape was observed for the error plots in area-based interpolation methods [36, 124]. The reason was not explained in [36, 124], but the sinusoidal shape of the error plots is due to an effect named pixel-locking (or peak-locking) [57, 134]. Estimated positions in area-based interpolation methods tend to be biased toward
integer values because of the pixel-locking effect [57, 134]. Thus, subvoxel errors of the area-based interpolation methods of [36, 124] had a large variation with positive values before, and negative values after, 0.5 voxel [36, 124]. The proposed phase-based method of finding subvoxels was not influenced by voxel-locking effects since the subvoxel shifts were found from the phase information of the 3D images in the Fourier domain. The maximum error of phase-based methods happens at subvoxel shifts of around 0.5 voxel (Figure 6-11). Figure 6-11 illustrates that the mean bias errors of the proposed phase-based method were small negative numbers for all the measured subimages, which indicates that the subvoxel part of the PGreg algorithm tends to underestimate the shifts by a small amount.

![Figure 6-11: The mean and standard deviation of bias errors of the PGreg algorithm in estimating subvoxel shifts at subimages of V3s.](image)

The independent effects of the addition of the phase unwrapping and the selection of the phase data with phase error values between the range of $[-\pi/2, \pi/2]$ were also investigated at $(16 \times 16 \times 16)$ voxel$^3$ subimages of V1s, which had the highest average error (Figure 6-10). Uniformly distributed random shifts were applied to subimages of V1s (Section 6.3.5) that contained Gaussian noise with standard deviations from 0 to 50. The phase unwrapping step, and the step involving the selection of the phase data with phase error values within the range $[-\pi/2, \pi/2]$ were excluded from the subvoxel part of the PGreg algorithm, and the average errors were measured. Figure 6-12 shows the average error for the subvoxel part of the
algorithm with and without these two steps. As Figure 6-12 illustrates, the addition of these two steps was effective in decreasing the error in this test.

![Graph showing 3D measurement error in (16 x 16 x 16) voxel^3 subimages of V1s](image)

Figure 6-12: The effects of the addition of the phase unwrapping and the selection of the phase values with phase error within the range of \([-\pi/2, \pi/2]\) in the subvoxel part of the PGreg algorithm tested at (16 x 16 x 16) voxel^3 subimages of V1s.

6.5 Summary

An algorithm was proposed for subunit registration of multidimensional data to address many of the limitations of the existing methods. The PGreg algorithm was proposed to find integer and subunit shifts. The proposed FCC algorithm for the integer part of the PGreg algorithm has significant flexibility, enabling it to be tailored to a specific application or to a specific N-D registration problem.
Demonstrations of two options from 2D and 3D versions of the PGreg algorithm were tested in this study. The first demonstration included one 2D/3D Savitzky-Golay filter, and the second demonstration included two/three 1D Savitzky-Golay filters. The 2D tests illustrated that FCC with two 1D Savitzky-Golay filters could achieve lower failure rates compared to the SGD-GC algorithm [35] (Figure 6-5, Figure 6-6). It should be noted that the SGD-GC algorithm has previously been shown to perform better than the other state-of-the-art 2D image registration algorithms [35]. The 3D tests indicated that FCC with either three 1D Savitzky-Golay filters or one 3D Savitzky-Golay filter had significantly lower failure rates compared to 3D-CC (Figure 6-8). While 3D-CC failed to estimate large shifts or shifts in low-contrast 3D images (Figure 6-9), the average failure rates of FCC with one 3D filter and three 1D filters in estimating large shifts at (16 x 16 x 16) voxel^3, (32 x 32 x 32) voxel^3, and (64 x 64 x 64) voxel^3 subimages of V1s, V2s, and V3s were 0.046 and 0.004, respectively (Figure 6-8).

In both the 2D and 3D cases, when high levels of Gaussian noise were present, the FCC algorithm with two/three 1D filters performed better than FCC with one 2D/3D filter. This could have been because the nonzero values of 2D/3D filter are more than that of a 1D filter. Therefore, a larger number of non-local noisy values will be included in the shift estimations, which increases the uncertainty and thus the error of measurements. Furthermore, isolating the shift estimations to each dimension of the data can increase the efficiency of shift estimations. A similar approach was taken by Tong et al. [135] to find subpixel shifts using singular value decomposition (SVD). However, the PGreg algorithm is considerably less computationally complex than the SVD method of Tong et al. [135].

The PGreg algorithm was tested at images subjected to Gaussian noise with standard deviations from 0 to 50. It was shown that the PGreg algorithm is capable of finding large shifts, can handle low-contrast images, and is robust to Gaussian noise. The average errors of the PGreg algorithm were less than 0.2 voxel for the majority of the test cases degraded by Gaussian noise (Figure 6-10). The average errors for noise-free (16 x 16 x 16) voxel^3,
An Accurate and Fast Algorithm for Subunit Registration of Arbitrary Dimensional Data

(32 × 32 × 32) voxel$^3$, and (64 × 64 × 64) voxel$^3$ test subimages were 0.015 voxel, 0.00065 voxel, 0.00011 voxel, respectively (Figure 6-10).

The mean bias error of the subvoxel phase-based part of the PGreg algorithm was also calculated and was compared to the method of Wang et al. [124]. The PGreg algorithm could achieve significantly smaller mean bias errors compared to the method of Wang et al. [124] (i.e. 27 times smaller at (17 × 17 × 17) voxel$^3$ subimages, 250 times smaller at (25 × 25 × 25) voxel$^3$ subimages, and 830 times smaller at (33 × 33 × 33) voxel$^3$ subimages). Moreover, it was shown that, unlike area-based interpolation methods [36, 124], the PGreg algorithm does not suffer from voxel-locking effects (Figure 6-11).

The PGreg algorithm is not computationally complex or demanding, because: firstly, the filters of the FCC method can be precomputed; secondly, applying filters in the Fourier domain instead of spatial domain as required in the SGD-GC algorithm [35]; and thirdly, the FCC algorithm uses real-valued FFTs, whereas the SGD-GC algorithm uses complex-value FFT. Real-value FFTs can be computed more efficiently than complex-value FFTs. For instance, the computation costs of two real-value FFTs are only slightly more than one complex-value FFT [136]. Moreover, the 3D version of the PGreg algorithm performs well in finding large shifts, and is significantly less computationally demanding than the iterative method of Bar-Kochba et al. [126].

The FCC algorithm in the integer step of the PGreg algorithm is a flexible algorithm in which the filters can be chosen based on the specifications of the N-D data. However, only one type of filter (i.e. the Savitzky-Golay filter) was investigated in the current study. Further research is needed to fully investigate the flexibility of the FCC algorithm, and the new capabilities enabled by this method.
7 A Low-cost, Hand-held Stereoscopic Device for Measuring Deformations of Skin In Vivo

The content of this chapter is based largely on the following conference paper:


Abstract

Measuring the deformation of skin in vivo is useful in a number of applications. For example, the response of skin to a variety of mechanical loadings can provide information about the health of the underlying tissue. A number of devices have been developed for measuring the surface deformation of in vivo skin. However, existing devices are typically incapable of covering large areas of skin, or are expensive. To address these issues, we present the design and evaluation of a hand-held low-cost stereoscopic device for in vivo measurement of the dynamic surface deformation of skin. A camera rig with sufficient mechanical strength was designed to hold four high-speed synchronised cameras to cover a field of view (FOV) of approximately 200 mm × 200 mm. The camera bus interface to the PC and the camera settings
were selected based on the required specifications of measuring dynamic surface deformations. A code was developed to enable the four cameras to simultaneously capture 150 frame/s. The FOV of the designed camera rig was tested to ensure it was matched with the FOV estimated for the SolidWorks design using camera and lens specifications. The total mass of the device was approximately 3 kg, and the hardware cost was less than 3000 USD for four cameras and their lenses.

6.2 Introduction

Tracking the surface deformations of soft tissues (e.g. skin, and breast) can be used to help identify tissue mechanical properties. The biomechanical properties of skin have been studied extensively for various applications [17][98], and computational modelling of skin was used to assist in studying wrinkling [137], simulating stretching and compression of skin for animations [138], simulating plastic surgery [96], and in the design of high-performance clothing [139]. Pioneering studies that have characterised the mechanical properties of in vitro skin were published in the 1970s. In 1973, Lanir and Fung analysed the mechanical behaviour of rabbit skin during biaxial tension tests [140], and tracked markers in a single camera image [141].

Skin is anisotropic and heterogeneous [30], and typically exhibits different characteristics between the in vivo and in vitro states. If one is interested in the behaviour of living skin, it is necessary to identify and measure detailed localised surface deformations in vivo. Even though biaxial tests are a common way of studying the mechanical properties of skin, such experiments are difficult to perform on human skin in vivo [12]. A number of alternative techniques, such as tissue compression [13], suction [14], and indentation [15], have been used to estimate the mechanical properties of skin in vivo. Measurement devices such as a micro-robot [100], a motion capture system [101], a single camera system [103], and a polarised LED sphere with macro camera [138], have been developed to measure in vivo skin surface deformations. However, motion capture systems (such as [101]) are unable to provide accurate measurements, single camera systems [138][103] can only cover flat, limited areas of skin, and the micro-robot system [100] is complicated and expensive to build, and only can be used for small areas of skin. A comprehensive analysis of stretching and compression of large regions
of skin would be possible if the dynamic deformations could be simultaneously captured from different viewing angles. To address this, we designed a low-cost, hand-held stereoscopic device to track dynamic deformations of skin in vivo using four different viewing angles. This portable device can be easily held in one hand to measure localised surface deformations over various parts of the body, such as the hand or face. Design considerations included the provision of a suitable field of view (FOV) for in vivo measurement of skin deformations, adequate depth of field (DOF) for good quality images, sufficient mechanical strength and rigidity of the camera rig, and symmetric views for overlapping FOVs. This novel device was tested by capturing synchronised images of a hand from four different angles at 150 frames per second (fps).

6.3 Camera and lens selection

The first step in designing a stereoscopic system is to select a suitable camera. Machine vision cameras are commonly used for measurement purposes. In general, the selection of a camera involves a trade-off between cost and other specifications, such as image resolution, maximum frame-rate, physical size, and image quality. High-frame-rate cameras have the capability to record intermediate states of the skin during deformation. Therefore, they are preferable for in vivo measurement, where skin can often undergo complex and rapid deformations. Nonetheless, high-frame-rate cameras are typically expensive, and thus unsuitable for low-cost systems. For this stereoscopic device, the aim was to have a reasonable trade-off between frame-rate and cost. The frame-rate (i.e. data throughput) and cost are primarily determined by the bus interface from the camera to the computer. Various camera bus interfaces are introduced in the following section, and used to determine a suitable selection for this specific application.

6.3.1 Selecting the camera bus interface

The main camera bus interfaces for PC-based systems currently include [89]: 
• FireWire (IEEE 1394);
• Gigabit Ethernet (GigE);
• Camera Link;
• Universal Serial Bus (USB 2 and USB 3).

FireWire is a serial bus interface for high-speed communications (FireWire is the name chosen by Apple Inc. for IEEE 1394 interface standard). The IEEE 1394a provides 40 MB/s data transfer rate [89], which cannot handle frame-rates higher than 32 fps for 1024 pixel × 1280 pixel resolution images (8-bit pixel intensity values). Since the establishment of the USB interface as the standard port in PCs, the FireWire interface is no longer generally available in PCs. Therefore, FireWire cameras are not considered to be a suitable choice for this system. Even though Apple Inc. introduced Thunderbolt [142] in 2011 to replace FireWire, machine vision cameras that incorporate the Thunderbolt interface have only been introduced very recently [143], and are not yet available in the market.

GigE uses a standard Ethernet port for data communication. The maximum theoretical bandwidth of GigE is 125 MB/s [89], which is insufficient for frame-rates higher than 100 fps for 1024 pixel × 1280 pixel resolution images (8-bit pixel intensity values), hence unsuitable for this system.

The Camera Link interface was the first standard developed for high-speed data transfer for digital cameras in industrial applications. The Camera Link interface has three bandwidth configurations designed for different applications: the base, medium, and full bandwidth configurations. The data transfer rate for these Camera Link configurations are summarised in Table 7-1.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Data rate (Gbit/s)</th>
<th>Data rate (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>2.04</td>
<td>255</td>
</tr>
<tr>
<td>Medium</td>
<td>4.08</td>
<td>510</td>
</tr>
<tr>
<td>Full</td>
<td>5.44</td>
<td>680</td>
</tr>
</tbody>
</table>

Table 7-1: Camera link bandwidth configurations
Although the Camera Link base configuration specifies a relatively high data transfer rate (i.e. 255 MB/s), the majority of cameras achieve only about 100 MB/s, which is not suitable for frame-rates higher than 100 fps for 1024 pixel × 1280 pixel resolution images. However, the Camera Link medium and full bandwidth configurations provide a sufficient data transfer rate for high frame-rate applications. Some of the advantages of the Camera Link interface are:

- Transmission of image data is independent of the CPU and the operating system; hence cameras can be hardware triggered with a very short latency;
- Synchronisation between Camera Link cameras is straightforward;
- The camera settings can be controlled using Camera Link drivers;
- Frame grabbers manage image data transmission to local memory via direct memory access (DMA), which does not overload the CPU;
- Frame grabbers have processing units that are able to perform basic processing on the image data at high-speed.

Limitations of the Camera Link interface include:

- Camera Link is not a standard PC interface, hence frame grabber cards are needed for transferring the image data from the camera to the host PC;
- Camera Link cameras, frame grabbers, cables and accessories are typically expensive;
- New hardware technologies rapidly outdate Camera Link frame grabbers.

Even though the medium and full configurations of the Camera Link interface may be suitable options for this stereoscopic system, these options were not chosen due to the above limitations.

The USB 3 vision standard (also known as SuperSpeed USB) was published in January 2013 for use in camera systems. USB 3 vision has a number of improvements over USB 2, such as
the cable locking option, higher bandwidth, lower CPU demand, increased power delivery, and reduced power consumption [144]. The theoretical data transfer rate using USB 3 vision is 5 Gbit/s (625 MB/s). The USB 3 vision interface is rapidly growing in popularity, and has become the primary choice for cost-effective solutions. Some of the advantages of the USB 3 interface include:

- The USB 3 interface is now a standard port, hence frame grabber cards are not needed for USB 3 cameras;
- USB 3 cameras, cables and accessories are typically low-cost;
- The USB 3 interface can provide power over the data cable, which eliminates the need for a separate power source;
- USB 3 cameras can transfer image data directly to the host memory via DMA with no overload on the CPU.

Nevertheless, USB 3 also has limitations:

- Although the theoretical data transfer rate using USB 3 is 625 MB/s, it is in practice limited to 400 MB/s (portions of the USB 3 bandwidth are dedicated to symbol encoding, link level flow control, and protocol overhead);
- USB 3 vision does not support frame grabbers; thus synchronisation of USB 3 cameras is more challenging compared with Camera Link cameras.

Table 7-2 shows a comparison between the key technical features of the Camera Link and USB 3 vision interfaces. Even though the data transfer rate of USB 3 is less than the Camera Link interfaces, the difference is not critical for measuring dynamic surface deformations. In addition, the maximum cable length of USB 3 (5 m) is sufficient for PC-based applications. Therefore, considering the lower price of USB 3 cameras and accessories compared with the Camera Link interface, USB 3 was chosen as the camera interface of this stereoscopic device.
A Low-cost, Hand-held Stereoscopic Device for Measuring Deformations of Skin In Vivo

Table 7-2: Camera link and USB 3 bandwidth configurations and maximum cable length

<table>
<thead>
<tr>
<th>Camera interface</th>
<th>Data transfer rate (MB/s)</th>
<th>Maximum cable length (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera Link</td>
<td>510 or 680</td>
<td>10</td>
</tr>
<tr>
<td>USB 3</td>
<td>400</td>
<td>5</td>
</tr>
</tbody>
</table>

6.3.2 Selecting the camera and lens

Monochrome USB 3 cameras from Point Grey (FL3-U3-13Y3M-C) were selected for this device [145]. The maximum frame-rate of these cameras is 150 fps, and the maximum image resolution is 1280 pixel × 1024 pixel. Therefore, the maximum data transfer of these cameras is 196.6 MB/s, which is approximately half of the maximum bandwidth of the USB 3 vision interface (Table 7-2). The single unit price of FL3-U3-13Y3M-C cameras is 595 USD [146]. In comparison, the single unit price for a Camera Link camera with 150 fps from Point Grey (GZL-CL-41C6M-C) is 2195 USD [147]; moreover, a frame grabber card, which typically costs around 700 USD [148], is also needed for a Camera Link camera.

The next step was to select the camera lens. Short focal length lenses provide a large angle of view (AOV) (i.e. a large FOV at a constant distance to the camera); thus such lenses can allow the physical dimension of the camera setup to be more compact for a particular FOV. Nevertheless, short focal length lenses usually have a high level of barrel distortion, which may cause inaccuracies in deformation measurements. To strike a balance between having a reasonable physical dimension for the camera setup, and an acceptable range of barrel distortion, 6 mm focal length lenses were chosen for this stereoscopic system (DF6HA-1B from Fujifilm). This lens provides 44.54° AOV for FL3-U3-13Y3M-C cameras.

6.4 Camera rig design

The mechanical design of this hand-held stereoscopic device considered the following criteria:
• The FOV of approximately 200 mm × 200 mm was required to cover an adequate surface area of the skin from different viewing angles;
• Large enough overlapping FOVs were required for measuring skin deformations in 3D. Each part of the skin must be imaged with at least two cameras to reconstruct the deformations in 3D;
• The angle between different camera axes at the centre of the FOV needed to be close to 90°. Orienting the cameras in this way results in isotropic stereoscopic resolution;
• The physical dimensions and mass of the stereoscopic system needed to be small enough to be portable and hand-held;
• The rig required enough mechanical strength to hold the cameras rigidly during hand-held use;
• The rig needed to be constructed using cost-effective materials and processes.

A ring arrangement for the cameras was a suitable configuration for tracking the skin deformations from different viewing angles. Four cameras were arranged as two opposite pairs to cover the object. Increasing the number of cameras can improve the 3D reconstruction process, but it would also make the device more expensive and heavier. Based on the design specifications, the positions of the four cameras were calculated, and a camera rig was developed.

6.4.1 Determining the cameras positions

The focal length of the camera lens is equivalent to the distance between the back principal point of the lens and the camera image plane. The back principal point is thus the physical location of the focal point of the camera that should be considered in the calculation of the FOV. The positions of the cameras in each opposing pair were calculated by considering the target FOV of 200 mm × 200 mm, the required 90° angle between the axes of the cameras at the centre of the FOV, and the AOV of the lens (Figure 7-1). In the design in Figure 7-1, the distance between the opposing cameras is 276 mm, and the height of the principal points is 138 mm. Note that the FOV is trapezoidal for a rectangular camera image plane.
6.4.2 Design of the camera rig

The rig needed to accommodate the four cameras in the positions calculated in Section 6.4.1. The geometric specification of the camera rig was chosen based on the conditions discussed in Section 6.4. After several iterations, the design illustrated in Figure 7-2 was selected for this device. In this design, all of the components were separate 2D parts, which were assembled to build the 3D rig. As Figure 7-2 illustrates, the cameras were arranged in a ring configuration with two pairs of opposing cameras. Four LEDs were used to illuminate the surface of the skin. The LEDs were positioned on a circular rail, enabling the height and position of LEDs to be adjusted to manage the direction of the light for objects of differing shapes. Each camera was connected to the central plate of the device using two curved plates to ensure sufficient mechanical strength. The curved plates were attached and fixed to the central plate in an inter-locking manner as shown in Figure 7-3. The connections were glued to ensure that the camera rig was sufficiently stable. It was also possible to screw the curved...
plates to the central plate (Figure 7-3). The cameras were fixed to the curved plates with attachments and screws as shown in Figure 7-4.

Figure 7-2: The design of the stereoscopic device. Four cameras were placed in a ring configuration in two opposite pairs. The heights and the distances between the cameras were chosen based on the calculations in Section 6.4.1.

Figure 7-3: The top view (a) and the bottom view (b) of the central plate of the design. The curved plates were fixed and glued to this central plate in an inter-locking manner. It was also possible to screw the curved plates to the central plate.
A Low-cost, Hand-held Stereoscopic Device for Measuring Deformations of Skin In Vivo

Figure 7-4: The side view (a) and bottom view (b) of the camera attachments to the curved plates of the camera rig.

The camera FOV was trapezoidal in shape. Figure 7-5 shows the FOV for each of the four cameras of the stereoscopic device. The overlapping FOV can be seen in Figure 7-5(b).

After finalising the design, a laser cutter was used to cut sheets of acrylic to build the 2D components of the camera rig. The acrylic sheets were 6 mm thick to provide a reasonable trade-off between mechanical strength and mass for the camera rig. Figure 7-6 shows the final hand-held stereoscopic device.

To compare the theoretical calculation of the FOV with the measured FOV, tests were carried out with this hand-held stereoscopic device, and the results confirmed the validity of our theoretical calculations for each camera.
Figure 7-5: The side view (a) and bottom view (b) of the fields of view of each of the four cameras. The overlapping FOVs of the cameras are approximately inside the polygon shown in the image (b).

Figure 7-6: The designed hand-held stereoscopic device for in vivo measurement of skin surface deformations.

6.4.3 Mass

The mass of the portable part of the stereoscopic device was approximately 2 kg. By including the approximate mass of four cameras (each 52 g), four lenses (each 55 g), USB 3
cables (each 150 g), and LED heat sinks (each 60 g), the total mass of the portable device was approximately 3 kg.

6.5 Camera settings

6.5.1 Depth of field (DOF)

DOF is the distance between the nearest and farthest objects that have a sharp focus (i.e. are not blurred). Equation 7.1 shows how the DOF of a lens can be approximated [89]:

\[
\text{DOF} \approx \frac{2 \times C \times F_n \times f_d \times (f_d - f)}{f^2}
\]

where \( C \) is the acceptable diameter of the circle of confusion (i.e. the smallest part of the image which is acceptably sharp), \( F_n \) is the \( f \)-number of the lens (i.e. the ratio of the focal length to the diameter of the pupil of the lens), \( f_d \) is the distance for which the lens is focused, and \( f \) is the focal length of the lens. For predefined FOV dimensions with a fixed focal length lens (such as DF6HA-1B), the \( f \)-number is the main factor that determines the DOF. Larger \( f \)-numbers result in sharper camera images. However, since pupil sizes are smaller for larger \( f \)-numbers, more light would be needed to illuminate the object. Based on the design of this stereoscopic device, the maximum DOF in the overlapping FOV was less than 240 mm. The focusing distances of the lenses were set at the centre of the overlapping FOV (174 mm from the lenses), and the permissible circle of confusion for the cameras was 15 \( \mu \)m based on the camera data sheet. Substituting these values into Equation 7.1, provided the acceptable \( f \)-number for the cameras in the stereoscopic device as \( F_n = 9.85 \). Therefore, the \( f \)-number was chosen to be 10 for the lenses of the cameras.

6.5.2 Illumination

Cameras needed adequate light for the \( f \)-number of 10 at 150 fps. The camera specification in the datasheet shows that the camera sensor has a high sensitivity to the wavelengths close
to the green colour. Thus, four high-power bright green LEDs (750 lm) were chosen for the hand-held device. The LEDs could be changed to other colours depending on the optical characteristics of the tissue. Circular polarising filters were used for each camera to reduce specular reflections, which may result from the use of non-diffused LED point source lights.

6.5.3 Capturing image data and synchronising the cameras

Equation 7.2 and Equation 7.3 show the required data transfer rates when using one and four USB 3 cameras (FL3-U3-13Y3M-C) at 150 fps, respectively.

Data transfer rate (one camera)

\[
\text{Data transfer rate (one camera)} = \frac{1}{8} \times 150 \text{ fps} \times 1280 \text{ pixel} \times 1024 \text{ pixel} \times 8 \text{ bit} \\
\approx 196.6 \text{ MB/s}
\]

Data transfer rate (four cameras) = 196.6 MB/s \times 4 = 786.4 MB/s

Tests showed that the storage rates for a SATA hard drive and for a solid-state drive (SSD) were approximately 42.5 MB/s, and 177.5 MB/s, respectively. Therefore, these hard drive technologies were not suitable for storage of the image data from the four cameras. Instead, the image data were first buffered in DDR3 RAM of the PC during recordings, and subsequently transferred to the hard drive. The storage rate of DDR3 RAM is approximately 1 GB/s, which was adequate to simultaneously capture the image data from all four cameras. This approach limits the recording time to the amount of available DDR3 RAM. For instance, the recording time for a PC with 32 GB DDR3 RAM was limited to less than 40 seconds. For longer recording times, double buffering between DDR3 RAM and the hard drive can be used. In this study, the 40 seconds recording duration was sufficient for measuring dynamical skin deformations at high frame-rates.

The cameras were simultaneously triggered in software via separate CPU threads to capture synchronised data. A digital millisecond timer was used to verify the synchronisation of the cameras for the duration of the recordings (the maximum recording duration was 30 seconds for these tests).
6.6 Field of view of the hand-held stereoscopic device

Figure 7-7 shows an example of the images of a hand from each of the four cameras in the designed hand-held stereoscopic device. The pixel size of this device was approximately 175 µm for this FOV (this camera rig was used for the experiments in Chapter 8 and Chapter 9). As shown in Figure 7-7, the cameras were able to span the entire hand from different viewing angles. Figure 7-8 shows an example of measuring 2D displacement of the skin on the hand for two sequential images captured at 150 fps. The displacements were measured by tracking the intrinsic features of the skin using the subpixel image registration method described in Chapter 2 and tested in Chapter 4. The displacement vectors are superimposed on the image, and the magnitudes of the displacement values were indicated using a colour spectrum. Figure 7-8 demonstrated that the illumination and the f-number of the cameras were appropriately selected to provide good quality images with sufficient sharpness to perform deformation measurements.
Figure 7-7: Images of a hand from each of the four cameras of the designed hand-held stereoscopic device. The camera images spanned the whole surface of the hand from each of the four viewing angles.

Figure 7-8: A sample 2D displacement measurement from the surface of the skin using two sequential images of a hand from one of the cameras of the hand-held device.
6.7 Conclusions

This chapter describes the design of a hand-held stereoscopic device for measuring skin deformation in vivo from different viewing angles. The results showed that the cameras can capture synchronised images from a FOV of approximately $200 \text{ mm} \times 200 \text{ mm}$. The camera settings and lighting were chosen appropriately to provide good quality images in the FOV with approximately 240 mm DOF. The total mass of the designed device was approximately 3 kg, and the hardware cost was less than 3000 USD for four cameras and their lenses. The cameras of this device had sufficient overlapping FOV (Figure 7-5 and Figure 7-7) to enable extending the measurements to 3D, and had sufficient image sharpness to measure skin deformation in vivo (Figure 7-8). This device is thus a suitable cost-effective hand-held setup for measuring 3D skin deformation for various parts of the body.
8 Robust and Accurate Multiple Cameras Stereographic Calibration

The content of this chapter is largely based on the following journal paper, which is in review for the International Journal of Computer Vision:


8.1 Abstract

The simultaneous calibration of multiple cameras is an important task in three-dimensional (3D) computer vision systems. The accuracy of stereoscopic systems in performing 3D measurements is often dependent upon the accuracy of camera calibration. However, the camera calibration process is challenging, particularly when using more than two cameras. Most of the existing multi-camera calibration methods have limitations that prevent them from achieving accurate, robust, and fast calibration even for two cameras. For example, the existing methods require good initial estimates of camera parameters, and can only handle a small number of calibration images. Most of these limitations arise from the inefficiency of the optimisation process used to estimate and refine the unknown parameters of the cameras and lenses. Here, these limitations were addressed by proposing a novel procedure to estimate the parameters of multiple camera stereo systems. The trust-region-reflective optimisation algorithm was used with an error matrix instead of the Levenberg-Marquardt (LM) algorithm
with a single error value. Two 3D error functions were introduced and a method was proposed to effectively integrate them into the objective function of the optimisation process. This method has a fast convergence rate, and can estimate the parameters accurately for an arbitrary number of cameras, even with inaccurate initial estimates. Furthermore, this optimisation process is able to simultaneously calibrate all of the cameras of a stereo system using many calibration images, which improves the accuracy of camera parameter estimations compared to the methods that only can calibrate stereo-pairs or are limited to the use of a small number of images.

The advantages of the method are demonstrated and compared to traditional methods in several tests performed using a typical checkerboard calibration template. Results demonstrate that the method is able to estimate the parameters accurately even when the traditional methods fail, and can outperform traditional methods in accuracy, convergence speed, and robustness. Even though a checkerboard template was used for the tests, the method is able to use any camera calibration target for identifying the parameters.

### 8.2 Introduction

Three-dimensional (3D) computer vision systems have many applications in robotics [149], shape reconstruction [150], quality control [78, 151], and 3D measurements in experimental mechanics [152, 153]. The majority of 3D computer vision systems use stereoscopic cameras to capture images from various viewing angles to generate 3D models of objects. The calibration of stereoscopic cameras plays an important role in determining the accuracy of 3D reconstructions and 3D measurements. For a single camera, calibration involves identifying the camera’s intrinsic parameters and the lens distortion parameters. The intrinsic camera parameters relate an object to its image in the camera image plane, and the lens distortion parameters characterise the distortion effects of the lens. Multiple camera systems include a further step to identify extrinsic camera parameters that specify the 3D positions of each camera in a global world-coordinate system. The extrinsic parameters of cameras are required for estimating the 3D position of material points on the object from two-dimensional (2D) camera images.
Robust and Accurate Multiple Cameras Stereographic Calibration

In general, calibration methods can be divided into two main groups: traditional template-based methods [154]; and self-calibration, or motion-based methods [155]. Even though self-calibration methods have simplified and shortened the process of camera calibration by eliminating the need for using a calibration template, they cannot provide high accuracy since the corresponding control points can be difficult to accurately identify and match across the full fields of all cameras. Template-based calibration methods are the most commonly used methods for applications that require accurate camera parameters, such as experimental mechanics [156].

The two calibration methods proposed by Tsai [157], and Zhang [158] are the most widely used classical template-based methods [105]. In these methods, a set of 3D control points is extracted from images of a calibration template of a known shape and size. Checkerboard patterns with known square sizes are the most commonly used templates, since it is relatively easy to extract the control points at the checkerboard corner locations. The calibration template of the method of Tsai [157] can be a 2D or a 3D object, while the method of Zhang [158] is specifically developed for 2D planar calibration patterns. In both of these methods, the extracted control points of the calibration template are used to estimate the intrinsic parameters of the cameras. In several studies, it was shown that the method of Zhang [158] could estimate the camera parameters more accurately than the method of Tsai [157]. However, Zhang’s method [158] was more time-consuming, and computationally intensive because of its inefficient and slow optimisation process that estimates the parameters using many calibration images [159–161].

The method of Zhang [158] has been widely used in computer vision applications, and is available in various implementations. The camera calibration toolbox for Matlab, developed by Bouguet [162], is a popular implementation of this method. This Matlab toolbox is an extension of the method of Zhang for multiple cameras and is based on the camera model proposed by Heikkila and Silven [163]. Bouguet’s toolbox is freely available for download. However, it requires significant user input, including manual selection of the origin on each of
the checkerboard images. This is a time-consuming process, particularly for more than two cameras. The C language implementation of this toolbox is included in the OpenCV library [164]. The built-in automated checkerboard corner-finding algorithm of OpenCV’s camera calibration function has eliminated the need for manual input [165]. However, the OpenCV calibration function is only available for two-camera systems [166]. Even though the OpenCV camera calibration function is a compiled code, its checkerboard corner-finding algorithm, and its optimisation process, are time-consuming. Furthermore, the optimisation process of this code cannot easily handle more than 80 calibration images. Limiting the number of calibration images is a barrier to accurate camera calibration because the whole field of view often cannot be covered with such a small number of calibration images. In addition, having more images helps to average out noise from the data set. Recently, a stereo camera calibration toolbox was added to the computer vision toolbox of Matlab [167], which has a faster checkerboard corner detection algorithm compared with the OpenCV stereo calibration, but still has a limit on the maximum number of calibration images. The developers of this toolbox recommended using between 10 and 20 pairs of calibration images [167], whereas in many applications it is not feasible to adequately cover the field of view with such a limited number of calibration images. Furthermore, this toolbox is able to calibrate only two cameras. Developing an accurate and robust algorithm for calibrating systems with more than two cameras thus remains a challenging, yet important, area of research.

Numerous approaches have been proposed in the literature for increasing the accuracy or the speed of camera calibration algorithms. In one approach [168, 169], decreasing the lens distortion effects was targeted in 3D computer vision systems. This approach is effective for 3D systems that have high levels of lens distortion, such as omnidirectional cameras, wide-angle lenses, and fisheye lenses. In a second approach [170], the accuracy of finding control points was addressed by improving the checkerboard corner-finding algorithm, or proposing new calibration targets. Some examples of the proposed calibration targets are templates that comprise circular control points [171], two orthogonal 1D objects [172], or four collinear 1D markers [173]. These methods were developed to increase the accuracy of finding the intrinsic camera parameters, which is only one aspect of calibrating multi-camera systems. Even though
Robust and Accurate Multiple Cameras Stereographic Calibration

Extrinsic calibration has an important influence on the accuracy of multi-camera systems, and is typically the most time-consuming part of multi-camera calibration, it has yet to be adequately investigated. The most common way of finding the extrinsic parameters of multi-camera systems is using a two-step method. In the first step, the cameras’ intrinsic parameters and initial estimates of the extrinsic parameters are identified, and this is followed by a nonlinear least-squares minimisation process to refine the extrinsic parameters [153, 154, 174, 175]. A widely used objective function for this optimisation process is the sum of reprojection errors for all the calibration images for all the cameras (the reprojection error for each calibration image is the sum of squared 2D distances between the measured and the control points projected on the camera image plane). Subsequently, the initial estimates of camera parameters are refined by minimising the reprojection error in a nonlinear least-squares optimisation. A similar procedure is used for feature-based 3D reconstruction and photogrammetry using the bundle adjustment algorithm [176] (the history and methods of photogrammetric bundle adjustment were reviewed by Triggs et al. [177]).

An important aspect of the methods based on minimising the reprojection error is the optimisation procedure. Inefficient optimisation algorithms, or poorly defined objective functions, limit the performance of the procedure. For instance, a limited number of calibration images was used in [167, 174], the optimisation process was slow or ineffective for simultaneous parameter optimisation in [178], accurate initial values were required for the parameters in [175], and the camera intrinsic parameters were not included in the optimisation process in [179]. These limitations are especially problematic in 3D systems that have more than two cameras. To simplify the problem, albeit at the cost of reducing the accuracy, stereo-pair calibration was used in [153], or initial values were estimated from stereo-pairs in [174]. The use of stereo-pairs is inadequate to address the limitations of these traditional methods of calibrating multiple cameras, since the extrinsic parameters are coupled in a multi-camera system. Even though some methods have been proposed in the literature for increasing the accuracy and/or the speed of calibrating multi-camera systems, most of them have been
developed for only pairs of cameras. Zhao et al. [179] proposed to add structured light and phase-matching to the bundle adjustment algorithm to increase the accuracy and speed of calibrating stereo-cameras for large field of view measurements. However, structured light equipment is expensive, requires an additional calibration, and is not suitable for close range applications. Furthermore, Zhao et al. [179] assumed that the intrinsic parameters were found accurately, and neglected the correlation between the camera intrinsic and extrinsic parameters in the parameter estimation process by only including extrinsic parameters in the optimisation process. Huang et al. [180] used the sum of reprojection errors for camera calibration, but added geometric constraints to the objective function of their method, improving the accuracy, and increasing the convergence rate for a single camera calibration. Nevertheless, the method of Huang et al. [180] was not extended to more than one camera. Ren et al. [175] added an a priori distortion correction function to their optimisation process for calibrating their stereo camera system, but their inclusion of the lens distortion parameters in the optimisation process only addressed one of the problems of traditional methods for two cameras.

To address the limitations of the traditional methods of calibrating multi-camera systems, a novel parameter estimation procedure and a new objective function for a robust and fast multi-camera calibration are proposed. This calibration method has number of significant advantages over existing traditional methods, including the ability to: optimise the intrinsic and extrinsic parameters of all the cameras using a single optimisation; estimate the parameters from imprecise initial values; calibrate many cameras simultaneously; and use a large set of calibration images. In addition, the optimisation process of the proposed method provides a faster convergence rate compared to traditional methods, and can estimate the camera parameters for situations in which many traditional methods fail to converge to accurate parameter values. The proposed optimisation procedure can be used as the final step of any calibration method to identify the camera parameters.

Section 8.3 provides a brief overview of the theory and principles of multi-camera calibration. The proposed method of optimising the unknown parameters of a multi-camera
system, the method of validating results, and the method of analysing the performance are described in Section 8.4. The multi-camera system used in the tests and experiments is introduced in section 8.5. Results are presented and described in Section 8.6. Finally, Section 8.7 is dedicated to discussion and conclusions.

8.3 Theory and principles

8.3.1 The pinhole camera model

The pinhole camera model is a commonly used simple mathematical representation of a camera without a lens and with a very small aperture opening. This model is useful to solve the camera equations with geometric optics, and was used in the calibration method of Tsai [157] and Zhang [158]. In the pinhole model, the camera maps the physical location of 3D points to 2D coordinates in the camera image plane using:

$$ s\hat{p} = K [R_M|T] \hat{P} $$

where, $\hat{P}$ are the homogeneous coordinates of a point in 3D, $\hat{p}$ are the homogeneous coordinates of the corresponding point in the camera image plane, $s$ is a scale factor, $K$ is a matrix that contains the intrinsic camera parameters, and $R_M$ and $T$ are, respectively, the rotation matrix and translation vector between the world and the camera coordinate systems. The rotation matrix ($R_M$) may be converted to the rotation vector ($R_V$) using the Rodrigues formula [176] to reduce the number of parameters from nine to three. $K [R|T]$ is also known as the camera projection matrix. The intrinsic matrix ($K$) is unique for each camera, and is given by:

$$ K = \begin{bmatrix} f_x & \alpha & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} $$

where $f_x$, $f_y$ are the focal lengths, $\alpha$ is the skew, and $c_x$, $c_y$ are the principal points.
where, $f_x$ and $f_y$ are scale factors in the $x$ and $y$ coordinate directions, $\alpha$ is the skew between $x$ and $y$ axes, and $c_x$ and $c_y$ are respectively the $x$ and $y$ coordinates of the principal point in the camera image plane. Expanding Equation 8.1 using Equation 8.2, the relation between the physical 3D position of a point $([x, y, z])$ and its corresponding pixel position $([u, v])$ in the camera image plane is found using Equation 8.3.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & a & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$  \hspace{1cm} 8.3

### 8.3.2 Lens distortion model

The pinhole camera model in Equation 8.3 represents the camera aperture as a small hole that can focus light perfectly on the camera image plane. However, in practice, lenses are required to focus the light, which distorts the camera images. Lens distortions are defined mathematically as displacements between the observed pixel positions of the image features and their calculated positions $[u, v]$ in Equation 8.3. Radial and tangential distortions [105] are the two most common mathematical models for lens distortions in the method of Zhang [158]. Radial distortion is typically corrected in camera images using Equation 8.5.

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \left( 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 \right) \begin{bmatrix} x_u \\ y_u \end{bmatrix}$$ \hspace{1cm} 8.4

where, $(x_d, y_d)$ are distorted locations, $(x_u, y_u)$ are undistorted locations, $k_i$ are the distortion coefficients, and $r$ is the distance of the undistorted locations from the principal point. The radial distortion model of Equation 8.4 is in the form a Taylor series expansion around the principal point (or the image centre), and is symmetric about the centre. The tilting between the $x$ and $y$ axes is often modelled as a tangential distortion, with two distortion coefficients, $p_1$ and $p_2$. Equation 8.5 shows formulae for correcting radial and tangential distortions proposed by Brown [181].

$$x_d = x_u \left( 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 \right) + \left[ 2p_1 x_u y_u + p_2 \left( r^2 + 2x_u^2 \right) \right]$$ \hspace{1cm} 8.5
Robust and Accurate Multiple Cameras Stereographic Calibration

\[ y_d = y_u (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + [p_1 (r^2 + 2 y_u^2) + 2 p_2 x_u y_u] \]

Tangential distortions are asymmetric about the centre, and the corrected positions of points depend on both of the current distorted \( x \) and \( y \) positions (\( x_d \) and \( y_d \)). In addition to radial and tangential distortion models, the thin prism model [182] was proposed for correcting lens distortions in the method of Zhang [158]. The thin prism model removes the lens distortion by adding the optical properties of a thin prism to the lens model. However, in many applications, adding the prism distortion model into the distortion removal function increases the complexity without providing much improvement in the accuracy. Note that the lens distortion models for the method of Zhang [158] are inverse mapping functions that map undistorted locations to distorted locations.

8.3.3 Calibration of intrinsic camera parameters

The classic camera calibration methods of Tsai [157] and Zhang [158] are two-step methods, in which the initial values for the unknown parameters of the camera and the lens are found in the first step, and a nonlinear optimisation process refines the parameters in the second step. Even though the means of finding the initial parameters are different in these two methods (details are described in [105]), the same optimisation process can be used to refine the parameters in the second step. Zhang [158] proposed to use the reprojection error in the objective function of the optimisation process, since it relates the known coordinates of the 3D control points to the unknown parameters of the camera and lens distortion coefficients. The reprojection error for a set of 3D control points is the sum of square distances between the coordinates of the reprojected 3D points (\( M \)) and the measured corresponding points in the camera image (\( m \)), as expressed in Equation 8.6.

\[
\sum_{n=1}^{N} \sum_{l=1}^{L} \left\| m_{(l,n)} - M_{(l,n)} \right\|^2
\] 8.6
where \( N \) is the number of calibration images, \( L \) is the number of control points in one calibration image (i.e. number of corners in the checkerboard example), and \( \| . \|^2 \) represent the \( L_2 \)-norm. The camera projection matrix was used to estimate the locations of \( M \) in Equation 8.6. However, radial and tangential lens distortion coefficients can also be incorporated into the reprojection error to map the distorted pixel locations to the undistorted locations. Therefore, \( M \) can be a function of camera intrinsic parameters (\( K \) in Equation 8.2), the camera 3D pose (\( R_Y \) and \( T \)), and lens distortion coefficients (\( k_1, k_2, k_3, p_1 \), and \( p_2 \) in Equation 8.5), as indicated in Equation 8.7.

\[
M = f(K, R_Y, T, k_1, k_2, k_3, p_1, p_2) \tag{8.7}
\]

The camera parameters can be optimised by substituting Equation 8.7 in Equation 8.6, and minimising the reprojection error in an optimisation process, as indicated in Equation 8.8.

\[
\arg\min_{K, R, T, k_1, k_2, k_3, p_1, p_2} \sum_{n=1}^{N} \sum_{l=1}^{L} \| m_{(l,n)} - f(K, R_Y, T, k_1, k_2, k_3, p_1, p_2) \|^2 \tag{8.8}
\]

Equation 8.8 represents a nonlinear least-squares problem to minimise the reprojection errors. Zhang [158] proposed using the Levenberg-Marquardt (LM) algorithm for this optimisation process. The LM algorithm is relatively simple to implement and is the most commonly used algorithm for minimising the reprojection error. The details of the LM algorithm and its convergence criteria can be found in [183].

### 8.3.4 Calibration of extrinsic camera parameters

Equation 8.8 is extendable to multiple cameras by adding the reprojection errors of the images of each camera to the optimisation process, and selecting a common origin to define the coordinates of 3D points, and the \( R_Y \) and \( T \) vectors of all the cameras in a world coordinate system. The \( R_Y \) and \( T \) vectors of each camera locate the 3D position and orientation of a camera in the world coordinate system (i.e. relate the camera coordinate system to the world coordinate system). The process of finding the \( R_Y \) and \( T \) vectors for each camera is called extrinsic camera calibration. The optimisation process of Equation 8.8 can be
Robust and Accurate Multiple Cameras Stereographic Calibration

extended to Equation 8.9 to minimise the sum of reprojection errors for all cameras to determine the refined parameters for each camera. The parameters in Equation 8.9 can then be optimised for all of the cameras.

\[
\arg\min_{[K],[R],[T],[k_1],[k_2],[k_3],[p_1],[p_2]} \sum_{c=1}^{C} \sum_{n=1}^{N} \sum_{l=1}^{L} \left\| m_{c(l,n)} - M_{c(l,n)}(K_c, R_c, T_c, k_{c,1}, k_{c,2}, k_{c,3}, p_{c,1}, p_{c,2}) \right\|^2
\]

Equation 8.9

\(C\) in Equation 8.9 is the number of cameras, and the parameters inside square brackets each represent arrays of values for each of the cameras.

The tangential distortion parameters \((p_1, p_2)\) and the \(\alpha\) parameter in the \(K\) matrix (8.2) have similar distortion effects on the images. Therefore, in practice, when \((p_1, p_2)\) are being taken into account, \(\alpha\) becomes very close to zero, and can be neglected. Each camera thus adds 15 parameters (3 for the rotation vector, 3 for the translation vector, 4 for the intrinsic camera parameters, and 5 for the lens distortion coefficients) and \((N \times L)\) 3D control points to the optimisation process of Equation 8.9.

8.4 Method

The proposed multiple camera calibration method is a two-step process. Initial values of the intrinsic and extrinsic parameters are found in the first step (Section 8.4.1), and then used to determine the optimal parameters in the second step (Section 8.4.2). The methods for validating the results and analysing the performance of the optimisation process are described in Section 8.4.3 and Section 8.4.4, respectively.

8.4.1 Finding the initial values of the camera parameters

The initial values of the camera and lens parameters were determined using a method similar to that of Zhang [158] and its OpenCV implementation for calibrating a single camera [164].
The OpenCV algorithm for finding the checkerboard corners [165] was modified to increase its accuracy and speed. The OpenCV checkerboard corner finding algorithm is a two-step method; in the first step, the checkerboard is detected in an image and initial values for the corner locations are found; in the second step, the corner locations are refined to subpixel accuracy using intensity gradient directions in an iterative process [184]. This process was found to be particularly slow in the first step. To address this, a preprocessing step was added to improve the corner detection. This step converted the colour images to greyscale images, and applied a 2D median filter with a neighbourhood size of 3 pixel × 3 pixel. The median filter improved the performance of the algorithm for finding the checkerboard corners by removing some of the camera sensor noise and sudden changes in the illumination. However, median filters shift the borders of the image (i.e. the corner locations of the checkerboard), so were only used in the first step for finding the initial locations of the corners. The unprocessed images were used for the next step of the algorithm for finding the locations to subpixel accuracy. The neighbourhood size of the gradient search was chosen according to the checkerboard square size in the second step of the OpenCV algorithm for finding the locations of checkerboard corners to subpixel accuracy.

The initial values for the extrinsic parameters of the cameras (i.e. 3D camera positions and orientations) were found using 3D pose estimation for a set of images taken at various positions and orientations of the checkerboard template. The initial values of the tangential distortion coefficients and $\alpha$ in the camera intrinsic parameter matrix were assumed to be zero at this step to facilitate the OpenCV optimisation process by reducing the number of parameters. The tangential distortion coefficients were later estimated in the global optimisation process for estimating the parameters. The images of the same 3D position of the calibration template differ across the cameras, since in each camera the images are based on that camera’s coordinate system. The perspective transformation (homography) between the corner locations in the image and their corresponding metric locations in the ideal calibration template was used to estimate the 3D position of the checkerboard template in each image. The random sample consensus (RANSAC) algorithm [185] was used to find the perspective transform, assuming that the ideal calibration template was placed at the $z = 0$
plane of the camera coordinate system with its first corner located at \([x \ y \ z] = [0 \ 0 \ 0]\). With this assumption, the 3D position of the checkerboard template in the camera coordinate system was found for each calibration image and in each camera. Next, the 3D positions of the cameras were estimated using the known 3D positions of the checkerboard template and by choosing a world coordinate system (i.e. a common origin). To simplify the process of adding extra cameras to the stereoscopic system, the coordinate system of one of the cameras was selected as the world coordinate system, and transformed the coordinate systems of the other cameras to this common coordinate system. Assuming that the 3D positions of the checkerboard template in the coordinate systems of both cameras are known, and that the world coordinate system is selected to be the coordinate system of camera 1, the 3D position of camera 2 in the world coordinate system could be found using:

\[
R_r = R_2 \times (R_1)^T \tag{8.10}
\]

\[
T_r = T_2 - (R_r \times T_1) \tag{8.11}
\]

where, \((R_1, T_1)\) and \((R_2, T_2)\) are matrices indicating the 3D position of the checkerboard template in the coordinate systems of camera 1 and camera 2, respectively, and \((R_1)^T\) denotes the transposed matrix of \(R_1\) (i.e. the inverse rotation). The \((R_r, T_r)\) values in Equation 8.10 and Equation 8.11 are estimated for each calibration image and vary across a set of images. However, a single set of values should be chosen for the 3D position of camera 2. Because some of the estimated values are outliers, the selection of the average values of all the estimated \((R_r, T_r)\) matrices as the \((R_r, T_r)\) of the camera will introduce error. Therefore, to minimise the error, the proposed method first sorts the values of the estimated \((R_r, T_r)\) matrices, and then uses the average of the three middle sorted values of each camera as the estimate of its \((R_r, T_r)\) matrices.

Note that, even though the methods described in this section for finding the initial values are able to find relatively good initial values, it is demonstrated below that the proposed
algorithm can reliably estimate the parameters using significantly inaccurate initial values. For instance, the initial values for the camera parameters and their 3D positions could be chosen purely based on the lens and camera specifications and the physical configuration of the cameras.

8.4.2 Optimising the intrinsic and extrinsic camera parameters

The values found in the first step were used as initial estimates in the next step to refine the parameters in an optimisation process. Several novel parts are proposed in this optimisation step including, a novel objective function (Section 8.4.2.1), a new optimisation algorithm, a novel optimisation procedure (Section 8.4.2.2), a method to remove outliers (Section 8.4.2.3), and a method to increase the convergence rate of the optimisation process (Section 8.4.2.4).

8.4.2.1 Objective function

The reprojection error (Equation 8.9) has often been used as the objective function when refining the parameters in traditional methods of multi-camera calibration. Even though the reprojection error defines a good relation between the 3D points and the camera parameters for the optimisation process, it is not sufficient for a fast and accurate optimisation. This issue has been, to some extent, addressed in the literature. For instance, Huang et al. [180] added constraints, based on the 2D geometry of the calibration template, to the optimisation process of a single camera to increase the accuracy of estimating the parameters. Even though adding constraints improves the convergence rate, constraints provide limited information to the optimisation process. The optimisation process could be improved in multiple camera systems, where corresponding points can be triangulated [176] and the calibration templates can be reconstructed in 3D.

Two error functions were defined based on the 3D information of the reconstructed calibration template, and were added to the objective function of the proposed method along with reprojection errors. Addition of these error functions helps to increase the accuracy and robustness of the optimisation process when calibrating multiple cameras. The first defined error function used 3D length errors, which were calculated by subtracting the 3D distances between adjacent triangulated checkerboard corners from the known square size of the
checkerboard template (i.e. 3D length errors). The second defined error function was the difference between the 3D reconstructed corners of the calibration template and the corners of an ideal template (i.e. 3D shape errors). 3D shape errors were calculated by measuring the Euclidian distances between the triangulated corners and the expected 3D locations of the corners of an ideal checkerboard in the same location as that of the calibration template. However, the square size of the ideal checkerboard template may have been uniformly scaled due to the uncertainty in finding the 3D location of the calibration template. Thus, 3D shape errors can only illustrate the geometric variation of the triangulated corners from an ideal checkerboard template, without taking into account the correct square size. For this reason, 3D shape errors were used in combination with 3D length errors (i.e. the first defined error function) to take into account the correct scale.

Even though adding these two error functions to the objective function of the optimisation process helps to increase the accuracy and speed, there remain some challenges that need to be addressed. The 3D length and 3D shape errors are measured in meters, but the reprojection errors are measured in pixel units. Different units cannot be combined effectively together in a single objective function, since they generally have different scales and physical meanings. To overcome this issue, the average pixel size of the field of view was approximated in meters, and was used to convert the units of the 3D length and 3D shape error functions to pixel units. Thus, all the three error functions of the objective function had the same unit of pixel, which enabled us to combine each of the error terms in a consistent objective function.

The errors associated with the lens distortion were taken into account in the objective function by mapping the distorted checkerboard corners to undistorted locations prior to estimating the error functions at each iteration of the optimisation process. The radial and tangential distortion model (Equation 8.5) was used for this purpose.
8.4.2.2 Optimisation algorithm and optimisation process

Equation 8.8 is an effective optimisation method for finding the parameters of a single camera, but the extension of this optimisation algorithm for multiple camera calibration (Equation 8.10) can be problematic. The number of parameters and the control points of the optimisation process in Equation 8.9 (checkerboard corners for a checkerboard template) increase substantially when cameras and calibration images are added (each camera adds 15 extra intrinsic and extrinsic parameters, and each calibration image adds extra control points (see Section 8.3.4)). For instance, for a system with four cameras and a dataset of 80 calibration images of a checkerboard of size 9 squares × 12 squares (8 × 11 inner corners as control points), 60 parameters (i.e. 4 × 15) need to be optimised using 28160 (i.e. 8 × 11 × 4 × 80) control points. The inability of existing multiple camera calibration methods that use the LM algorithm to effectively handle a large number of parameters and control points is the main cause of many of their limitations. The solutions to this problem have mainly focused on using stereo-pairs (e.g. [153]), or limiting the number of calibration images (e.g. [174]), both of which compromise the accuracy of camera calibration. This issue was addressed by proposing to use a more robust optimisation algorithm than the LM algorithm and a new optimisation process.

Despite the popularity of using the LM algorithm for minimising the sum of reprojection errors (Equation 8.10) for camera calibration, this algorithm is not well suited to the task. The LM algorithm works well for problems with small residuals at optimality, but not when the sum of squares is large. This arises because the Hessian matrix is approximated from only the first derivative information, which is only reasonable when the residual is small. In a similar problem, Lourakis and Argyros [186] showed that the LM algorithm was not the most efficient algorithm for bundle adjustment in feature-based 3D reconstruction (bundle adjustment methods are described in [105] and [177]). There have been some efforts to decrease the computation time of the LM algorithm in the bundle adjustment application, such as a sparse implementation in [187], and a multi-core implementation in [188]. However, while these approaches improve the speed and memory usage of the LM algorithm, they are not able to address the other limitations in handling a large number of parameters and control points.
Recently, trust-region methods have been used to minimise the error in nonlinear least-squares problems. Trust-region methods have very reliable convergence properties [183], and have been shown to outperform the LM algorithm, particularly for solving problems with a sparse structure, such as bundle adjustment [186], and simultaneous localisation and mapping (SLAM) [189]. Minimisation of the reprojection error for camera calibration is another example of a problem with a sparse structure. The 3D length and 3D shape errors in the proposed objective function (see Section 8.4.2.1) also have sparse structures. Because of the advantages of trust-region methods over the LM algorithm, including their superior performance in solving sparse problems, the trust-region-reflective algorithm was chosen to optimise the camera parameter values.

Conventional methods of calibrating multiple cameras minimise the error using the sum of reprojection errors for all of the cameras (Equation 8.10). The reprojection errors are assumed to be distributed uniformly over the field of view and across the cameras. However, due to the lens distortion effects, the reprojection errors are higher near to the peripheries of the images, and can vary over the field of view and across the cameras of a multi-camera system. Even though the three error functions of the proposed method have the same unit of pixel and are able to be summed consistently, the sum of errors was not used. Instead, an error in the form of a matrix was used in the trust-region-reflective algorithm as follows:

$$\arg\min_{[k],[R],[\tau],[k_1],[k_2],[k_3],[p_1],[p_2]} \begin{bmatrix} RE_{(1,1)} & \ldots & RE_{(C,1)} & LE_1 & SE_1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ RE_{(1,L \times N)} & \ldots & RE_{(C,L \times N)} & LE_{(L \times N)} & SE_{(L \times N)} \end{bmatrix}$$

where $L$ is the number of checkerboard corners, $N$ is the number of calibration images, $C$ is the number of cameras, $RE$ is the reprojection error, $LE$ is the length error, $SE$ is the 3D shape error, and all are measured at each checkerboard corner location (note that $LE$ and $SE$ are 3D measurements, and are only defined once for a system with multiple cameras). In the error matrix of Equation 8.12, no assumption was made about the distribution of the errors,
and comprehensive error information was provided for the optimisation algorithm at each checkerboard corner location of each image across all the cameras. The error matrix (Equation 8.12) was estimated in the optimisation process after undistorting the images. Thus, in addition to the camera parameters, lens distortion parameters were included in the optimisation process for removing the distortion effects from the calibration images.

8.4.2.3 Removing the outliers

The accuracy of finding the checkerboard corners is dependent on a number of factors, such as the quality of the calibration images. The process of corner finding is error-prone when images are noisy, blurred, or have areas with specular reflections. As a result, the error measurements can be inaccurate for such images. The quality of the optimisation process can be improved by removing the outliers caused by the failure of the corner finding algorithm. In the proposed method, the outliers were detected and removed from the error matrix defined in Equation 8.12 based on the average reprojection error values in the whole dataset for each camera. The average reprojection error value was calculated for each camera using:

$$Average \ RE = \frac{\sum_{k=1}^{L \times N} RE_{(c,k)}}{L \times N}$$  

8.13

Reprojection error values that were greater than 10 times the average reprojection error estimated using Equation 8.13 were identified as outliers, and were removed from the error matrix in Equation 8.12. The idea here was to exclude outliers, but to include points that may have larger error due to lens distortions, such as the images with checkerboard corners at the peripheries. The error margin of 10 times greater than the average error guarantees that the points are outliers.

8.4.2.4 Increasing the convergence rate

An optimisation problem is poorly scaled if variations in any of the parameters alter the objective function significantly more than variations in the other parameters [183]. The calibration parameters for a system with multiple cameras have various ranges and units. For instance, rotation parameters range between 0 to 2π radians, while the translation vector parameters is in units of metres and can vary considerably. Furthermore, objective functions
have different levels of sensitivity to the input calibration parameters. Despite the importance of having a well-scaled optimisation problem, this issue has not been considered in the literature for camera calibration. In the proposed method, this issue was addressed by scaling the camera parameters prior to using them as the inputs of the optimisation process. The scaling allows us to employ a single optimisation process to optimise all of the input parameters. The scaling of parameters in the proposed method was performed in two steps. In the first step, the input parameters of each camera were divided into groups that have the same units and similar magnitudes. The parameter groups were: the camera focal length, principal point, lens distortion coefficients, rotation vector, and translation vector. The parameters of each group were normalised with respect to the largest value of that group. In addition to the scale of the camera parameters, the sensitivity of the objective function to changes of the parameters also affects the convergence rate and robustness of the optimisation process. This was addressed in the second step of the scaling process, where the groups of input parameters from the first step were scaled based on the sensitivity of the objective function to the changes of that group of parameters. The sensitivity of the objective function to each input parameter is often indicated based on its partial derivatives [183], which can be estimated using the Jacobian matrix \( J \) of the objective function defined in Equation 8.12 \((O) \) [183] using:

\[
J(x_1, \ldots, x_{15 \times C}) = \begin{bmatrix}
\frac{\partial O_1}{\partial x_1} & \cdots & \frac{\partial O_1}{\partial x_{15 \times C}} \\
\vdots & \ddots & \vdots \\
\frac{\partial O_{NE}}{\partial x_1} & \cdots & \frac{\partial O_{NE}}{\partial x_{15 \times C}}
\end{bmatrix}
\]

where \( x_i \) are the camera calibration parameters, and \( NE \) is the number of elements in the error matrix in Equation 8.12 \((NE = L \times N \times (C + 2))\). The average values of the columns of Equation 8.14 are an approximation of the partial derivative of the objective function \((O)\) with respect to each parameter, \((x_i)\), and were thus used as the measures of the sensitivity of the objective function to each parameter. In this second step of scaling the parameters, the groups
of input parameters were scaled according to the average value of the columns of the $J$ matrix, so that more sensitive parameters become larger and vice versa. This results in a well-scaled problem, for which any changes in the parameters have a similar effect on the error functions (or the objective function). Note that all of the input camera parameters were scaled using this two-step method prior to using them as inputs of the objective function while the scaling values were constant during the optimisation process. The input parameters were scaled back to their original values to estimate the error parameters inside the objective function.

8.4.3 Validation of the results

The average reprojection error of the control points (i.e. the checkerboard corners in the case of a checkerboard template) is the most common metric for assessing the quality of the camera calibration (e.g. [174][175][180]). Instead of the average error, the root mean squared (RMS) error was used, which gives a higher weight to large errors, to assess the calibration result. The reprojection error is very dependent on the set of images used for the calibration. To illustrate the effect of this dependency, the RMS reprojection error was computed for a set of 100 images and a subset of 20 images randomly selected from that dataset.

In addition to reprojection errors, the quality of the extrinsic camera parameters was assessed using epipolar lines [176]. Epipolar lines describe the geometric relations between the 3D points in two cameras, where a point in one camera corresponds to a line in the second camera (refer to [176] for further details). Epipolar lines have been previously used to assess the quality of camera calibration, particularly for self-calibration methods that do not have a calibration template [190]. In the proposed method, the epipolar lines were estimated for stereo-pairs using the fundamental matrix [176] of the cameras. As with the method used by Furukawa et al. [190], the distances between the epipolar lines and the control points were used to visually assess the quality of the extrinsic camera calibration.

The number of calibration images, and the field of view that they cover directly influence the average reprojection error in the calibration process. When using different image datasets, the same multi-camera system and algorithm will yield different average reprojection errors (such as in [174]). The average reprojection error is generally lower for datasets in which most
calibration images are nearly parallel to the camera image plane, have less variation in the position and orientation, or are concentrated in the central part of the field of view (where lens distortion is low). However, the unknown parameters cannot be estimated accurately with inadequate input data due to a limited number of images. To have a realistic reprojection error assessment for the proposed method, a calibration image dataset was used that covered the whole field of view and had sufficient variation in the angle and distance to the cameras.

The effectiveness of accounting for lens distortion effects was also examined for the proposed method. Lens barrel distortion causes straight lines to become curved, while the distortion correction process should be able to remove this effect. The accuracy of lens distortion coefficients was evaluated by removing distortion from the checkerboard images of a test dataset (a dataset that has not been used for performing the camera calibration) and fitting lines to the 2D corner positions from the same row (or column) of the checkerboard template. The total error of the line fits for rows (or columns) of checkerboard images was used as an indication of the quality of the distortion correction process. Even though the line fitting error includes the errors introduced by the corner finding algorithms, removing the lens distortion effects should decrease the line fitting error.

8.4.4 Performance analysis

The robustness of the proposed optimisation process to the use of inaccurate initial values was assessed by adding random perturbations to all of the camera parameters used in the optimisation process (i.e. the intrinsic and extrinsic parameters of the cameras), and checking the convergence rate of the optimisation process as well as the final errors. The perturbation was randomly chosen from a uniform distribution in the range of -25% to +25% of every parameter, and was added to all of the input parameters prior to the optimisation process.

The roles of the novel parts of the proposed method (3D error functions and the parameter weightings) in the optimisation process were examined by separately removing each of them from the optimisation process, and comparing the final reprojection errors and the
convergence rates of the optimisation process with the case where all parts had been included. In these tests, the RMS length errors of the corner locations of a checkerboard template were calculated for a test dataset consisting of 50 calibration images that had not been used in the calibration process.

8.5 Experimental setup

The experimental setup used for the tests comprised four monochrome USB 3 cameras (Point Grey FL3-U3-13Y3M-C), equipped with 6 mm focal length lenses (DF6HA-1B from Fujinon) (Figure 8-1). The details of the camera setup are described in [79]. The cameras were focused on the field of view using a focusing pattern.

![Figure 8-1: The arrangement of the four cameras of the setup](image)

The checkerboard template was printed using a laser printer (Ricoh MP C2503) and was carefully glued to a 3 mm thick acrylic sheet using an adhesive spray, resulting in a flat 2D template (Figure 8-2). The checkerboard square size and number of squares were selected based on the size of the field of view, and the average distance of the field of view to the cameras. The field of view of this camera system was approximately 200 mm × 200 mm, and the average distance to the cameras was approximately 200 mm [79]. A checkerboard template of size 9 × 12 (i.e. 8 × 11 inner corners) with a square size of 6 mm was chosen as the
calibration template (Figure 8-2). 100 calibration images were taken to cover the whole field of view at various distances and angles to the four cameras of the setup. However, only 80 images were used for calibrating single cameras with the OpenCV function due to the inherent limit of this function in handling large image datasets.

![Figure 8-2: An image of the calibration template](image)

8.6 Results and discussion

The initial intrinsic parameters of the cameras were found using the OpenCV single camera calibration function and the modified corner finding algorithm. The modified corner finding algorithm performed approximately 2 times faster and provided 20 % improvement in detecting the checkerboard in the calibration images compared to the original OpenCV corner finding function. Table 8-1 lists the average reprojection errors for the initial intrinsic parameters of each of the cameras found using single camera calibration.
Table 8-1: The average reprojection error in calibrating intrinsic parameters of a single camera of the system using the modified single camera calibration function of OpenCV

<table>
<thead>
<tr>
<th>Camera Number</th>
<th>Average reprojection error (pixel)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(intrinsic parameters of a single camera)</td>
</tr>
<tr>
<td>Camera 1</td>
<td>0.169</td>
</tr>
<tr>
<td>Camera 2</td>
<td>0.177</td>
</tr>
<tr>
<td>Camera 3</td>
<td>0.150</td>
</tr>
<tr>
<td>Camera 4</td>
<td>0.207</td>
</tr>
</tbody>
</table>

The initial extrinsic parameters for the four cameras were found using the method described in Section 8.4.1. The initial values of the intrinsic and extrinsic parameters, obtained from the first step, were optimised using the method described in Section 8.4.2. Figure 8-3 shows the RMS reprojection errors, RMS 3D error, and RMS length error for a calibration dataset consisting of 100 images, and a test subset of 20 images randomly selected from that dataset. As Figure 8-3 shows, the proposed method converged after a few iterations for both calibration datasets, and yielded in small errors. The errors obtained from the optimisation process were lower for the dataset consisting of 20 images compared to the dataset consisting of 100 images, but such a small calibration image dataset cannot represent the actual intrinsic and extrinsic parameters of the cameras.

The accuracy of each set of the lens distortion coefficients and camera parameters found using the 20-image and 100-image calibration datasets was tested on a dataset consisting of 50 calibration images. Figure 8-4 shows the checkerboard corners found in one of the calibration images, and Figure 8-5 shows two sample lines fitted to one row of corners in a distorted image (on the left) and in its undistorted version (on the right). The fitting errors were calculated for all rows of all calibration images of the test dataset for distorted and undistorted images (Figure 8-6). The images were undistorted using the camera parameters estimated from 20 calibration images and 100 calibration images. Although the average fitting error was decreased for both sets of parameters, the average errors were lower in all the cameras for the camera parameters estimated from 100 calibration images compared to the one estimated from 20 calibration images (Figure 8-6). This indicates that the reprojection error is not the only factor that indicates the accuracy of camera calibration.
The accuracy of the estimated camera parameters was further investigated by finding an epipolar line in one of the cameras and visually comparing that with its corresponding point in the reference camera (Figure 8-7). The epipolar lines were found using the initial camera parameters (Figure 8-8), parameters estimated from 20 calibration images (Figure 8-9), and parameters estimated from 100 calibration images (Figure 8-10). Although the initial parameters had relatively small errors (the starting points of plots in Figure 8-3), the epipolar line of Figure 8-8 visually reflects this error (i.e. the point and the epipolar line do not correspond to the same location of the checkerboard image). While it was difficult to distinguish the difference between the accuracy of the epipolar lines in Figure 8-9 and Figure 8-10, both of them show a close correspondence to the points in the left images of Figure 8-9 and Figure 8-10.
Figure 8-3: The RMS reprojection errors, the 3D RMS error, and the RMS length error for a dataset consisting of 100 images and a subset of 20 images randomly selected from that dataset.
Figure 8-4: Red crosses show the checkerboard corners found in one of the calibration images from the test dataset (the image has been cropped for visualisation purposes).

Figure 8-5: Two sample lines have been fitted to corners of the checkerboard pattern in a distorted image (left) and its undistorted version (right).

Figure 8-6: The average fitting error for fitting lines to rows of checkerboard corners in a test dataset consisting of 50 calibration images. The errors were calculated for distorted images, and undistorted images using parameters estimated from 20 calibration images and 100 calibration images.
Figure 8-7: An epipolar line in one the cameras (right) and its corresponding point in the second camera (left)

Figure 8-8: A sample epipolar line (right) and its corresponding point (left) found using initial camera parameters

Figure 8-9: A sample epipolar line (right) and its corresponding point (left) found using the estimated parameters from 20 calibration images

Figure 8-10: A sample epipolar line (right) and its corresponding point (left) found using the estimated parameters from 100 calibration images
Robust and Accurate Multiple Cameras Stereographic Calibration

In Figure 8-11, the proposed method is compared to the traditional method of calibrating multiple cameras using the LM algorithm for the same dataset consisting of 100 calibration images. The results in Figure 8-11 show that the proposed method not only resulted in smaller errors, but also had a faster convergence rate compared to the traditional method. Furthermore, the traditional method was unable to reduce the reprojection error as much as the proposed method even after 20 iterations, and was unable to provide any improvement in the 3D and length errors (Figure 8-11).

The robustness of the proposed method in optimising the parameters from inaccurate initial parameters was tested by adding a substantial random error to each of the initial values, as described in Section 8.4.4. Figure 8-12 shows the results of this test. As can be seen in the error plots, despite the large amount of error caused by the inaccurate initial values, the proposed method reduced the error effectively with a fast convergence rate. After 20 iterations, the average RMS reprojection error for the four cameras decreased from 230.17 pixel to 0.35 pixel, the RMS 3D error decreased from 0.800 mm to 0.065 mm, and the RMS length error decreased from 1.1561 mm to 0.025 mm. In contrast, as Figure 8-12 illustrates, the traditional method only reduced the RMS reprojection for the first five iterations, while after 20 iterations, the average RMS reprojection error of the four cameras was 25.73 pixel (73.51 times larger than the proposed method), the RMS 3D error was 1.23 mm (1.53 times larger than the proposed method), and the RMS length error was 1.04 mm (41.60 times larger than the proposed method) (Figure 8-12).

Figure 8-13 shows the contribution of two novel parts of the proposed method (two 3D error functions of the objective function and the weighting of the camera parameters) in improving the multiple camera calibration, as described in Section 8.4.4. Based on the proposed method, two alternative objective functions were compared. In case I, the 3D error functions were excluded from the objective function (orange plots in Figure 8-13), and in case II the 3D error functions were included in the objective function, but the camera parameters and error values were unweighted (blue plots in Figure 8-13). As illustrated in Figure 8-13, the RMS
Results and discussion

backprojection errors for the proposed method (green plots in Figure 8-13), case I, and case II follow a similar trend. These results indicate that the trust-region optimisation algorithm and the matrix of errors that have been proposed (see Section 8.4.2) are more efficient compared to the LM algorithm and the sum of errors in the traditional methods, which were unable to reduce the errors (blue plots in Figure 8-12). A comparison of the final RMS reprojection errors in Figure 8-13 shows that the proposed method had the smallest errors, whereas the largest errors were for case II in all the four cameras. The differences in errors were more pronounced for the RMS 3D and RMS length errors (Figure 8-13). The large 3D error in case I occurred because the optimisation process only minimised the backprojection error, and did not include any error functions related to 3D measurements. Therefore, although the reprojection errors were small, checkerboard corners were 3D reconstructed with an inaccurate square size. Case II included 3D error functions, but they were not weighted properly, or the optimisation process was not a well-scaled problem. Therefore, case II could not reduce the 3D errors (blue plots in Figure 8-13) as effectively as the proposed method with scaled parameters (green plots in Figure 8-13).

Note that the weighting of the error values and the scaling of the initial camera parameters is more influential if the ratio of the checkerboard corner size to the reprojection error is large, and/or intrinsic or extrinsic camera parameters have large variations. In the experimental setup, the ratio of the checkerboard corner size to the reprojection error was approximately $\frac{6 \text{ mm}}{0.3 \text{ pixel}} = 20 \text{ mm/pixel}$, all of the cameras and lenses were of the same type, and the configuration of cameras was symmetric (see Section 8.5). Thus, the weighting of the error values and the scaling of the initial camera parameters did not have a major effect in increasing the convergence rate and reducing the errors.

The intrinsic and extrinsic camera parameters that were estimated using the proposed method, case I, and case II in the previous step were evaluated by finding errors for a test dataset of 50 calibration images. Figure 8-14 shows the RMS length error measured using these estimated parameters in the control dataset. The RMS length error of the proposed method was 0.025 mm, which was very similar to the final estimated error in the multiple camera calibration.
process (green plots in Figure 8-13). Case I and case II resulted in errors of 1.18 mm and 0.17 mm, respectively, in estimating the checkerboard corner length. Even though the RMS reprojection errors were not markedly different between the proposed method, case I, and case II, the length estimates were inaccurate in case I and case II. This once again illustrates the effectiveness of adding 3D error functions to the optimisation process for multiple camera calibration, and appropriate weighting of the camera parameters and errors.
Figure 8-11: The RMS reprojection errors, the average length error, and 3D RMS error for a dataset consisting of 100 images for the proposed method compared to the traditional method of calibrating multiple cameras using the LM algorithm.
Figure 8-12: The RMS reprojection errors, the average length error, and 3D RMS error obtained using the proposed method (orange plots) or the traditional LM method (blue plots) with a
randomly perturbed set of initial camera parameters. Green plots show the errors for initial camera parameters without the perturbations.

Figure 8-13: The RMS reprojection errors, the average length errors, and 3D RMS errors estimated for the proposed method, case I (i.e. 3D error functions were excluded from the objective function), and case II (i.e. 3D error functions were included in the objective function, but the camera parameters and errors were unweighted).
8.7 Summary

The traditional methods of multiple camera calibration using the LM algorithm and sum of reprojection errors have a number of limitations. These methods are slow, require good initial values for the parameters of the optimisation process, can struggle with large sets of calibration images, and cannot simultaneously calibrate all of the intrinsic and extrinsic parameters of a multi-camera system. A method that could address many of these limitations was proposed. The tests showed that, although a small dataset of calibration images resulted in a smaller reprojection error compared to a large dataset (Figure 8-3), the camera parameters estimated using the small dataset are less accurate (Figure 8-6). Therefore, the RMS backprojection error by itself is not sufficient to be used as an indicator of the accuracy of the parameters estimated using a multiple camera calibration method.

The lens distortion parameters estimated using the proposed method were validated using the errors in the line fitting to the checkerboard corners (Figure 8-4 to Figure 8-6), and the extrinsic parameters were qualitatively validated using epipolar lines (Figure 8-7 to Figure 8-10). Tests showed that the proposed method had a faster convergence rate and higher...
accuracy in minimising the RMS reprojection error compared to the traditional methods of multiple camera calibration, when the initial values were relatively accurate (Figure 8-11). Furthermore, unlike the traditional methods, the proposed method was able to minimise the RMS 3D and RMS length errors (Figure 8-11). The difference between the performance of the proposed method and that of traditional methods was more substantial for inaccurate initial camera parameters (Figure 8-12). The very large final error of the traditional methods showed that they failed to calibrate a multi-camera system when the initial camera parameters were inaccurate. In contrast, the proposed method did not require accurate initial parameter values, and could minimise the backprojection and 3D errors (Figure 8-12).

The contributions of adding 3D error functions, and weighting the camera parameters and errors, were also examined (Figure 8-13). Plots in Figure 8-13 illustrated that the addition of 3D error functions helped to decrease the RMS 3D and RMS length errors, but weighting the camera parameters and errors did not have a significant effect in reducing the final errors. However, estimation of the square size of a checkerboard template in a dataset of 50 calibration images showed that both the addition of the 3D error functions and weighting the parameters helped to decrease the RMS length errors (Figure 8-14).

In summary, a new optimisation process has been proposed for calibrating multiple camera systems. The proposed method has faster convergence rate and higher accuracy compared to traditional methods using the LM algorithm and sum of backprojection errors. The proposed method is robust enough to be used for a system with many cameras and large datasets of calibration images. The new method could successfully optimise the camera parameters even with substantially inaccurate initial values, where traditional methods are unable to accurately estimate the calibration parameters of multiple camera systems.
9 A Model-based Technique for Calibration of Multiple Cameras

The content of this chapter is based largely on the following journal paper, which is in review for publication in the International Journal of Computer Vision:


9.1 Abstract

Accurate estimates of the intrinsic and extrinsic parameters of cameras in a stereoscopic system are important for achieving high precision measurements of three dimensional (3D) shape or deformation. Methods that use calibration targets (template-based), particularly those that use checkerboard templates, are the most common approaches for identifying parameters in multi-camera systems. However, methods for localising the control points of checkerboard templates (i.e. their corner locations) are often insufficiently accurate to obtain good estimates of camera parameters for high precision measurements. Some methods have been proposed previously to address this limitation, and to improve the localisation of the control points. However, these methods also have limitations as they are either unable to effectively model the lens distortion, or do not use an accurate and robust method to localise the control points.
Here, some of the limitations of existing methods are addressed, and a model-based technique for calibration of multi-camera systems is proposed. In this method, initial estimates of camera parameters are found using a recently developed multi-camera calibration method based on a checkerboard template. These parameters are then refined using a custom calibration target comprising an array of concentric circles and a reconstructed model of this calibration target. This approach enables us to simultaneously refine the control point locations and estimate the lens distortion effects. The discrepancies between the calibration target and its reconstructed model are measured using a robust and accurate algorithm for subpixel image registration. Zernike polynomials are used as mapping functions to define a forward lens distortion model.

An experimental apparatus was used to apply physical 3D shifts to a flat object in order to assess the accuracy of the estimated parameters. The results were compared to a conventional method of calibrating cameras and an accurate multi-camera calibration method based on a checkerboard template. Tests indicate that the camera parameters obtained using the new method can be used to estimate 3D shifts of a flat object significantly more precisely than the conventional methods using the Levenberg-Marquadt algorithm to minimise the sum of reprojection errors of a checkerboard template. This method is also demonstrated to achieve higher accuracies than a checkerboard-based multi-camera calibration method that uses a novel optimisation process to estimate the camera parameters.

9.2 Introduction

Maximising the accuracy of camera calibration is important in high-precision machine vision systems that perform three-dimensional (3D) measurements of surface deformation or shape. The two main techniques for camera calibration are self-calibration (or auto-calibration) and template-based methods [191]. Self-calibration methods calibrate the camera by identifying corresponding features in a series of images, while template-based methods use the known geometric properties of a calibration target to estimate the parameters of the camera. Self-calibration methods are simple and fast for unknown scenes, but these methods are less accurate and robust than template-based methods [192, 193]. The most widely used template-
based calibration method is that of Zhang [158], which uses a checkerboard pattern as the calibration template. The location of the corners of the checkerboard, the number of rows and columns of the template, and its square size are known geometric features that are used to estimate the intrinsic and extrinsic parameters of the cameras. The location of the corners in a checkerboard template are extracted using algorithms that are based on intensity gradients, such as the OpenCV corner-finding algorithm [165].

Even though template-based methods are more reliable than self-calibration methods for controlled environments, they have some limitations. For instance, the corners of the checkerboard, that are used as control points in conventional methods, cannot be localised accurately in many applications [158, 194–196]. To address this limitation, Kruger et al. [170] generated a grey-value model of the checkerboard template and performed a least-squares minimisation between the model and the calibration images, thereby improving the accuracy of the corner detection. The average displacement error in detecting the corners achieved by their method was 0.032 pixel. Another approach to address the limitation of finding the control points was to use a different calibration target instead of the checkerboard pattern. Some examples of proposed calibration targets are templates that comprise filled circular control points [171], rings or concentric circles [195–198], spheres [199], orthogonal 1D objects [172], or collinear 1D markers [173]. Unlike the checkerboard corner detection, localising the control points in concentric circles (or rings) is not dependent on the intensity gradient of a single point, and is thus less affected by camera noise and illumination variations. However, localisation of the control points of concentric circle calibration targets has some limitations. Perspective distortions in the calibration images and imperfections of the calibration target (due to manufacturing limitations) decrease the accuracy of localising the control points. For instance, Kruger et al. [170] concluded that circular calibration templates are less reliable than checkerboard templates in images that have perspective distortion and/or nonlinear lens distortions [170]. Datta et al. [194] addressed this issue by proposing an iterative refinement method to improve the accuracy of localising the control points in calibration.
images that have perspective distortions. In their method the parameters obtained from the conventional camera calibration method were used to undistort and project calibration images to canonical fronto-parallel images (i.e. images that are parallel to the image plane of the camera). The fronto-parallel images were then used to localise control points, which were projected back to recompute the camera parameters, in an iterative manner. Their method improved the localisation of control points, which resulted in a decrease in the reprojection error (distance between the observed and computed projected points) in calibration targets such as checkerboard, circle, and ring templates. The approach of using fronto-parallel images was further developed by Vo et al. in [195] and [196]. Vo et al. [195] proposed refining the control points in the fronto-parallel images by comparing the images with a synthesised image of the calibration template using digital image correlation (DIC). In another study, Vo et al. [196] used their camera calibration technique to find the 3D position of the calibration template, which was used to calibrate a fringe-projection-based system.

The approach of localising the control points in fronto-parallel images has several shortcomings. Firstly, it increases the computation time and complexity of calibrating cameras, since, for every calibration image and at each iteration, the control points must be relocalised, and the camera parameters must be re-estimated. Secondly, projecting the calibration images to fronto-parallel images has an inherent ambiguity in the unknown scale of the calibration template in these images. Thus, in the methods proposed by Datta et al. [194] and Vo et al. [195, 196], it is not possible to decrease the errors associated with inaccurate lengths between the control points of the calibration template, which are usually due to printing inaccuracies or unevenness of the calibration target. Thirdly, although relocalising the control points in fronto-parallel images increases accuracy at low levels of perspective distortion, the accuracy decreases considerably at high levels of perspective or nonlinear distortion. Inaccurate estimations of camera parameters, and the location of control points, result in nonlinear distortion in fronto-parallel images. Therefore, this method becomes ineffective in multi-camera systems where the cameras are not closely aligned. Another important limitation of the method proposed by Vo et al. [195, 196] is that it cannot take into account the errors caused by the remaining lens distortion effects in localising the control points in calibration
templates. In the method of Vo et al. [195, 196], the initial parameters of the cameras were found, and the images were undistorted using the initial lens distortion coefficients. The control points were then found in the undistorted images and were projected to fronto-parallel images, which were compared to the control points of an ideal synthesised image of the calibration template; the remaining effects of the lens distortion cannot be identified and corrected in this process. Moreover, the method of Vo et al. [195, 196] modifies the location of the control points to move them toward their ideal location in the calibration template. This process is essentially the same as the process used to estimate the reprojection error, which is the objective function of the optimisation process to minimise the error of the camera calibration. Therefore, in the method of Vo et al. [195, 196], the control points were moved in a way that minimises the reprojection error without necessarily representing the actual location of the control points. Hence, although the method of Vo et al. [195, 196] results in a small reprojection error, the error is not a true representation of the precision of the calibration process.

Douxchamps et al. [197] proposed a method for localising the control points of a calibration template that considers the lens distortion effects. The cameras were calibrated, and ray tracing was used to build a synthetic image of the calibration template at the estimated location of the calibration target in calibration images. The ray-traced model of the calibration template and the image of the calibration target were matched using an optimisation process to maximise the match between their bright and dark areas, which corresponded to high and low intensities, respectively. Douxchamps et al. [197] assumed that nonlinear distortions were negligible, the light intensity was uniform, and the calibration target was a Lambertian surface (i.e. isotropic surface luminance). However, in practice, it is very difficult to satisfy the latter two assumptions. Moreover, the matching criteria require a time-consuming optimisation, which is sensitive to local changes in illumination in the calibration images.

Some of the limitations of existing camera calibration methods are addressed in this chapter by developing a novel model-based technique for localising the control points of a calibration
template when calibrating cameras. This method does not require an iterative estimation of control points, and thus results in a faster calibration process compared to methods that use fronto-parallel images. In contrast to the methods proposed by Datta et al. [194] and Vo et al. [195][196], the model-based method proposed in this chapter can accurately estimate and correct for lens distortion effects. The estimation of lens distortion effects is performed using a recently developed algorithm known as phase-based Savitzky-Golay gradient-correlation (P-SG-GC) [35] to measure the discrepancies between the calibration images and the reconstructed model of the calibration template. The P-SG-GC algorithm has been shown to be very accurate and robust in estimating translational shifts between images [35]. Furthermore, the P-SG-GC algorithm [35] is faster and more robust to changes in illumination compared to the intensity-matching method of Douxchamps et al. [197].

The lens distortion model is another important factor that affects the accuracy of camera calibration. Several models have been proposed to characterise lens distortion [200]. While the generic radial and tangential lens distortion model of Brown [181] is sufficient for most lenses [161], a more comprehensive lens distortion model can help to improve precision [201]. To increase the accuracy of camera calibration, both Vo et al. [195][196] and Douxchamps et al. [197] included prism distortion coefficients in their lens distortion model. The main point of difference was that Douxchamps et al. [197] used a higher order radial distortion model compared to Vo et al. [195][196]. The models used by Vo et al. [195][196] and Douxchamps et al. [197] are inverse models that map distortion-free locations to distorted locations. Using an alternative approach, Rahman et al. [201] used a forward lens model to map distorted locations to undistorted locations, and were able to improve precision at the cost of model complexity.

To avoid solving an inverse problem, and to increase precision, a forward lens distortion model was implemented as part of this chapter. Zernike polynomials [202] were used as the mapping functions as they are orthogonal over the continuous unit circle, and they can readily capture and model different aspects of the data shape [203] (further advantages of Zernike polynomials are discussed in [204]). Because of the latter advantage, Zernike polynomials have
been used as shape-descriptors in some computer vision applications, such as [205], [206]. Rahbar et al. [207] used Zernike polynomials for blind correction of lens aberration, and showed that they can achieve a smaller error than when using a Taylor series expansion in modelling lens aberration. In the method described in this chapter, after the lens distortion effects were characterised, two separate sets of Zernike polynomials were used to parameterise the function that maps the distorted \( x \) and \( y \) locations of the points to their undistorted locations.

Even though localising the control points of the calibration template and characterising the lens distortion are important aspects of camera calibration, they alone are not sufficient for an accurate multi-camera calibration. Optimising the parameters of a multi-camera system is a challenging task in stereoscopic systems that have more than two cameras. An optimisation technique was proposed by HajiRassoulia et al. [38] to address this issue. However, this technique was based on a checkerboard calibration template for which the accuracy of localising the control points (i.e. the corners of the checkerboard) was limited. In the present work, a similar optimisation process as in [38] was used, but methods were implemented to further improve the localisation of the control points and the characterisation of the lens distortion.

The accuracy of the estimated camera parameters was tested experimentally and quantified by measuring known 3D translational shifts of a flat object using a stereoscopic system consisting of four cameras and a translational stage. The results are compared with two alternative camera calibration methods: the conventional method of calibrating multiple cameras using the method of Zhang et al. [158] and the Levenberg-Marquadt (LM) algorithm to minimise the sum of reprojection errors of a checkerboard template; and the multi-camera calibration method proposed by HajiRassoulia et al. [38].
Section 9.3 gives a brief description of the experimental setup and the calibration target used in this study. The method is described in Section 9.4. Results and discussion are provided in Section 9.5, and Section 9.6 presents the summary.

9.3 Experimental setup and calibration target

The stereoscopic system used for the experiments included four monochrome USB 3 cameras (Point Grey FL3-U3-13Y3M-C), equipped with 6 mm focal length lenses (DF6HA-1B from Fujinon, Japan) (Figure 9-1). The image resolution of these cameras was 1280 pixel × 1024 pixel. The overlapping field of view (FOV) of this stereoscopic system was approximately 175 mm × 175 mm at a distance of 200 mm from the centre of the overlapping FOV to the cameras (details are described in [79]).

![Figure 9-1: The stereoscopic system used for multi-camera calibration.](image)

The calibration template used for the stereoscopic system comprised groups of concentric circles positioned on a 3 × 4 grid (Figure 9-2). The distance between the centres of each group of circles was 30 mm. The calibration template size, the radius of the circles, and the distance between the concentric circles were selected based on the FOV and the average distance of the FOV to the cameras. The calibration template was designed using SolidWorks (Version 2015, USA), which was converted to a scalable vector graphics (SVG) image and printed using
A Model-based Technique for Calibration of Multiple Cameras

A laser printer (Ricoh MP C2503, 600 dots per inch). The printed template was carefully glued to a 3 mm thick acrylic sheet using an adhesive spray, thus providing a flat 2D template.

The images of the calibration template in the cameras of a multi-camera system are affine-transformed versions of each other (a situation where lens distortion effects are negligible). The four groups of concentric circles at the corners of the calibration template were used as markers to estimate the 3D location of the calibration template. The centre of the five concentric circles at the bottom right of the calibration template was selected as the first marker (Figure 9-2). The second, third, and fourth markers were chosen as the concentric circles in the top right, bottom left, and top left of the calibration template, respectively. The design of the markers for the calibration template enables them to be uniquely identified, even under large perspective distortions.

![Figure 9-2: The calibration template used in this study. The template was designed in SolidWorks and was converted to a scalable vector graphics (SVG) image.](image-url)
9.4 Calibration method

The calibration process comprised two main steps. In the first step, the initial values of the parameters of the multi-camera system were estimated using a checkerboard template and the calibration method of HajiRassouliha et al. [38]. In the second step, the model-based technique of this chapter was employed to refine the parameters and estimate the parameters of the lens distortion model. The new parameters were then used to recalibrate the cameras.

A checkerboard template was selected for estimating the initial values of the parameters at the first step because of its reliability in the presence of lens and perspective distortions [170]. The estimated initial values of the lens distortion coefficients allowed the correction of most of the lens distortion in the images. Removing the lens distortion paved the way to use concentric circle templates, which are typically more sensitive to lens distortions than checkerboard templates [170], but can provide higher accuracies for localising the control points in low-distortion images [194–196]. The concentric circle template described in Section 9.3 (Figure 9-2) was thus used in the second step to refine the parameters.

The details of estimating the initial values of the parameters of the multi-camera system are described in Section 9.4.1. The methods used for refining the parameters and recalibrating the cameras of the multi-camera system are described in Section 9.4.2. Section 9.4.3 describes the experiments performed to demonstrate the accuracy of the estimated calibration parameters for localisation of 3D points.

9.4.1 Estimating initial values of the parameters of the multi-camera system

A checkerboard calibration target was used to calibrate the cameras in order to estimate initial values of the parameters of the multi-camera system. Eighty images of the calibration target at various positions in the FOV were taken with each camera. The multi-camera technique proposed by HajiRassouliha et al. [38], which uses the radial and tangential lens distortion model, was used to calibrate the cameras.
9.4.2 Refining the parameters and recalibrating the cameras

After estimating the initial values of the parameters in Section 9.4.1, a new set of eighty images were taken with each camera of the concentric circle calibration target (Figure 9-2) in the FOV at various distances and positions with respect to the cameras. The lens distortion effects were removed from the calibration images using the initial estimates of the parameters. The centres of the concentric circles of the calibration template (control points) were then found in the undistorted calibration images and were mapped to the control points of the calibration template model (Section 9.4.2.1). The locations of the mapped control points were refined, and lens distortions were estimated (Section 9.4.2.2). A lens distortion model was defined and was used to reduce the lens distortion effects from the images (Section 9.4.2.3). Finally, the cameras were recalibrated based on the new estimated parameters (Section 0).

9.4.2.1 Finding and mapping the control points of the calibration template

The process of finding and mapping the control points of the calibration template in the undistorted calibration images involved the following steps:

1. The calibration template was segmented from the background. To achieve this, a Canny edge detection algorithm [208] was applied to the calibration images to convert the greyscale image into a binary format. In the binary image, all 8-connected components were selected and labelled. (In 8-connected components, pixels are connected to each other in their 8-neighbourhood. Refer to [209] for more details). Components with area less than 100 pixel², or greater than 1000 pixel², were considered to be part of the background and were removed from the images. These thresholds were selected based on the camera image size (i.e. 1280 pixel × 1024 pixel) and the circle sizes in the calibration images, to ensure that only the background was removed, without removing the circles even in images that had large perspective distortions.

2. The circles of the calibration template were found, based on the geometric characteristics of the components of the binary image. To perform this step, for
every component of the binary image, an ellipse was estimated having the same normalised second central moment as that component, and the ratio of the length of the major axis to the minor axis of that ellipse ($Ra$) was calculated (note that circles become elliptical under perspective distortion in the camera images). The smallest value of $Ra$ is 1 (a perfect circle), while large values of $Ra$ are associated with the components that are stretched in one direction. These typically correspond to remaining sections of the background in the binary images of the calibration template. The components of the binary image where $Ra$ was smaller than 2 were selected as the components that had geometric characteristics consistent with the circles of the calibration template. Even though the upper threshold for $Ra$ is dependent on the perspective distortion in calibration images, the selected value of $Ra$ was found to be valid for a wide range of calibration images.

3. The centres of the circles of the calibration template were found, and were selected as control points for the calibration template. A least-squares method was used to find the best-fit ellipse for the data points of each component of the binary image. The centres of the fitted ellipses were used as estimates of the centres of the circles of the calibration template. Note that perspective distortions and distances between the calibration target and the cameras cause discrepancies between the measured centres of the ellipses in the images and the perspective transformed centre of the circles of the calibration target. However, the estimated centres of the ellipses were only used to find an estimate of the calibration target position.

Even though the concentric circles have a common centre, because of the errors in identifying the centre of each circle, several closely placed centres were identified for the members of each group of concentric circles. A $k$-means clustering method was thus used to divide the identified centre positions into a number of clusters equal to the number of control points, and the average values of the $x$ and $y$ positions of the centres in each cluster were selected as the centre of that group of concentric circles and used as a control point of the calibration template.
4. The estimated control points were mapped to their corresponding template positions using the four defined markers of the calibration template in order to estimate the position and orientation of the calibration template. The four markers were identified in the calibration images based on the number of circles in each group of concentric circles (marker 1 had five, marker 2 had four, and marker 3 and marker 4 each had three concentric circles). The number of circles was found after clustering the centres using the $k$-means clustering in the previous step. Marker 3 and marker 4, which had the same number of circles, were distinguished based on the Euclidean distances of their centres to the centre of marker 1 in the calibration images.

After identifying and mapping the control points, a model-based technique, described in the next section, was used to refine the locations of the control points, and to estimate the lens distortion effects.

9.4.2.2 Refining the locations of control points and estimating the lens distortion effects

The process of localising the control points of the calibration template involves two main sources of error. The first source is the error of the algorithm used in localising the control points, and the second is the effects of lens distortion. While these two sources of error are not easily separable, the following model-based technique was used to estimate the total error:

- A model of the calibration template was generated using its SVG image (Figure 9-2). The distances between the control points were converted from physical units of length to pixels based on the dimensions specified in the SolidWorks model.

- The control points of the calibration template model were identified as described in Section 9.4.2.1.
• The projective transformation between the 12 control points of the calibration template model and the 12 control points of the calibration target in the undistorted calibration images was found using a linear least-squares method. The projective transformations between the calibration model and the calibration target images could be estimated more accurately from the undistorted images than from the distorted images.

• The synthetic calibration template model was reconstructed in the estimated location of the calibration target using the projective transformation. The locations of the control points of the calibration template in the reconstructed model were all known from the specifications of the calibration target model.

• The 2D local discrepancies (subpixel shifts in the $x$ and $y$ directions of the image coordinate system measured in pixels) between the control points of the reconstructed model and the control point locations calculated from the calibration images were estimated using the P-SG-GC subpixel image registration algorithm [35] (denoted $D_x$ and $D_y$, respectively). $D_x$ and $D_y$ were estimated for 128 pixel × 128 pixel subimages centred at the $(x, y)$ coordinates of the control points of the calibration template in both the reconstructed model and the calibration images. The selection of the subimage size depends on the camera image characteristics, such as resolution, texture, and noise. The 128 pixel × 128 pixel subimage for the setup provided a good trade-off between the locality of displacement measurements and the need to ensure an adequate number of image features in the subimages. Image features are useful for performing subpixel image registration, and the concentric circles of the calibration template provided suitable features for this purpose.

The discrepancies, $D_x$ and $D_y$, were due to errors in localising the control points and by lens distortion effects. However, the selection of the control points of the reconstructed model instead of the initial estimates helped to reduce the errors of localising the control points. Thus, it was expected that the errors associated with localising the control points would be
smaller and more randomly distributed compared to the errors associated with lens distortion. It was therefore assumed that most of the measured $D_x$ and $D_y$ values were due to the remaining effects of lens distortion. Based on this assumption, the control points of the reconstructed calibration model at each image were selected as the refined control points of the calibration target in that image, and $D_x$ and $D_y$ values were used to characterise the lens distortion effects.

9.4.2.3 The lens distortion model

Equation 8.5 defines the conventional radial and tangential distortion model, where $(x_d, y_d)$ are the distorted locations in the images, $(x_u, y_u)$ are the undistorted locations, $k_1$ are the radial distortion coefficients, $p_1$ and $p_2$ are the tangential distortion coefficients, and $r$ is the distance of the undistorted locations from the principal point ($Pr$). The radial distortion model of Equation 8.5 is symmetric about the centre, and in the form of a Taylor series expansion around $Pr$.

$$rd = \sqrt{(x_u - Pr_x) + (y_u - Pr_y)}$$

$$x_d = x_u(1 + k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6) + [2p_1 x_u y_u + p_2 (r_d^2 + 2x_u^2)]$$

$$y_d = y_u(1 + k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6) + [p_1 (r^2 + 2y_u^2) + 2p_2 x_u y_u]$$

Even though it is relatively straightforward to estimate the traditional radial and tangential distortion model using a checkerboard template, the lens distortion model is an inverse model. Furthermore, as Rahbar et al. [207] showed, the Taylor series used in this model limits the accuracy to which lens distortion can be represented. To address this limitation, Zernike polynomials were used in a direct lens distortion model (the details and equations of the Zernike polynomials are described in [204]).
The measured $D_x$ and $D_y$ values from Section 9.4.2.2 provided data about lens distortion at the locations of the control points of the calibration target in all of the calibration images of each camera. To characterise this behaviour for each camera, independent Zernike polynomials were fitted to each of the measured $D_x$ and $D_y$ values in each of the cameras. The $x$ and $y$ inputs of the Zernike polynomials were the $x$ and $y$ coordinates of the control points in the calibration images, and the Zernike polynomials values were fitted to the measured $D_x$ and $D_y$ values at that location. To take advantage of the orthogonality of Zernike polynomials within the unit circle [203], the $x$ and $y$ positions of the control points were normalised within the unit circle prior to being used as the inputs of the Zernike polynomials. By fitting Zernike polynomials to the $D_x$ and $D_y$ values, the polynomials became mapping functions that estimate the shift that is caused by the lens distortion effects at the $x$ and $y$ coordinates of the points in the distorted images. The undistorted locations of the image points thus could be quantified by subtracting the estimated shifts of the Zernike polynomials from their distorted locations. The reason that Zernike polynomials were used to map the shift between distorted and undistorted images, rather than to map the distorted locations to undistorted locations, is their suitability for fitting to symmetric shapes that are similar to lens distortions or lens aberrations [207][210].

Zernike polynomials are only orthogonal over the continuous unit circle [204], whereas the $D_x$ and $D_y$ data characterising the lens distortion are discrete. However, the Gram-Schmidt orthogonalisation technique [210] allows the expansion of discrete data in terms of the Zernike polynomials while retaining orthogonality (details can be found in [204] and [210]). The Gram-Schmidt technique was therefore used to characterise the discrete data of the lens distortion effects using Zernike polynomials.

The order of the Zernike polynomials was chosen based on the values of $D_x$ and $D_y$. The complete set of third order Zernike polynomials (Table 9-1) was chosen to account for the complexity while avoiding over-fitting of the data in this application. However, the order of Zernike polynomials used to characterise the lens distortion could be determined quantitatively for each specific lens using information theory methods, such as Akaike
A Model-based Technique for Calibration of Multiple Cameras

information criterion [211], Bayesian information criterion [212], or Mallow’s $C_p$ criterion [213]. Since the terms in Zernike polynomials are orthogonal, higher order basis functions could be added without affecting the coefficients of the lower order terms. Moreover, the shapes of low-order Zernike polynomials are related to dominant lens aberrations, such as tilt, astigmatism, coma, and defocus [207, 214]. The shape properties of Zernike polynomials were helpful in reducing the random error caused by mislocalisation of the control points in the calibration images.

Table 9-1: The third order Zernike polynomials used to model the lens distortions. $\rho$ is the radial distance and $\theta$ is the azimuthal angle.

<table>
<thead>
<tr>
<th>Order (n)</th>
<th>Frequency (m)</th>
<th>$Z_n^m (\rho, \theta)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>$2 \rho \sin(\theta)$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$2 \rho \cos(\theta)$</td>
</tr>
<tr>
<td>2</td>
<td>-2</td>
<td>$\sqrt{6} \rho^2 \sin(2\theta)$</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>$\sqrt{3}(2 \rho^2 - 1)$</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>$\sqrt{6} \rho^2 \cos(2\theta)$</td>
</tr>
<tr>
<td>3</td>
<td>-3</td>
<td>$\sqrt{8} \rho^3 \sin(3\theta)$</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>$\sqrt{8}(3 \rho^3 - 2\rho)\sin(\theta)$</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>$\sqrt{8}(3 \rho^3 - 2\rho)\cos(\theta)$</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>$\sqrt{8} \rho^3 \cos(3\theta)$</td>
</tr>
</tbody>
</table>

9.4.2.4 Recalibrating the cameras

The calibration process was repeated to refine the parameters of the multi-camera system using the refined control points of the concentric calibration target obtained from the calibration images. The radial and tangential distortion coefficients were included in the process of recalibrating the cameras (using initial values of zero) to test if the Zernike polynomials could
Effectively remove radial and tangential distortions. The four cameras of the stereoscopic system were recalibrated using the multi-camera calibration technique proposed by HajiRassouliha et al. [38]. The initial values of the intrinsic and extrinsic parameters of the cameras were the values estimated in Section 9.4.1.

9.4.3 Experiments performed to demonstrate the accuracy of the estimated parameters

To test the accuracy of the estimated parameters, the multi-camera system was used to estimate a physical shift applied to a flat object in the FOV of the cameras (Figure 9-1). The flat object was attached to a micrometer-driven manual linear stage with an accuracy of 2 µm over a 25 mm travel range (M-423 [104], Newport, U.S.A.). One image was taken from the initial position of the flat object in all of the four cameras. The flat object was then translated by approximately 10 µm, 50 µm, 200 µm, 500 µm, 1000 µm, 2000 µm, and 3000 µm. Images were taken by all of the four cameras at the end of each translational shift.

The first step in estimating 3D shifts is to match corresponding points in the cameras to 3D reconstruct the surface of the flat object. Matching corresponding points in arbitrarily positioned cameras (such as the multi-camera system of this study (Figure 9-1)) is a more challenging task than in two closely aligned cameras since image views can have substantial differences in arbitrarily positioned cameras. As a result, conventional methods of block-matching, which typically use cross-correlation to match the corresponding points in cameras, are not suitable. HajiRassouliha et al. in [152] proposed a method to address this issue. In this approach, the projective transformation between the images of the cameras was determined using at least 10 manually selected control points. One camera was used as the reference camera, and the projective transformation was applied to the images of the remaining cameras to obtain views similar to the reference camera. The resulting projective transformation was used only as an initial estimate, so it did not need to be particularly accurate. In [152], the corresponding points were found by extracting and matching image features. Since the images from the cameras were similar after applying the projective transformation, the accuracy of this method was improved by using block-matching methods and subpixel image registration to find the corresponding points from the transformed images. The choice of reference
camera did not significantly affect the result of this experiment (Camera 1 was chosen arbitrarily as the reference camera). A (24 × 39) virtual grid of points with a step size of 10 pixels (i.e. 936 points) was selected on the surface of the flat object in the reference camera. The large number of points provided good estimates of the variations of the 3D displacement measurements over the surface of the flat object. The P-SG-GC subpixel image registration algorithm [35] with subimages of size 128 pixel × 128 pixel was then used to match the corresponding points between the image of the reference camera and the transformed images of the non-reference cameras. The selection of subimage size was based on the flatness of the test object and perspective distortions in the camera images. The matched corresponding points were transformed back to the original coordinate system of the non-reference cameras using the inverse of the projective transformation.

The 2D shifts between the initial and shifted states of the flat object were found at the positions of the matched corresponding points in the images from all the cameras using the P-SG-GC subpixel image registration algorithm [35]. To test the effect of the subimage size on the accuracy of the measurements, two subimage sizes were tested to find the 2D shifts between the initial and shifted states of the flat object. The subimage size was selected as 32 pixel × 32 pixel and 64 pixel × 64 pixel for all the 2D measurements in the first and second set of measurements, respectively. The subimages were relatively small in order to limit the measurements to localised shifts.

The matched corresponding points and their estimated shifted locations were first mapped to their undistorted positions using the method described in Section 9.4.2.3. The undistorted points were then triangulated using the intrinsic and extrinsic parameters of the cameras estimated as in Section 0. This resulted in 8 sets of 3D points (i.e. one set for the initial state of the flat object and 7 sets for the 7 applied shifts using the translational linear stage). The 3D shifts were estimated by measuring the 3D Euclidean distances between the points of the initial state and points of the shifted state. The measured 3D distances were averaged for each set of 3D points (936 points in total), and this average was used as the final estimate of the
algorithm for that applied shift. The absolute error for each algorithm was calculated by subtracting the averaged values from the actual physical shifts applied to the flat object using the linear translational stage. In addition, the standard deviation (STD) of the 3D Euclidean distances was used as an indication of the resolution of the measurements.

The results obtained from the estimated parameters using this method were compared to the parameters estimated using the conventional multi-camera calibration method based on the LM algorithm to minimise the sum of reprojection errors, and the multi-camera calibration method proposed by HajiRassouliha et al. [38]. The method of HajiRassouliha et al. [38] was used to estimate the initial values of the parameters in Section 9.4.1. The same eighty calibration images used to estimate the initial values in Section 9.4.1 were used to calibrate the cameras of the stereoscopic system using the earlier multi-camera calibration method. The procedures for estimating the 3D shifts and errors were the same as for the method proposed in this chapter, except for removing the lens distortion, so that in these methods the images were undistorted prior to triangulation using the lens distortion coefficients estimated using each method.

9.5 Results and discussion

9.5.1 Estimating the initial values of the parameters of the multi-camera system

The initial intrinsic and extrinsic parameters of the cameras were estimated using the multi-camera calibration method of HajiRassouliha et al. [38]. Table 9-2 shows the average RMS reprojection error obtained using 80 calibration images of a checkerboard template. In addition to the reprojection errors, the multi-camera calibration method of HajiRassouliha et al. [38] estimates two other error functions: “length error” and “3D error”. The RMS length error and RMS 3D error obtained for the experimental setup of this study and calibration images were 25 µm and 53 µm, respectively. The results of the method for refining the initial values of intrinsic and extrinsic parameters of the cameras are presented in the next section.
Table 9-2: The RMS reprojection error obtained using the multi-camera calibration method of HajiRassouliha et al. [38]

<table>
<thead>
<tr>
<th>Camera Number</th>
<th>The RMS reprojection error (pixel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera 1</td>
<td>0.091</td>
</tr>
<tr>
<td>Camera 2</td>
<td>0.109</td>
</tr>
<tr>
<td>Camera 3</td>
<td>0.095</td>
</tr>
<tr>
<td>Camera 4</td>
<td>0.100</td>
</tr>
</tbody>
</table>

9.5.2 Refining the parameters and recalibrating the cameras

The calibration template shown in Figure 9-2 was used to refine the initial estimates of the intrinsic and extrinsic parameters of the cameras. Figure 9-3 shows a sample concentric calibration image from one the cameras of the multi-camera system. As can be seen in Figure 9-3, the perspective distortion effect has caused the circles of the calibration template to appear as ellipses in the calibration images.

Figure 9-3: A sample image of the concentric calibration template recorded using one of the cameras of the multi-camera system.

9.5.2.1 Finding and mapping the control points of the calibration template

The Canny edge detection algorithm [208] was applied to the calibration images to extract the circles from the template. Figure 9-4(a) shows a sample output for which parts of the background are present in the image. The first step of the method described in Section 9.4.2.1 removed the background. The result is shown in Figure 9-4(b). Figure 9-4(a) shows an example of removing a large part that remained after applying the Canny edge detection algorithm,
Results and discussion

using the first step of the method described in Section 9.4.2.1. However, parts of the background remained in some calibration images, such as the example shown in Figure 9-5. The second step of the method described in Section 9.4.2.1 was used to remove remaining parts of the background. Figure 9-6 shows a sample result after removing all parts of the background from the calibration images. The concentric circles of Figure 9-6 were segmented from the images and used for fitting ellipses.

Figure 9-4: (a) An example of applying the Canny edge detection algorithm to a calibration image. (b) The background was removed using the first step of the method described in Section 9.4.2.1.

Figure 9-5: An example of using the second step of the method described in Section 9.4.2.1 to remove the remaining parts of the background.
Figure 9-6: A sample result of removing all parts of the background from calibration images. The concentric circles were segmented for use in the ellipse fitting process.

The least-squares method described in the third step of Section 9.4.2.1 was used to find the centres of the segmented concentric circles of the calibration images. Figure 9-7 shows three sample results. The centres of the ellipses are shown with red crosses in Figure 9-7. As can be seen, the red crosses that show the centres of the concentric circles are close to each other (in the test stereoscopic system, the cameras were poisoned approximately 200 mm from the calibration target (refer to [79] and Chapter 7 for details)). As described in the third step of Section 9.4.2.1, the $k$-mean clustering method was used to cluster the centres into a number of groups equal to the number of control points (12 in this case). The average values of the $x$ and $y$ coordinates of the centres in each cluster were used as the control points. Figure 9-8 shows a sample result of localising the control points of the calibration template.

Figure 9-7: Three samples of the ellipse fitting results. The least-squares method described in the third step of Section 9.4.2.1 was used to find the best-fit ellipses to the segmented concentric circles. The centres of the ellipses are shown with red crosses.
Results and discussion

The control points located in the calibration images were mapped to control points of the calibration template using the method described in the fourth step of Section 9.4.2.1. A sample result of mapped control points in one calibration image from all four cameras is shown in Figure 9-9. The lines in Figure 9-9 connect the control points of the calibration template in a consistent manner.

Despite the large perspective distortion, the diversity of the calibration images in the four cameras of the stereoscopic system, and the presence of background material in the images, the method described in Section 9.4.2.1 was able to map the centres in 307 of 320 calibration images in the four cameras (i.e. a 96% success rate). The failure of the algorithm in 13 calibration images was due to several factors, such as the specular reflection from the calibration target surface, or the inability of the algorithm to remove all of the background material.
9.5.2.2 Refining the locations of control points and estimating the lens distortion effects

The initial locations of the mapped control points were refined using a model-based technique. A model of the calibration template was generated in the 3D estimated location of the calibration template in the calibration images using steps one to four of the method described in Section 9.4.2.2. Figure 9-10 shows a sample calibration image and the generated model of the calibration template. Zoomed views of the concentric circles of the calibration template in the calibration image and the model are also provided in Figure 9-10 for comparison. As the zoomed views illustrate, the predicted view of the calibration template (i.e. the model of the calibration template) was similar to the actual calibration image. However, the reconstructed model of the calibration template and the calibration image show some minor local discrepancies due to the errors in localising the control points, and the lens distortion effects (as discussed in Section 9.4.2.2). The discrepancies between the calibration image and the reconstructed model of the calibration template \((D_x \text{ and } D_y)\) were measured in
subimages of size 128 pixel × 128 pixel using the P-SG-GC algorithm [35] (as described in the fifth part of the method in Section 9.4.2.2). Figure 9-11 shows two sample subimages from the calibration image and the model of the calibration template. Some parts of the concentric circles did not fit in the subimages of sample 2 in Figure 9-11. The pattern was thus cropped in both subimages. Because sufficient image features were present in both subimages, it did not affect the subpixel shift measurements.

The local discrepancy data (subpixel shifts) were measured in the subimages of all the calibration images of the four cameras of the stereoscopic system in the x and y directions. The discrepancy data were used to create the lens distortion model in the next section.

![Figure 9-10: A sample calibration image (top left) and the generated model of the calibration template using steps one to four of the method described in Section 9.4.2.2 (top right). Zoomed views of the concentric circles of the calibration template in the calibration image and the model are shown below each image.](image-url)
9.5.2.3 The lens distortion model

The values of $D_x$ and $D_y$ were used to model the lens distortion in the $x$ and $y$ directions, as described in 9.4.2.3. Two separate third order Zernike polynomials (Table 9-1) were fitted to the $D_x$ and $D_y$ values of each camera. The RMS fitting errors are shown in Table 9-3. The fitting errors did not considerably change by increasing the order of the Zernike polynomials (data not shown). A portion of the fitting error is due to the random errors in $D_x$ and $D_y$ data that are associated with localising the control points, which was reduced by fitting the $D_x$ and $D_y$ data to symmetric patterns of the Zernike polynomials. The fitted Zernike polynomials were then used as mapping functions to model the lens distortion by mapping from distorted locations to undistorted location in each camera.

Figure 9-11: Two samples of 128 pixel × 128 pixel subimages of the calibration image (left) and the reconstructed model of the calibration template (right). The subpixel shifts between the subimages were measured using the P-SG-GC algorithm [35].
Table 9-3: RMS error of fitting Zernike polynomials up to 3rd order (Table 9-1) to $D_x$ and $D_y$.

<table>
<thead>
<tr>
<th>Camera Number</th>
<th>RMS error of fitting the third order Zernike polynomials to: D_x (pixel)</th>
<th>D_y (pixel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera 1</td>
<td>0.20</td>
<td>0.10</td>
</tr>
<tr>
<td>Camera 2</td>
<td>0.14</td>
<td>0.06</td>
</tr>
<tr>
<td>Camera 3</td>
<td>0.13</td>
<td>0.04</td>
</tr>
<tr>
<td>Camera 4</td>
<td>0.13</td>
<td>0.02</td>
</tr>
</tbody>
</table>

9.5.2.4 Recalibrating the cameras

The refined locations of the control points in all of the calibration images (estimated in Section 9.5.2.2) were used to recalibrate the cameras using the multi-camera calibration method of HajiRassouliha et al. [38], as described in Section 9.4.2.2. The standard radial and tangential distortion coefficients (Equation 8.5) included in the calibration process were remained close to zero following the optimisation process. This illustrates that the radial and tangential lens distortion effects were effectively corrected using Zernike polynomials and methods described in Sections 9.4.2.2 and 9.4.2.3.

9.5.3 Experiments performed to demonstrate the accuracy of the estimated parameters

As discussed in [38], reprojection errors are not good indicators of the accuracy of the estimated parameters of the cameras. Therefore, the accuracy of the optimal camera parameters was estimated using the experiments described in Section 9.4.3. Figure 9-12 shows a sample image of the flat object attached to a manual linear stage in the FOV of one of the cameras of the stereoscopic system. The flat object was shifted using the linear stage by the amounts described in Section 9.4.3. Because of the location of the flat object in the FOV (Figure 9-1 and Figure 9-12), the shifts were out-of-plane in all four cameras. Figure 9-13 shows the points selected in the reference camera (green dots in the top left image) and the corresponding points in the other three cameras (red dots in the other images) found using
the method described in Section 9.4.3. The arrangement of the corresponding points in the non-reference cameras clearly shows the perspective distortion effects.

The 2D shifts between the initial image of the flat object and shifted images were estimated using the P-SG-GC algorithm [35], as described in Section 9.4.3. Table 9-4 shows the average magnitude of the measured 2D displacements in all four cameras for each of the shifts applied to the flat object (note that because of perspective distortions (Figure 9-12), the 2D displacements varied across the flat object). These displacements were calculated using 64 pixel × 64 pixel subimages. The average magnitudes of 2D displacements in Table 9-4 show that the applied shifts ranged from small subpixel shifts (average 0.06 pixel) to large shifts (average 18.21 pixel).

Figure 9-12. A sample image of the flat object attached to a linear stage in the FOV of one of the cameras of the stereoscopic system.
Figure 9-13: A virtual grid of points was selected on the surface of the flat object in the reference camera (green dots in the top left image). The grid size was 24 points × 39 points (963 points) with a step size of 10 pixels. The corresponding points were found in the other three cameras using the method described in Section 9.4.3 (red dots in the top right and two bottom images). The images were cropped and zoomed for visualisation purposes.

Table 9-4: The average magnitude of 2D displacements measured in all four cameras for each translational shift of the flat object. The values were estimated using a subimage size of 64 pixel × 64 pixel, and the translational stage had a position uncertainty of 2 µm.

<table>
<thead>
<tr>
<th>Approximate translational stage shift (µm)</th>
<th>10</th>
<th>50</th>
<th>200</th>
<th>500</th>
<th>1000</th>
<th>2000</th>
<th>3000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean magnitude of 2D displacements (pixel)</td>
<td>0.06</td>
<td>0.31</td>
<td>1.22</td>
<td>3.05</td>
<td>6.08</td>
<td>12.14</td>
<td>18.21</td>
</tr>
</tbody>
</table>

The corresponding matched points in the initial and shifted images of the flat object in the four cameras were triangulated to find the 3D surface and 3D shifts, as described in Section 9.4.3. For comparison, the cameras were calibrated using the conventional method of calibrating multiple cameras (details are described in Section 9.4.3). Table 9-5 shows the RMS reprojection error of the conventional method, measured with the cameras of the stereoscopic system. Comparison of Table 9-5 with Table 9-2 shows that the multi-camera calibration method of HajiRassouliha et al. [38] can achieve smaller reprojection errors using the same calibration images and same initial values for the parameters.
Table 9-5: The RMS reprojection error obtained using the conventional method of calibrating multiple cameras.

<table>
<thead>
<tr>
<th>Camera number</th>
<th>RMS reprojection error (pixel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera 1</td>
<td>1.35</td>
</tr>
<tr>
<td>Camera 2</td>
<td>2.14</td>
</tr>
<tr>
<td>Camera 3</td>
<td>3.85</td>
</tr>
<tr>
<td>Camera 4</td>
<td>2.40</td>
</tr>
</tbody>
</table>

The camera parameters obtained using this method, the method of HajiRassouliha et al. [38], and the conventional method were used to triangulate the points, and to estimate the 3D shifts. Figure 9-14 compares the errors of each algorithm in estimating the 3D shifts and the STD of the 3D measurements derived from 32 pixel × 32 pixel and 64 pixel × 64 pixel subimages. As illustrated in Figure 9-14 the measurement errors obtained using the parameters of the model-based method proposed in this chapter are smaller than the two existing methods for all of the applied shifts. In particular, use of the parameters obtained by the model-based method resulted in a significantly smaller error for shifts larger than 200 µm compared to the conventional method. The shifts larger than 200 µm were on average larger than 1 pixel (Table 9-4). Even though the parameters obtained from the multi-camera calibration method of HajiRassouliha et al. [38] resulted in a noticeably smaller error than the conventional method, the errors were larger than the proposed model-based method for shifts larger than 500 µm. The relative difference between the three methods was greatest for the 3000 µm shift (Figure 9-14(b)). Comparison of shifts ≤1000 µm in Figure 9-14(a) and Figure 9-14(b) also indicated that increasing the subimage size from 32 pixel × 32 pixel to 64 pixel × 64 pixel did not improve the accuracy for any of the three methods. This shows that the error associated with 2D measurements of subpixel shifts was negligible. Shifts larger than 1000 µm were not estimated using 32 pixel × 32 pixel subimages since some parts of the 2D shifts were larger than 16 pixels, which is the theoretical limit of block-matching methods for this subimage size.
Results and discussion

Figure 9-14: The measurement errors in estimating 3D shifts of the flat object using the proposed method, the method of HajiRassouliha et al. [38], and the conventional method, for 32 pixel × 32 pixel subimages (a) and 64 pixel × 64 pixel subimages (b). Error bars show the STD of the 3D measurements. Refer to Table 9-4 for the average magnitude of 2D displacements for each 3D shift.

The average measurement errors and the STDs of all three methods for subimages of size 32 pixel × 32 pixel (Figure 9-14 (a)) and 64 pixel × 64 pixel (Figure 9-14 (b)) are presented in Table 9-6. The average 3D measurement error of the model-based method was less than the other two methods for both subimage sizes, and within the 2 µm position uncertainty of the translational stage (Section 9.4.3). The STDs of the measurements were similar for the model-
based method and the method of HajiRassouliha et al. [38], which were both significantly smaller than those of the conventional method (Table 9-6).

Table 9-6: The mean error ± STD for all of the applied shifts in Figure 9-14(a) (subimage size of 32 pixel × 32 pixel) and Figure 9-14(b) (subimage size of 64 pixel × 64 pixel).

<table>
<thead>
<tr>
<th>Method</th>
<th>Subimage: 32 pixel × 32 pixel</th>
<th>Subimage: 64 pixel × 64 pixel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean ± STD (µm)</td>
<td>Mean ± STD (µm)</td>
</tr>
<tr>
<td>The conventional method</td>
<td>11.4 ± 2.6</td>
<td>29.4 ± 4.1</td>
</tr>
<tr>
<td>HajiRassouliha et al. [38]</td>
<td>2.8 ± 2.2</td>
<td>5.2 ± 2.2</td>
</tr>
<tr>
<td>The proposed method</td>
<td>1.2 ± 2.2</td>
<td>1.4 ± 2.2</td>
</tr>
</tbody>
</table>

9.6 Summary

Localising the control points in calibration images has an important impact on improving the accuracy of template-based methods for calibrating multi-camera systems. In this chapter, some of the limitations of the previous methods were addressed, and a novel model-based technique for calibration of multi-camera systems was proposed. This method leverages the advantages of both a checkerboard template and a concentric rings-based template. A custom circle-array calibration target provided sufficient image features for finding and refining the control points. Concentric circles proved more robust to changes in image illumination and camera noise in comparison to the traditional checkerboard templates.

A checkerboard template was used to find the initial values of the parameters of the stereoscopic system, as shown in Figure 9-1. Methods were implemented to find, map and refine the control points of the concentric circles template, achieving a 96% success rate. The $D_x$ and $D_y$ values were fitted to two separate third order Zernike polynomials to model the lens distortions in the $x$ and $y$ directions, respectively.

The cameras were recalibrated using the refined control points and the multi-camera calibration method of HajiRassouliha et al. [38]. The standard radial and tangential distortion coefficients were remained close to zero after the camera calibration optimisation process was
completed. This illustrated that Zernike polynomials were able to remove radial and tangential distortion effects from the images.

The accuracy of the estimated parameters was assessed using specifically designed experiments. The results obtained from the model-based method proposed in this chapter showed significantly smaller errors than the conventional method, and outperformed the method of HajiRassouliha et al. [38]. The average measurement errors of the proposed model-based method were 1.2 µm and 1.4 µm in 32 pixel × 32 pixel and 64 pixel × 64 pixel subimages, respectively. These errors were within the 2 µm position uncertainty of the translation stage used to apply the shifts.

The use of a reconstructed model of the calibration template in the model-based method helped to compensate for any imperfections of the calibration target, such as length errors or lack of surface flatness. Thus, it eliminates the need to use precise calibration targets, which are typically expensive or difficult to manufacture. Furthermore, the model was reconstructed in the estimated 3D position of the calibration target rather than in the fronto-parallel plane proposed by Vo et al. in [7] and [8]. This approach helped to estimate and correct the lens distortion effects. An accurate subpixel image registration technique [16] (described in Chapter 2) was used to estimate the lens distortion effects. The selection of an area around the control point, rather than a single control point as in checkerboard templates, helped to provide a better estimate of the lens distortion effects. In addition, the third order Zernike polynomials (Table 9-1) that were used as mapping functions to model the lens distortions have enough flexibility to model complicated distortion effects [215].

In summary, a model-based technique was proposed for calibrating multi-camera systems. The technique outperforms previously published methods. While the method was implemented using a calibration target consisting of concentric circles, in principle, it can be used with any calibration target for which a model of the template can be reconstructed.
Measurement of deformation is a fundamental step in the assessment of the mechanical properties of objects, or in the development of computational models that describe the mechanical behaviour of objects. Digital image correlation (DIC) and digital volume correlation (DVC) are the main techniques used to measure relative deformations. DIC and DVC techniques comprise several algorithms that together affect the precision and robustness of the measurements. Prior to this thesis, DIC and DVC techniques could not provide precise or robust measurements suitable for many tasks, including many applications in bioengineering. The aim of this thesis was to address the limitations of DIC and DVC techniques by increasing the overall precision and robustness of deformation measurements. This was achieved by developing a novel set of tools and algorithms that can be incorporated into DIC and DVC techniques. The developed methods were comprehensively evaluated and tested, demonstrating significant improvements over existing state-of-the-art algorithms. The high precision and robustness of the methods developed in this thesis enable the successful use of DIC and DVC techniques in many applications that were previously impossible. Even though the methods were developed specifically to address the limitations of DIC and DVC techniques, they could be used in any similar computer vision task that demands high precision or robustness.
Section 10.1 of provided a summary of main findings and contributions of the chapters of this thesis. Future work and future directions were described in Section 10.2. Finally, Section 10.3 listed the outputs of this thesis, including journal publications, conference papers and abstracts, awards, patents, research grants, and invited presentations.

10.1 Summary of the main findings and contributions

Chapter 2: Subpixel image registration

A 2D subpixel image registration algorithm was proposed, which could address some of the limitations of 2D deformation measurement in DIC techniques. The algorithm was evaluated, and results were compared to other state-of-the-art algorithms. The method performed considerably better than other algorithms, even in the presence of high levels of Gaussian and salt and pepper noise. Hence, this method can be used to measure 2D deformations in DIC techniques with a significantly higher precision and robustness compared to state-of-the-art algorithms.

Chapter 3: The Effect of Camera Settings on the Image Noise Level and Subpixel Deformation Measurements

A method was proposed for evaluating the performance of deformation measurement algorithms in the presence of camera noise. Various sources of camera noise were introduced, and their quantities estimated for the camera used in the experiment. The method can provide useful information for selecting appropriate camera settings for system requirements of each application. The selection of camera settings involves a trade-off between the image noise level and maximum possible camera frame rate. Among the camera parameters, the camera gain had the greatest effect on the image noise level.

The results suggested that the effect of camera noise on the measurement accuracy is dependent on the algorithm that is used for measuring deformations, and the use of low-noise cameras is particularly justifiable for accurate algorithms, such as the subpixel image registration method proposed in Chapter 2.
Conclusions and future work

Chapter 4: Subpixel Measurement of Living Skin Deformation Using Intrinsic Features

The subpixel image registration method proposed in Chapter 2 was first evaluated using a single camera and a test setup to apply physical shifts to a flat object using a translation stage. The algorithm was used to estimate shifts in images. The results illustrated a high degree of linearity between the physical and estimated shifts. Next, the performance of the algorithm for measuring skin deformations was tested where the algorithm was used to track deformations of skin in vivo using only intrinsic features. This indicated that, unlike existing state-of-the-art algorithms that require richly textured images, the subpixel image registration algorithm developed in Chapter 2 could track skin deformations without the need to apply additional features to the skin’s surface.

Chapter 5: Motion Correction Using Subpixel Image Registration

The capability of the subpixel image registration algorithm, described in Chapter 2, for correcting motion artefacts was tested using a series of experiments with a camera, a flat object, a translation stage, and a rotation stage. The results illustrated that the algorithm could precisely and reliably correct motion for a range of applied translational and rotational motions applied to the flat object, and scaling of the object in camera images. The results indicated that the algorithm may be used as a reliable method for correcting for the relative motion between a camera system and a target object. In particular, motion correction is a necessary step for accurate measurement of skin deformations in vivo where the recordings are not stabilised due to the movements of the camera system, movements of the person, or breathing.

Chapter 6: An Accurate and Fast Algorithm for Subunit Registration of Arbitrary Dimensional Data

An accurate and fast subunit registration algorithm for arbitrary dimensional data was proposed. The method was evaluated using a wide range of test images/volumes, and results
were compared to state-of-the-art algorithms. The method addressed some of the limitations of volume deformation measurement in DVC techniques. For example, the algorithm could measure large deformations even in low-textured images and volumes where conventional methods fail. Moreover, the method had significantly higher precision and fewer computations compared to conventional methods. The method could thus be used in applications in which conventional DVC techniques were incapable of measuring deformations, such as analysis of optical coherence tomography (OCT) images.

Chapter 7: A Low-cost, Hand-held Stereoscopic Device for Measuring Deformations of Skin In Vivo

A hand-held device was designed to measure skin deformation in vivo. The design had sufficient mechanical strength to be mechanically stable during the experiments. A program was also developed and tested to simultaneously capture images from the four cameras of the system at 150 frame/s. The camera parameters were selected based on the required specifications of the system. The specifications of this hand-held device can be used to design similar devices for measuring deformations or geometries of soft tissues in vivo.

Chapter 8: Robust and Accurate Multiple Cameras Stereographic Calibration

A multi-camera calibration method was proposed to increase the accuracy of camera calibration, and to address the limitations of existing methods in simultaneous calibration of multiple cameras. The method was tested using a large number of calibration images captured using the cameras of the stereoscopic system designed in Chapter 7. The method demonstrated significant lower calibration errors and a faster convergence rate compared to traditional methods. The method can thus be used in computer vision applications that require the use of multiple cameras, such as capturing surface deformations using more than two cameras in 3D-DIC techniques.

Chapter 9: A Model-based Technique for Calibration of Multiple Cameras

A novel model-based technique for calibration of multi-camera systems was presented to improve the template-based methods for calibrating multi-camera systems. This method
Conclusions and future work

leverages the advantages of both a checkerboard template and a concentric rings-based template. The method addressed some of the limitations of the previous camera calibration methods in localising control points from calibration images and dealing with imperfect calibration targets. Furthermore, a method was proposed to estimate and correct lens distortions. The accuracy of the estimated camera parameters using this technique was assessed using specifically designed experiments. The results obtained from this method showed significantly smaller errors than the conventional method of calibrating cameras using checkerboard templates.

10.2 Future work

Chapter 2: Subpixel image registration

The method presented in this chapter was specifically designed to find translational shifts with subpixel precision. Since the images are divided into small subimages in DIC techniques, the deformations could be approximated with translational shift components. However, extending this method to non-rigid registration would improve the performance of the algorithm for applications in which the subimages have undergone non-rigid deformations.

This method is built upon an algorithm that is readily parallelisable. A hardware implementation of this method using field-programmable gate arrays (FPGAs), or graphics processing units (GPUs), would possibly result in a near-real-time performance, which is an advantage for many applications. Possible hardware accelerator choices are comprehensively discussed and compared in Appendix A. Furthermore, to illustrate the possible achievable speed-up ratios, an FPGA implementation of cross-correlation is presented and compared to its CPU implementation in Appendix B.
Chapter 3: The Effect of Camera Settings on the Image Noise Level and Subpixel Deformation Measurements

The noise measurements performed in this chapter can provide an estimation of the precision of deformation measurements from the images captured using the specific camera used in this study. The optimal camera parameters can then be selected to minimise measurement errors for the requirements of each application. However, the measurements could also be used to define the parameters of an appropriate camera noise model, which would allow analytical determination of the optimal camera settings of each application.

Chapter 4: Subpixel Measurement of Living Skin Deformation Using Intrinsic Features

The subpixel image registration method of Chapter 2 was able to provide measurements of skin deformation in vivo using only intrinsic features. The primary application of measuring skin deformations in vivo is to assess mechanical properties that may be used to inform biomechanical models. However, this technique could also be used to measure subtle deformations of skin cause by pulsation of blood in near surface veins or arteries. One application of such measurements would be to estimate jugular venous waves or carotid arterial pulse contours to noninvasively diagnose and monitor heart failures.

Chapter 5: Motion Correction Using Subpixel Image Registration

The proposed motion correction method used a fixed number of control points on the images to register subimages of the shifted images to the initial image. However, the number of control points could be selected depending on the nature of the motion observed in the images, thereby minimising computation time. Furthermore, some other comparison metrics, such as peak signal to noise ratio (PSNR) and the structural similarity (SSIM) index would give extra information about the quality of registration of the shifted image to the initial image.
Conclusions and future work

Chapter 6: An Accurate and Fast Algorithm for Subunit Registration of Arbitrary Dimensional Data

The proposed registration algorithm is a flexible method where the filters can be chosen based on the specifications of the N-D data. Further research is needed to fully exploit the flexibility of this method, and investigate the new capabilities it enables. One possibility is to adaptively choose filters based on the subset subimage data. Another improvement would be to use a variable subset size depending on the type of deformation. Similar to the registration method proposed in Chapter 2, this registration technique could take advantage of parallel processing. Considering the low complexity and low computational costs of this method, parallel implementation of this method could significantly reduce the long processing times typical of DVC techniques.

Chapter 7: A Low-cost, Hand-held Stereoscopic Device for Measuring Deformations of Skin In Vivo

The hand-held device of this chapter was designed to have sufficient mechanical strength. However, computer modelling of the design, including analysis of the forces applied by the equipment masses, would provide a quantitative method of predicting the mechanical deformations. A computer model could also help to optimise the mechanical design.

Chapter 8: Robust and Accurate Multiple Cameras Stereographic Calibration

The multi-camera calibration method was shown to be capable of calibrating four cameras using a large data set of calibration images. The stereoscopic system used for the evaluations had a symmetric design with four cameras of the same type. However, the advantages of this method could be illustrated more clearly for non-symmetric stereoscopic systems with more than four cameras.
Chapter 9: A Model-based Technique for Calibration of Multiple Cameras

The model-based multi-camera calibration method was tested with a concentric ring template. However, in principle, the method can be used with any calibration target for which a model of the template can be reconstructed. Further research is required to investigate the effects of choosing a variety of calibration targets for use with this method.

10.3 Thesis outputs

The following publications, awards, patents, research grants, and presentations were produced or facilitated during the course of this thesis.

10.3.1 Journal publications


6. **HajiRassouliha, A.**, Taberner, A. J., Nash, M. P., Nielsen, P. M. F., “Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms”, in review for the Image and Vision Computing journal.
Conclusions and future work


10.3.2 Conference papers and abstracts


Methods in Biomechanics and Biomedical Engineering (CMBBE), Eindhoven (2014).


10.3.3 Awards

1. 2014: Best student oral presentation at the Auckland Bioengineering Institute (ABI) research forum.

2. 2015: Accepted as one of the three postgraduate students from the engineering department for the Engineers in Clinical Residence (ECR) program. The purpose of this program was to develop innovative clinically-oriented technologies.

3. 2017: Best student poster presentation runner-up at the ABI research forum.

10.3.4 Patents


10.3.5 Research grants

1. Principal investigator: 5,000 NZD seed fund to hire a student for a summer project to assess the biomechanics of cornea using subpixel image registration and artificial neural network algorithms (2015).


4. Associate investigator: 20,000 NZD seed fund from New Zealand Artificial Limb Service (NZALS) for a project titled: "Mapping stump stiffness" (2017).
10.3.6 Invited presentations

1. The MedTech Centre of Research Excellence (CoRE) launch exhibition (October 2015)

2. Technology Innovation and Knowledge Interchange (TIKI) tour at the North Shore Hospital (2016).

Appendix A: Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms

This chapter is based largely on the following journal paper, which is in review in Image and Vision Computing journal:

HajiRassouliha, A., Taberner, A.J., Nash, M.P., Nielsen, P.M.F.: Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms

A.1 Abstract

Computer vision and image processing algorithms form essential components of many industrial, medical, commercial, and research-related applications. Modern imaging systems provide high resolution images at high frame rates, and are often required to perform complex computations to process image data. However, in many applications rapid processing is required, or it is important to minimise delays for analysis results. In these applications, central processing units (CPUs) are inadequate, as they cannot perform the calculations with sufficient speed. To reduce the computation time, algorithms can be implemented in hardware accelerators such as digital signal processors (DSPs), field-programmable gate arrays (FPGAs), and graphics processing units (GPUs). However, the selection of a suitable hardware accelerator for a specific application is challenging. Numerous families of DSPs, FPGAs, and
GPUs are available, and the technical differences between various hardware accelerators make comparisons difficult. It is also important to know what speed can be achieved using a specific hardware accelerator for a particular algorithm, as the choice of hardware accelerator may depend on both the algorithm and the application. The technical details of hardware accelerators and their performance have been discussed in previous publications. However, there are limitations in many of these presentations, including: inadequate technical details to enable selection of a suitable hardware accelerator; comparisons of hardware accelerators at two different technological levels; and discussion of old technologies. To address these issues, in this review, important considerations when selecting suitable hardware accelerators for computer vision and image processing tasks were introduced and discussed, and a comprehensive review of hardware accelerators was presented. Moreover, the practical details of chip architectures, available tools and utilities, development time, and the relative advantages and disadvantages of using DSPs, FPGAs, and GPUs were discussed. This review aims to provide sufficient practical information about state-of-the-art DSPs, FPGAs, and GPUs to enable developers to make a comprehensive comparison between various hardware accelerators, and to select a hardware accelerator that is most suitable for their specific application.

A.2 Introduction

Computer vision and image processing algorithms are used in a variety of applications in experimental mechanics [1], medical technologies [118], and human action recognition [216]. Many of the algorithms that have been used in these applications are computationally demanding, and in practical applications it is necessary to rapidly analyse the data. One of the main techniques for decreasing computation time is to use hardware with high computational power. Although the processing power of the central processing units (CPUs) in personal computers (PCs) is increasing, it remains insufficient for many applications. In addition, PCs cannot be used for computer vision tasks in mobile or portable devices. Hardware accelerators (e.g. digital signal processors (DSPs), field programmable gate arrays (FPGAs), and graphics processing units (GPUs)) are designed to address the increasing need for performing fast
calculations in complicated algorithms. Furthermore, some hardware accelerators can be used in portable systems where it is not feasible to use PC-based systems.

Although DSPs, FPGAs, and GPUs have markedly different chip architectures, requiring different software development techniques, each can be used as a hardware accelerator to speed up computations. Microarchitecture and fabrication technologies are rapidly evolving, and commercial competition has motivated major hardware accelerator vendors to update and increase the capabilities of their products using the latest technological advances. However, different hardware accelerators are designed in ways that make them efficient for some algorithms but not others. Furthermore, the choice of a hardware accelerator is typically a trade-off between computational power, speed, development time, power consumption, and price. Identifying a suitable hardware accelerator for a specific algorithm or application can be thus very challenging.

Previously published reviews have investigated different aspects of using hardware accelerators in computer vision and image processing tasks. These review papers can be divided into four main groups, which are discussed here.

In the first group of review papers, a specific algorithm or application is chosen and various hardware accelerators for that task are compared. An example is stereo vision algorithms for real-time systems, as in [217]. These review papers may help with the choice of a suitable hardware accelerator for specific applications. However, the system requirements can vary considerably for other applications or algorithms. For example, in some applications real-time execution is important (see [217]), while for other applications it may be adequate to simply increase the processing speed. The choice of a suitable hardware accelerator depends significantly on the application and the algorithm.

In the second group of reviews, specific hardware accelerators are chosen to test the performance of algorithms and their implementation. For instance, algorithm implementations for a single FPGA and a single GPU for sliding-window applications are
discussed in [218]. In these hardware-oriented reviews, the fact that new technologies have many advantages over their older versions, was not considered, which does not help developers to find suitable modern hardware accelerators for their own applications. Furthermore, a specific FPGA or a specific GPU does not necessarily represent the capability of that type of hardware accelerator in general. Therefore, these review papers may not help researchers to obtain an accurate comparison between hardware accelerators, unless they decide to choose a hardware accelerator specifically from those that have been reviewed.

In the third group of reviews, a broader application is chosen and different hardware accelerators are discussed for that purpose. Some examples are: parallel computing with multicore CPUs, FPGAs, and GPUs in experimental mechanics [219]; medical image processing on GPUs [220], [221]; and medical image registration on GPUs [222] or multicore CPUs and GPUs [223]. There are also some technical details about the chip architectures in these papers. Even though these papers can provide useful information, some of them (such as [220], [221], [222], [223]) only discuss GPUs and do not cover FPGAs or DSPs. In addition, the hardware details are usually limited to a specific hardware and are of limited use for comparing different hardware accelerators.

In the fourth group of reviews (such as [224]), the chip architecture and software tools of hardware accelerators are discussed in detail. An example is heterogeneous computing (i.e. the combination of CPUs with FPGAs or GPUs) for general applications. Even though such reviews provide useful information, there is a need to update and simplify the technical details to provide practical advice for researchers on the choice of suitable hardware accelerators for computer vision and image processing applications.

This review combines the approach of the third and fourth groups of review papers described above. The aim was to provide enough useful and practical information to allow researchers to choose the most suitable hardware accelerator for computer vision and image processing applications. For this purpose, DSPs, FPGAs, and GPUs are discussed in separate sections, with practical details about selection of suitable hardware accelerators for different computer vision and image processing applications.
One of the main challenges in reviewing different hardware accelerators is to provide a fair comparison. Since the model names of DSPs, FPGAs, and GPUs are not indicative of their performance, a ‘speed normalisation’ factor was proposed [217] in an effort to improve the usefulness of comparisons in the same chip architecture family. However, hardware accelerators are too complicated to limit the performance comparison only to the processing speed, which cannot indicate the advantage of one hardware accelerator over another, especially when they do not belong to the same family. Moreover, the processing speed of an algorithm is not only dependent on the hardware accelerator, but also on the programmer’s skill. In order to provide a practical comparison between hardware accelerators in this review, the most important features of DSPs, FPGAs, and GPUs for computer vision and image processing algorithms are introduced and discussed. Then, based on the technical specifications, hardware accelerators are divided into groups with similar levels of performance.

Another limitation of some review papers (such as [219]) is the discussion of outdated hardware technologies, which offer little help in assessing the performance and capabilities of modern hardware accelerators. This review addresses this issue by reporting on the latest improvements, and covers recent papers (published since 2009) with a focus on the latest hardware technologies.

This review is organised as follows. DSPs, FPGAs, and GPUs are discussed in Sections A.3, A.4, and A.5, respectively. In each section, and for each hardware accelerator, different families, available development tools and utilities, development time, and the advantages and disadvantages of using the type of hardware accelerator are discussed. Each section concludes with a separate literature review and summary, and each literature review section presents separate tables with a summary of the application, algorithms being implemented, hardware type used, and performance (or data throughput) of the algorithm. In addition, the papers being reviewed are sorted chronologically and the year of introduction of FPGAs and GPUs (as an indicator of their hardware technology level) is reported. Since FPGAs and GPUs have
both been widely used in computer vision and image processing tasks, Section A.6 is devoted to the comparison of GPUs and FPGAs. Finally, Section A.7 summarises this review.

A.3 Digital signal processors (DSPs)

DSPs are microprocessors with an architecture that is specifically designed for performing signal processing tasks. Texas Instruments (TI) and Analog Devices (AD) are the two major companies in the DSP production market. TI-DSPs are more common in the computer vision and image processing research community than AD-DSPs, so this review focuses on TI-DSPs.

TI has designed various DSPs with different processing power ranges and capabilities for different purposes. TI-DSPs can be divided into 4 major groups [225]:

- **Ultra-low power DSPs** (C5000, and C55x) [226];
- **Power optimised DSPs** (C6000, C64x, C671x, C672x, C674x, and Open Multimedia Applications Platforms (OMAPs) [227];
- **DaVinci digital media processors** (DaVinci-DMPs) (DM64x, DM37x, DM64x, and DM81x) [228];
- **Multicore DSPs** (C66x, C667x, C665x, and C647x) [229][230].

The ultra-low power DSPs are cost-effective, but have low computational power, so are limited to simple computer vision and image processing algorithms.

Power optimised DSPs are mainly designed for portable or mobile devices where the power consumption is the most important feature. These DSPs can process algorithms with a moderate level of complexity. The main difference between the OMAP series and other members of this family is the addition of an ARM processor, which makes them a system on a chip (SOC). This ARM processor can handle interfacing with standard ports (e.g. USB and I²C), external memory modules (e.g. secure digital (SD), and multi-media cards (MMC)). As
an example of their application, OMAPs are used in some mobile phones to handle digital camera, screen, and external memory interfaces.

DaVinci-DMPs are designed for multimedia applications such as video and image processing, and video capture. These DMPs include video and image hardware codecs (e.g. MPEG, H.264, and JPEG) and hardware accelerators for video processing. Apart from a few DMPs in the DM64x series, the DaVinci-DMPs are a more advanced SOC (i.e. DSP + ARM) version of OMAPs. In comparison to power optimised DSPs and OMAPs, DaVinci-DMPs can perform more complex tasks at the expense of higher power consumption. DaVinci-DMPs also include peripherals needed for camera interfacing; hence video frame grabbers are a typical example of an application for these DSPs.

Multicore DSPs are optimised for computationally complex tasks and high performance computing (HPC). They consist of multiple DSP cores (up to 8 cores), which enable them to perform tasks in parallel. In this sense, the parallelism in multicore DSPs is similar to multicore CPUs in PCs. Multicore DSPs are designed based on two main architectures: Keystone architecture1 and Keystone architecture2 [231]. The principal difference between these two architectures is the addition of an ARM processor in the Keystone architecture2 to divide tasks between the DSP and the ARM processor. It should be considered that the reported maximum computational performance of multicore DSPs is based on the assumption that the task can be fully parallelised so that threads are executed in different cores simultaneously. However, this is often not feasible in practice, and advanced parallel programming techniques are required to optimise the computation speed of these DSPs.

The type of arithmetic computation support (i.e. fixed-point or floating-point operation) is another factor which needs to be considered in the selection of a suitable DSP. Floating-point operational support in DSPs facilitates the algorithm implementation, and increases the precision compared with fixed-point. In contrast, fixed-point operations can perform operations with fewer bits, but the programmer needs to carefully reposition the decimal point
after each mathematical operation. However, fixed-point DSPs are usually cheaper than floating-point DSPs. In DSPs, arithmetic computations are performed in multiplier–accumulator (MAC) units. Table A-1 summarises the fixed-/floating-point capabilities of the MAC units in various DSP series.

### Table A-1: TI-DSP families and the fixed-/floating-point properties of their MAC units

<table>
<thead>
<tr>
<th>DSP family</th>
<th>Fixed-/floating-point</th>
</tr>
</thead>
<tbody>
<tr>
<td>C5000 , C55x</td>
<td>Fixed-point</td>
</tr>
<tr>
<td>C64x</td>
<td>Fixed-point</td>
</tr>
<tr>
<td>C671x, C672x</td>
<td>Floating-point</td>
</tr>
<tr>
<td>C674x, OMAP</td>
<td>Fixed- and floating-point</td>
</tr>
<tr>
<td>DM37x, DM64x</td>
<td>Fixed-point</td>
</tr>
<tr>
<td>DM81x</td>
<td>Fixed- and floating-point</td>
</tr>
<tr>
<td>C66x, C667x, C665x</td>
<td>Fixed- and floating-point</td>
</tr>
<tr>
<td>C647x</td>
<td>Fixed-point</td>
</tr>
</tbody>
</table>

For the series with both fixed- and floating-point capabilities, MAC units can perform both types of calculations, but with fewer bits for floating-point operations. For instance, in a single clock cycle, the MAC units in the C66x can multiply either two 32-bit fixed-point or two 16-bit floating-point numbers.

### A.3.1 Available development tools and utilities for DSPs

Code Composer Studio (CCS) [232] is an integrated development environment (IDE) for developing, debugging, and compiling codes in TI-DSPs. The most widely used programming language for DSPs is C/C++. Although DSPs can also be programmed in assembly language, because of its complexity, it is only used by professional programmers in developing highly optimised codes. Free libraries have been developed to assist programmers in different tasks with optimised basic functions. Some of these libraries are:

- **DSP Library (DSPLIB) [233]**: originally developed for single core C6000 DSPs. However, it can also be used in multicore DSPs, since the architecture of multicore DSPs is based on the C6000, and is backwardly compatible. This library includes functions for some digital signal processing tasks, such as fast Fourier transform (FFT) and convolution algorithms.
• **Image Library (IMGLIB)** [234]: an optimised library for image processing on C64x or C55x DSPs. IMGLIB can also be used in multicore DSPs, since the C66x family support the C64x libraries [235]. This library contains some of the basic image processing functions, such as digital image filters.

• **Math Library for Floating Point Devices (MATHLIB)** [236]: an optimised floating-point library that includes functions for performing basic mathematical operations, and basic vector calculations.

A.3.2 Embedded operating systems

SYS/BIOS (formerly DSP/BIOS) is a real-time operating system (OS) developed by TI for programming its DSPs, microcontrollers, and ARMs [237]. SYS/BIOS is specifically designed for embedded systems where synchronisation of tasks and input/output data management are important. Alternatively, DSP families that include an ARM processor can support Linux. The OS in multicore DSPs is responsible for managing the interaction between the different cores, allocating tasks to a particular core, and deciding when to pass data across cores. The SYS/BIOS multicore software development kit (MCSDK) [238] helps programmers to manage these responsibilities in multicore DSPs by providing optimised platform-specific drivers, run-time libraries (OpenMP and OpenEM), and basic network protocols [239].

To analyse and profile the code while the application is running in multi-core DSPs, a real-time tool called a multicore system analyser (MCSA) [240] is added to the MCSDK. The MCSA provides real-time performance evaluation and monitors parameters such as the computation and task loads of each core, the computation time of various parts of the code, and concurrency of tasks in different cores.

A.3.3 Development time

Many useful tools are available for developing and debugging codes in C/C++ for DSPs (Sections 2.1 and 2.2), and available libraries include most of the general purpose algorithms
for computer vision and image processing applications. In general, the development time for a simple task in single core DSPs is relatively short. In contrast, developing an optimised code using parallel programming techniques for a multicore DSP is challenging, and complex tasks require advanced programing skills. This can lead to long development times when creating optimised codes for complex computer vision and image processing algorithms in multi-core DSPs.

A.3.4 Advantages of using DSPs

Important advantages of using DSPs for computer vision and image processing applications include:

1. The costs for developing a DSP-based portable system are typically low. DSP chips are generally cheaper than most of the other hardware accelerators for portable devices. In addition, TI provides free licenses for some versions of CCS and MCSDK with its hardware evaluation modules (EVMs), and all of the libraries introduced in Sections 2.1 and 2.2 are available free of charge.

2. Many computer vision and image processing applications involve sequential algorithms, and the chip architecture of DSPs is designed for the implementation of sequential tasks. Multi-core DSPs have added the capability of implementing coarse-grained parallelism for algorithms with a low-to-medium level of complexity.

3. The development time for simple computer vision and image processing algorithms in single core DSPs is, in general, relatively short.

4. Power consumption in DSPs is low. TI-OMAPs are specially designed for mobile applications making them a suitable option for portable devices that have limitations on their power consumption.

5. DSPs are suitable for handling peripherals, standard ports (e.g. USB, and SATA), and communication protocols (e.g. TCP/IP) in portable devices.

6. DaVinci-DMPs have video codec support, hence are suitable for video processing applications in portable devices.
A.3.5 Disadvantages of using DSPs

Disadvantages of using DSPs for computer vision and image processing applications include:

1. Multicore DSPs are designed for HPC applications with a low-to-medium level of complexity. They are not suitable for high data throughput or high speed applications.
2. DSPs are more suitable for sequential processing. Even though multicore DSPs are capable of performing coarse-grained parallelism, they are not a suitable choice for increasing the processing speed in massively parallel algorithms.
3. It is not usually efficient to use DSPs along with CPUs in PCs for increasing the processing speed of an algorithm. DSPs and CPUs both have a similar sequential processing nature, whereas programming with CPUs is easier and more efficient than using DSPs.
4. Although DSP chips are cheaper than FPGA and GPU chips for the same level of performance, TI-DSP boards are not usually cheap. A typical DSP board costs between 500 USD to 2000 USD [241], a similar price range to GPU boards. For PC-based systems that do not require external interfaces, DSP boards thus offer few advantages over GPU boards [242].

A.3.6 Review of applications that use DSPs

Table A-2 provides a summary of some selected computer vision and image processing algorithms that have been implemented on TI-DSPs in recent literature. As two examples of portable systems, an OMAP3530 was used for robotic applications in [243] and [244]. The DSP core of this OMAP is C64x, which is a fixed-point DSP (Table A-1). The DSP core and the ARM processor (Cortex8) of C64x work at 520 MHz and 720 MHz, respectively. The energy efficiency of the system implemented in [243] was better than all previously published DSP implementations. Even though this implementation was power efficient, the maximum
possible frame rate of the real-time application with this system was only 8 fps for an image size of 640 pixel × 480 pixel (Table A-2).

Table A-2: A summary of some selected computer vision and image processing algorithms implemented on TI-DSPs in the recent literature

<table>
<thead>
<tr>
<th>Application(s)</th>
<th>Algorithm(s) Implemented</th>
<th>Hardware (DSP)</th>
<th>Performance/Data throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stereo vision system for small robots (2011) [243]</td>
<td>• Sum of absolute differences (SAD) over a 7 pixel × 7 pixel window</td>
<td>OMAP3530 (Power optimised)</td>
<td>8 fps for 640 pixel × 480 pixel images and 60 levels of disparity (i.e. 147 M disparities/s)</td>
</tr>
<tr>
<td>Night-time vehicle detection (2011) [245]</td>
<td>• Bright object segmentation based on the image histogram • Spatial clustering • Feature-based identification and tracking.</td>
<td>DM642 (DaVinci-DMPs)</td>
<td>The implementation was tested with real highway images, but no performance was reported</td>
</tr>
<tr>
<td>Stereo vision system for tracking of moving objects (2011) [246]</td>
<td>• Image feature extraction (Colour interpolation, brightness compensation, conversion to the grey scale) • SAD algorithm for finding the disparity map • Edge stereo matching</td>
<td>C6416 (Power optimised)</td>
<td>8.92 fps for 356 pixel × 292 pixel images of a stereo pair camera (the processing time was approximately 112 ms for each image)</td>
</tr>
<tr>
<td>Guidance, navigation and control for UAV landing (2012) [247]</td>
<td>• Image feature extraction • Image matching and identification</td>
<td>DM642 (DaVinci-DMPs)</td>
<td>No performance or data throughput was reported</td>
</tr>
<tr>
<td>Motion estimation (2013) [248]</td>
<td>• Multi-channel gradient model</td>
<td>C6678 (Multicore)</td>
<td>9.75 fps for a video with 128 pixel × 128 pixel resolution</td>
</tr>
<tr>
<td>2D to 3D conversion based on disparity map estimation (2014) [249]</td>
<td>• Discrete wavelet transform (DWT) • Edge detection • Disparity map estimation • Colour segmentation using K-means clustering • Adaptive filtering</td>
<td>DM648 (DaVinci-DMPs)</td>
<td>8.83 fps for 1390 pixel × 1110 pixel images of a stereo pair (The average processing time was 113.25 s for each image)</td>
</tr>
<tr>
<td>Image registration (2014) [250]</td>
<td>• Multilevel Gauss-Newton minimisation for the rigid alignment of two images based on the SSD of the image intensities</td>
<td>4 x C6678 (Multicore)</td>
<td>The algorithm could register two 4096 pixel × 4096 pixel images at maximum 10.75 fps (93 ms for each pair)</td>
</tr>
<tr>
<td>Real-time image processing for robots (2014) [244]</td>
<td>• Finding landmarks and position estimation</td>
<td>OMAP3530 (Power optimised)</td>
<td>The algorithm could estimate the position from 576 pixel × 720 pixel images</td>
</tr>
</tbody>
</table>
DaVinci-DMPs have been used in [245], [247], and [249] for the implementation of an image processing system with a low level of complexity (details are in Table A-2). DM642 ([245], [247]) and DM648 ([249]) are both fixed-point DSPs (Table A-1) without an ARM processor. The maximum processing clock rates for the DSP cores of the DM642 and DM648 are 720 MHz and 1.1 GHz, respectively. In [245] and [247], the DM642 was used in a video processing application, for which the DaVinci-DMPs were specifically designed.

The C6416, which is a power optimised fixed-point DSP (Table A-2), was used in [246] to implement a simple stereo vision tracking algorithm based on the sum of absolute differences (SAD). This system included a CMOS camera, and was designed to be used in stand-alone portable devices or robots.

The C6678, which is a multicore DSP with 8 cores (each core operates at 1 GHz), is able to perform both fixed-point and floating-point operations, and was used for rather complicated tasks in [248] and [250]. In [248], a motion estimation algorithm was implemented and, even though the C6678 is among the highest-performing DSPs available, the authors could not achieve high frame rates for large image sizes [248]. For an image size of 128 pixel \(\times\) 128 pixel the maximum frame rate they could achieve was 9.79 fps, which outperformed the Intel Xeon CPU when one core was used (3.99 fps), but not when all 4 cores (8 threads) were used (20.91 fps). Nevertheless, the C6678 showed considerably lower power consumption compared with the CPU (single core and multiple core implementations). An image registration implementation for embedded systems was proposed in [250]. The authors combined four C6678 DSP boards to increase the processing power of their system, and achieved 10.75 fps for an image size of 4096 pixel \(\times\) 4096 pixel (a processing time of 93 ms for each image). Advantages of this system include being cost-effective, energy-efficient, and suitable for embedded systems. Nevertheless, the algorithm they implemented was a rigid image registration algorithm with moderate complexity, consisting of an optimisation process for minimising the sum of square differences (SSD) of the image intensity values (Table A-2).
A.3.7 Summary and conclusion for DSPs

In this section, DSPs and their capabilities have been introduced and their pros and cons have been presented. Examples of their application in the literature indicate that they are not particularly suitable for high-performance applications. Despite recent advances in TI-multicore DSPs [251], it still is not feasible to implement complex computer vision and image processing algorithms on DSPs, especially when the data throughput requirement is high. Furthermore, DSPs are not particularly suitable for PC-based systems since, apart from external interfaces, they do not provide significant advantages over GPUs. In contrast, DSPs are an energy-efficient and cost-effective solution for embedded systems, and for mobile or portable devices in which the computational demands are not high, and the power consumption level is critical.

A.4 Field-programmable gate arrays (FPGAs)

The FPGA chip incorporates arrays of reprogrammable logic gates. As opposed to CPUs, DSPs, and GPUs, FPGA fabrics do not have a pre-structured chip architecture or a central processing unit. Thus, prior to programming the reconfigurable FPGAs, the programmer should design a hardware architecture for their specific application using the logic gates inside the FPGA.

The FPGA hardware architecture is configured by interconnecting FPGA logic gates to perform a specific task, and requires reconfiguration for each new algorithm. FPGAs are thus often referred to as reconfigurable devices. The programming languages for FPGAs are quite different from those for CPUs, DSPs, and GPUs. Hardware description languages (HDLs), such as VHDL [252], and Verilog [253] are the most common programming languages for configuring FPGAs. However, HDLs are low-level and complicated programming languages, particularly for beginners. To simplify the programming of FPGAs, some high-level programming languages, such as C-like languages (e.g. SystemC [254]), and domain-specific languages (DSLs) [255–258] have been developed for FPGAs. Nevertheless, C-like languages are not particularly suitable for non-sequential algorithms, and DSLs are only suitable for a
limited range of tasks. In general, developing efficient algorithms in FPGAs requires a good understanding of hardware-level details.

Xilinx and Altera are the two main vendors of FPGAs currently being used by the computer vision and image processing research community. These FPGA families are covered in the following two sections.

A.4.1 Xilinx FPGA families

Table A-3 provides a summary of Xilinx FPGA families, their fabrication process technology, and their year of introduction. The Spartan series are low-cost FPGAs which are designed for relatively simple applications. The Virtex series are specifically designed for performing signal processing tasks, and are relatively expensive compared with other FPGA families. The Kintex and Artix series are low-performance and inexpensive versions of the Virtex-7. Zynq series are medium to high performance SOCs, which include an FPGA (Artix or Kintex) and an ARM processor to handle sequential tasks, memory interface, and standard input/output ports [259].
Table A-3: Xilinx FPGA families, their process technology, and their year of introduction

<table>
<thead>
<tr>
<th>FPGA Family</th>
<th>Process Technology (Year of Introduction)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-2</td>
<td>180 nm (2000)</td>
</tr>
<tr>
<td>Virtex-2</td>
<td>150 nm (2001)</td>
</tr>
<tr>
<td>Virtex-2 pro</td>
<td>130 nm (2002)</td>
</tr>
<tr>
<td>Virtex-2 pro x</td>
<td>130nm (2003)</td>
</tr>
<tr>
<td>Virtex-4, Spartan-3E</td>
<td>90 nm (2005)</td>
</tr>
<tr>
<td>Virtex-5</td>
<td>65 nm (2006)</td>
</tr>
<tr>
<td>Spartan-3A</td>
<td>90 nm (2007)</td>
</tr>
<tr>
<td>Virtex-5 FXT</td>
<td>65 nm (2008)</td>
</tr>
<tr>
<td>Virtex-6, Spartan-6</td>
<td>40 nm (2009)</td>
</tr>
<tr>
<td>Virtex-7, Kintex, Artix, Zynq</td>
<td>28 nm (2010)</td>
</tr>
<tr>
<td>UltraScale (Virtex, Kintex, Zynq)</td>
<td>20 nm (2013)</td>
</tr>
<tr>
<td>UltraScale+ (Virtex, Kintex, Zynq)</td>
<td>16 nm (2013)</td>
</tr>
</tbody>
</table>

The UltraScale and UltraScale+ families are the latest Xilinx technologies. UltraScale and UltraScale+ are manufactured using 20 nm and 16 nm planar fabrication processes, respectively. The Virtex UltraScale+ FPGA is Xilinx’s highest performance FPGA, and is designed for high-performance and high-speed applications. In general, the Spartan and Artix families have low to medium performance, whereas the Virtex and Kintex families can be used for algorithms with medium to high levels of computational complexity. The detailed specifications of Xilinx FPGAs for the Virtex-5, Virtex-6, 7-series (including Virtex-7, Kintex 7, and Artix 7), and UltraScale series are available in [260], [261], [262], and [263], respectively.
Important features for the selection of a suitable FPGA for computer vision and image processing tasks are summarised below:

- Configurable logic blocks (CLBs) are hardware resources and logic gates for algorithm implementation in Xilinx FPGAs. The logic cells of CLBs can be connected to each other to form large shift registers (SRs), multiplexers (MUXs), look-up tables (LUTs), and distributed memories. The interconnection of CLBs forms a unified logic circuit, which is the hardware representation of the algorithm. FPGAs that have large numbers of CLBs are a suitable choice for complex algorithms. However, the type of logic gates, and the number of input bits vary across the different FPGA families. Because of this, the number of CLBs in an FPGA is not a useful comparison metric.

- DSP slices are designed for robust implementation of basic mathematical operations and signal processing tasks. These blocks are called DSP48 [264] in Xilinx FPGAs, and can have different numbers of bits across the various Xilinx families. DSP48 blocks were first introduced in Virtex-2 families, and have been further developed in later Virtex families.

- DSP performance is quantified by the maximum number of mathematical operations that a single DSP slice of Xilinx FPGAs (DSP48) can perform per second. DSP performance can be the main bottleneck when dealing with high throughput data in real-time applications.

- Block RAMs are designed with SRAM architecture for the storage or buffering of data, and are especially important for storing image data inside the FPGA chip. The storage demands of image processing or computer vision algorithms, and the available block RAMs in an FPGA, should be considered when choosing an appropriate option.
A.4.1.1 Tools and utilities for Xilinx FPGAs

There are several tools to facilitate the development of code in Xilinx FPGAs. These tools, introduced by Xilinx or third parties, are discussed in this section.

The Xilinx integrated synthesis environment (ISE) design suite [265] is the Xilinx software package for configuring its FPGAs. Xilinx ISE is not a free tool, but Xilinx offers a free limited edition of ISE called WebPack [265], and provides a free licence with its own FPGA boards. The ISE WebPack does not support some of the Xilinx FPGA families, and does not have some of the features of the full version ISE (such as Xilinx SysGen).

Xilinx intellectual property cores (IP-cores) [266] are optimised hardware-implemented algorithms for performing various tasks for a wide range of applications. Most of the Xilinx IP-cores are not available free of charge, but the basic algorithms are included in the Xilinx ISE licence. Third party IP-cores are also available for specialised algorithms, which can be purchased separately. IP-cores cover most of the fundamental functions, and can help to significantly reduce the development time of computer vision and image processing algorithms in FPGAs. The drawback is that purchasing IP-cores increases the cost of the project.

Xilinx introduced the Vivado Design Suite [267] for configuring Xilinx 7-series FPGAs (i.e. Virtex-7, Kintex, Artix, and Zynq), and UltraScale families. However, other Xilinx FPGAs can also be configured using Vivado instead of Xilinx ISE. Vivado is able to synthesise codes faster than Xilinx ISE using a new algorithm for configuring the FPGAs [268]. Vivado also includes a high-level synthesis (HLS) tool for C-based IP generations in a high-level language (C, C++, or SystemC). The Xilinx HLS tool was demonstrated to be faster than HDLs (i.e. VHDL, Verilog) for developing optimised codes for sequential algorithms [269].

The Xilinx embedded development kit (EDK) [270] is a tool developed for designing and programming embedded processors inside the FPGA chip. Embedded processors have two main categories in Xilinx FPGAs, named Microblaze and PowerPC. Microblaze can be designed and added to an FPGA using the available logic cells of the FPGA (i.e. it is a soft processor), while PowerPC is a pre-built processor designed by Xilinx (i.e. it is a hard
PowerPCs are based on reduced instruction set computer (RISC) technology, and are only available in some of the Xilinx FPGA families. Xilinx embedded processors are programmed using the C/C++ language, and are suitable for sequential tasks or handling external interfaces (such as DDR SDRAMs). Xilinx embedded processor cores are inside the FPGA chip. It is thus simple to develop a data transfer interface between them and the rest of the code, and it is possible to achieve higher speeds compared with external interfaces. In fact, Xilinx embedded processor cores were developed to add the sequential processing power of these processors to the parallel processing of the FPGA. However, the clock rate of Xilinx embedded processors is only in the order of hundreds of MHz, hence are unsuitable for computationally expensive algorithms. The ARM processors in Zynq and Zynq UltraScale+ have a considerably higher performance compared to Microblaze and PowerPC. However, neither Xilinx embedded processors nor ARMs can cope well with complicated algorithms, and FPGAs are still not particularly suitable for heavily sequential tasks.

Xilinx system generator (SysGen) [271] is a tool designed to simplify the implementation of digital signal and image processing algorithms. It is a high-level tool for designing high-performance systems using FPGAs in MATLAB’s Simulink environment. Xilinx SysGen can be used to develop efficient codes for Xilinx FPGAs, and can help to significantly reduce the development time for complicated modular-based signal processing algorithms. FPGA basic logic cells and Xilinx IP-cores are accessible within Xilinx SysGen in a modular and block-based format. One of the advantages of Xilinx SysGen is its ability to use the software blocks of MATLAB’s Simulink to test and debug the code implemented in hardware blocks of the FPGA (i.e. software and hardware co-simulation). This feature of Xilinx SysGen can significantly reduce the test and debug time compared with only using HDLs for developing codes. Xilinx has a design and analysis tool called PlanAhead that is used in conjunction with the Xilinx ISE. PlanAhead can be used to perform floorplanning, verify the design, and analyse the implementation details of the algorithms. Some of the parameters which can be monitored using Xilinx PlanAhead are the occupied hardware resources (i.e. CLBs, Block RAMs, and
DSP48s), maximum speed, and the power consumption [272]. This tool can help designers to refine their code to increase its robustness and performance. Most of the PlanAhead software features were integrated into the Vivado design suite for the more recent FPGAs.

Xilinx Chipscope [273] is a software/hardware package for on-line and real-time debugging of the FPGA codes. Xilinx Chipscope requires its special hardware to directly capture real-time data from the FPGA. The captured data is stored in a buffer inside the FPGA during runtime, and is transferred to the PC via a USB port for debugging with the Xilinx Chipscope software. Even though this tool assists with the debugging process, the drawback is the need to have sufficient free memory space inside the FPGA for buffering the data.

SDAccel [274] is the Xilinx development environment for OpenCL. OpenCL is an open standard, which is maintained by a technology consortium called the Khronos Group [275]. The OpenCL development environment facilitates the development of C-based high-level codes for FPGAs. This tool can be used to develop and emulate kernels in the host PC, debug them and generate an implementation report for the FPGA, and then convert that kernel to an FPGA code. This tool helps programmers to decrease the development time for sequential algorithms by writing the code and debugging it in a high-level language. For instance, the PCIe interface can be configured with OpenCL to transfer data to an FPGA from the host PC.

In addition to official Xilinx tools, some third party graphical language tools are also available for programming Xilinx FPGAs. The National Instruments (NI) LabView FPGA module [276] is one of the most popular. These graphical languages may generate inefficient codes if the programmer is inexperienced in the use of high-level modules. To help develop efficient FPGA codes in NI LabView, Xilinx IP-cores were added to the versions since 2014. The main disadvantage of the NI LabView FPGA module is its limitation to NI FPGA boards. NI FPGA boards usually require special hardware accessories, such as the NI chassis (further details are described in [277]), and are considerably more expensive than similar Xilinx FPGA boards.
A.4.2 Altera FPGA families

Altera FPGAs comprise three main families: Stratix [278]; Arria [279]; and Cyclone [280]. Table A-4 shows the Altera FPGAs, their process technology, and their year of introduction. Among them, the Stratix series is designed for medium to high performance algorithms, and is similar to the Virtex series in Xilinx FPGAs. The Altera Stratix-10 is a high-performance FPGA that has twice the performance and 70% lower power consumption compared to Stratix-5 [281]. Comparing Table A-3 and Table A-4 shows that the Xilinx and Altera FPGAs of the same class of performance usually use the same fabrication process technology.

Table A-4: Altera FPGA families, their process technology, and their year of introduction

<table>
<thead>
<tr>
<th>FPGA Family</th>
<th>Process Technology (Year of Introduction)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stratix, Cyclone</td>
<td>130 nm (2002)</td>
</tr>
<tr>
<td>Stratix-GX</td>
<td>130 nm (2003)</td>
</tr>
<tr>
<td>Stratix-2, Cyclone-2</td>
<td>90 nm (2004)</td>
</tr>
<tr>
<td>Stratix-2 GX</td>
<td>90 nm (2005)</td>
</tr>
<tr>
<td>Stratix-3</td>
<td>65 nm (2006)</td>
</tr>
<tr>
<td>Cyclone-3, Arria-GX</td>
<td>65 nm (2007)</td>
</tr>
<tr>
<td>Stratix-4</td>
<td>40-nm (2008)</td>
</tr>
<tr>
<td>Cyclone-4, Arria-2</td>
<td>60 nm (2009)</td>
</tr>
<tr>
<td>Stratix-5, Arria-2-GZ</td>
<td>28 nm (2010)</td>
</tr>
<tr>
<td>Cyclone-5, Arria-5</td>
<td>28 nm (2011)</td>
</tr>
<tr>
<td>Arria-10</td>
<td>20 nm (2013)</td>
</tr>
<tr>
<td>Stratix-10</td>
<td>14 nm (2013)</td>
</tr>
</tbody>
</table>
Altera Stratix FPGAs have dedicated hardware blocks for performing robust mathematical and logical operations [282]. These blocks have similar functionality to DSP slices (DSP48) in Xilinx Virtex FPGAs, and can increase the speed and performance of signal processing algorithms.

A.4.2.1 Tools and utilities for Altera FPGAs

Altera has introduced some tools to facilitate the development of code in its FPGAs, similar to those previously discussed in Section A.4.1.1 for Xilinx FPGAs. These software tools are Quartus II for developing and compiling FPGA codes [283], the Nios II embedded design suite (EDS) for embedded software development [284], DSP builder for developing FPGA signal processing algorithms in MATLAB’s Simulink environment [285], and an SDK for OpenCL programming [286]. Table A-5 lists Altera’s tools for developing FPGA codes, and the corresponding tools for Xilinx FPGAs.

### Table A-5: Corresponding tools for developing codes in Altera and Xilinx FPGAs

<table>
<thead>
<tr>
<th>Altera FPGAs</th>
<th>Xilinx FPGAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Altera Quartus II</td>
<td>Xilinx ISE, Xilinx Vivado</td>
</tr>
<tr>
<td>Altera Nios II EDS</td>
<td>Xilinx EDK</td>
</tr>
<tr>
<td>Altera DSP builder</td>
<td>Xilinx System Generator</td>
</tr>
<tr>
<td>Altera OpenCL SDK</td>
<td>Xilinx SDAccel</td>
</tr>
</tbody>
</table>

Altera and third parties have developed IP-cores for performing various tasks in Altera FPGAs. These IP-cores can simplify the implementation of computer vision and image processing algorithms. In addition, some open-source libraries are available in OpenCL for Altera FPGAs.

A.4.3 Development time

The development time for FPGAs is longer than for other hardware accelerators. Even though recent offerings in software tools were intended to reduce the long development time
in FPGAs, developing robust codes in FPGAs is still challenging. Parallel programming in FPGAs requires skilled programmers with sufficient technical knowledge of the FPGA hardware details. High-level functions are not available for FPGAs without payment. Hence algorithms must often be developed using basic functions, which is a time-consuming process. For instance, in a study in 2012, it took 12 months for two postdoctoral employees to implement the algorithms for extracting dense optical flow, and image features in a Virtex-5 FPGA [287]. Nevertheless, it is expected that the development time of FPGAs will reduce with the emergence of the new generation of HDLs, such as Bluespec System Verilog (BSV) and Chisel [288].

A.4.4 Advantages of using FPGAs

Important advantages of using FPGAs for computer vision and image processing algorithms include:

- The processing speed of FPGAs is higher than other hardware accelerators. Recent FPGAs from both Xilinx and Altera can process billions of operations per second in parallel with their DSP blocks ([289] and [290], respectively). Such speeds have not been matched by any other hardware accelerator.

- FPGAs can have high data throughput, hence are good choices for data capture cards (e.g. video capture cards).

- The parallel and reconfigurable nature of the FPGAs is a useful feature, enabling FPGA hardware to be designed and configured for high performance applications.

- FPGAs are relatively energy efficient. In many studies, such as [287], it has been shown that FPGAs have the highest processing power when normalised for power consumption.
• Because of their low power demands, FPGAs are good candidates for use in portable devices.

• FPGAs have industrial and military grades for use in harsh working conditions;

• Programmers can implement flexible and efficient algorithms in FPGAs by reconfiguring the FPGA hardware optimised for the algorithm.

• FPGA codes can be adapted for use in application-specific integrated circuits (ASICs). ASICs reduce mass production costs.

A.4.5 Disadvantages of using FPGAs

Important disadvantages of using FPGAs for computer vision and image processing algorithms include:

• The long development time of FPGAs is their main drawback.

• FPGAs are expensive compared to GPU boards. The price of a modern FPGA board can be more than 10,000 USD.

• It is challenging to develop efficient codes in FPGAs. Even though new tools have been developed for FPGAs to make this process simpler and faster, the programmer still needs sufficient technical skills to develop robust codes in FPGAs.

• Even though some open source packages, such as RIFFA [291] and ThreadPoolComposer [292] are available to help developers, developing a PCIe interface for data communication between multiple FPGA boards and the host PC in high-speed applications is a complicated task.

A.4.6 Review of applications that use FPGAs

In the first part of this section, the use of FPGAs in stereo vision systems is discussed. In the second part, the use of FPGAs in other image processing and computer vision applications is reviewed.
A.4.6.1 FPGAs in stereo vision systems

The most common application of FPGAs in stereo vision systems is the implementation and optimisation of stereo-correspondence algorithms. In 2013, Tippetts et al. published a comprehensive review of various stereo vision algorithms, and their suitability for resource-limited systems (including FPGAs) [217]. In their review, the algorithms were evaluated based on accuracy and speed. Despite this being a common way of comparing algorithms, the accuracy and speed of an algorithm is dependent on the programming language, programmer skills, and the type of hardware being used. Therefore, such comparisons are not precise enough to show the suitability of the hardware accelerator itself. To compare different algorithms in FPGAs, Tippetts et al. [217] reviewed 12 papers, of which the highest rate of disparities for stereo reconstruction was reported for the implementation of Ambrosch et al. [293]. However, they reported Quartus II as the FPGA of the implementation of Ambrosch et al., whereas in fact Quartus II is not an FPGA, but Altera’s software tool for developing codes (Table A-5). Ambrosch et al. simulated their algorithm for an Altera Stratix-2 FPGA, and could achieve 10,108 M disparities/s [293] (reported to be 6,062.9 disparities/s in Tippetts et al.’s review paper [217]).

Stereo correspondence algorithms and their implementation for FPGAs were evaluated by Colodro-Conde et al. [294] in 2014. The algorithms involved a trade-off between the hardware resources they used, and the speed they could reach. Colodro-Conde et al. [294] concluded that SAD is the most suitable stereo correspondence algorithm for being implemented in FPGAs, because of its highly parallelised nature and relatively straightforward implementation.

Table A-6 provides a summary of some of the high performance FPGA implementations for stereo vision algorithms in recent literature. In these papers, three of the six algorithms were implemented using Altera Stratix FPGAs (Stratix-2, Stratix-3, and Stratix-4), and the other three were implemented using Xilinx Virtex FPGAs (Virtex-4, and Virtex-5). The most common algorithm for stereo vision systems in these papers was SAD, which is in accordance with its suitability for implementation in FPGAs [294]. However, details of the implemented
stereo matching algorithms differ between these papers, hindering direct comparison of the performance of these FPGAs.

**Table A-6: A summary of some selected FPGA implementation of stereo vision algorithms in the recent literature**

<table>
<thead>
<tr>
<th>Application(s)</th>
<th>Algorithm(s) Implemented</th>
<th>Hardware (FPGA)</th>
<th>Performance/Data Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAD-based stereo matching (2009) [293]</td>
<td>· SAD over a 9 pixel x 9 pixel window.</td>
<td>Simulated for Altera Stratix-2</td>
<td>10108 M disparities/s (i.e. 599 fps for 450 pixel x 375 pixel images and 100 disparity level)</td>
</tr>
<tr>
<td>Real-time stereo vision (2010) [295]</td>
<td>· Noise reduction</td>
<td>Xilinx Virtex-5</td>
<td>4050.1 M disparities/s (i.e. 103 fps for 640 pixel x 480 pixel images and 128 disparity level)</td>
</tr>
<tr>
<td>Real-time stereo vision system (2010) [296]</td>
<td>· Image rectification</td>
<td>Xilinx Virtex-4</td>
<td>4521.9 M disparities/s (i.e. 230 fps for 640 pixel x 480 pixel images and 64 disparity level)</td>
</tr>
<tr>
<td>Disparity map computation (2010) [297]</td>
<td>· SAD</td>
<td>Altera Stratix-4</td>
<td>7864.3 M disparities/s (i.e. 320 fps for 640 pixel x 480 pixel images and 80 disparity level)</td>
</tr>
<tr>
<td>Low-cost FPGA stereo vision system (2012) [298]</td>
<td>· Lens distortion removal (for radial and tangential distortion).</td>
<td>Xilinx Virtex-4</td>
<td>8985.6 M disparities/s (i.e. 325 fps for 640 pixel x 480 pixel images and 90 disparity level)</td>
</tr>
<tr>
<td>Stereo-vision system (2013) [299]</td>
<td>· Image rectification of stereo images using a lookup table;</td>
<td>Altera Stratix-3</td>
<td>3019.8 M disparities/s (i.e. 30 fps for 1024 pixel x 768 pixel images and 128 disparity level)</td>
</tr>
<tr>
<td>Edge-directed real-time disparity map computation (2013) [300]</td>
<td>· Sobel edge detection (consisted of a convolution unit with the Sobel horizontal and vertical kernels);</td>
<td>Xilinx Virtex-5</td>
<td>7864.3 M disparities/s (i.e. 50 fps for 1280 pixel x 1024 pixel images and 120 disparity level)</td>
</tr>
</tbody>
</table>

A.4.6.2 FPGA in non-stereo computer vision and image processing applications

Table A-7 provides a summary of some selected FPGA implementations for some non-stereo computer vision and image processing algorithms in the recent literature. Not all of these papers reported a speedup ratio. However, for those in which it was reported, the algorithms implemented in Virtex-6, or Virtex-7 FPGAs could achieve considerably higher
speedup ratios, compared to those in Virtex-2, or Virtex-4 FPGAs. Among the algorithms in Table A-7, the complexity was substantially greater for the last two algorithms, which were implemented in a Virtex-6 [301], and a Virtex-7 [302]. This suggests that Virtex-6 and Virtex-7 FPGAs are suitable options for complex computer vision and image processing algorithms.

As an example of using embedded processor cores, Microblaze was used in [303] to handle the external interfaces and manage the partial reconfiguration capability of the code (i.e. the reconfiguration of some parts of the code while the code is running). To shorten the development time, Xilinx SysGen was used in [304] and [305] to implement modular-based signal or image processing algorithms.
### Table A-7: A summary of some selected FPGA implementation of non-stereo computer vision and image processing algorithms in the recent literature

<table>
<thead>
<tr>
<th>Application(s)</th>
<th>Algorithm(s) Implemented</th>
<th>Hardware (FPGA)</th>
<th>Performance/ Data Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometric algebra in modelling and rendering 3D images (2009) [306]</td>
<td>• Vector calculations in Clifford algebra, Estimation of the pixel distances, and finding their minimum using SAD</td>
<td>Xilinx Virtex-2</td>
<td>Test results showed the potential to achieve speedup of 4 to 20-fold in comparison to the general purpose CPU implementation of Clifford algebra</td>
</tr>
<tr>
<td>Real-time 3D surface model reconstruction from Integral Images (2010) [307]</td>
<td>• Pre-processing (including Gaussian noise reduction, histogram stretching, and morphological operations); Model fitting (modified version of RANSAC algorithm); Model tracking</td>
<td>Xilinx Virtex-5</td>
<td>The proposed architecture was able to process 3D data at 34 images/s for image size of 2048 pixel × 2048 pixel</td>
</tr>
<tr>
<td>Roadway path extraction and tracking (2010) [304]</td>
<td>• Modified computer-generated hologram (CGH) algorithm with CGH kernels and cells</td>
<td>Xilinx Spartan-3</td>
<td>The implemented algorithm was able to process video sequences at 30 frames/s</td>
</tr>
<tr>
<td>Digital hologram generator (2010) [308]</td>
<td>• Ising Model with a Monte Carlo update. (The algorithm includes parallel convolutions units, delays and LUTs)</td>
<td>Xilinx Virtex-2</td>
<td>The implemented algorithm could generate one frame of the CGH with a size of 1408 pixel × 1050 pixel and 10,000 light sources in 0.0093 s</td>
</tr>
<tr>
<td>Atomistic magnetic spin simulations (2011) [309]</td>
<td>• 2D convolution</td>
<td>Xilinx Virtex-4</td>
<td>The FPGA implementation was faster than an Intel Xeon X5560 CPU at a clock rate of 2.80 GHz</td>
</tr>
<tr>
<td>Run-time self-reconfigurable 2D convolver (2011) [303]</td>
<td>• Symmetric image downscaling; Classifier sharing; Cascade merging</td>
<td>Xilinx Virtex-5</td>
<td>307 frames/s for image size of 640 pixel × 480 pixel</td>
</tr>
<tr>
<td>High-speed face detection (2012) [310]</td>
<td>• Mapping between the distorted and undistorted image; Pixel reconstruction (i.e. using the mapping function and interpolation to find the new pixel positions)</td>
<td>Xilinx Virtex-4</td>
<td>367 fps for the image size of 640 pixel × 480 pixel, and 120 fps for the image size of 1280 pixel × 720 pixel</td>
</tr>
<tr>
<td>Image rectification for stereo vision (2013) [311]</td>
<td>• Hierarchical 2D cross-correlation</td>
<td>Simulated for Xilinx Virtex-6</td>
<td>Simulations showed approximately speedup of 200-fold in the processing time in comparison to an Intel Xeon CPU, and an ordinary laptop’s GPU</td>
</tr>
<tr>
<td>2D cross-correlation for real-time surface tracking (2013) [305]</td>
<td>• Edge detection in 3D using rotors in geometric algebra</td>
<td>Xilinx Virtex-7</td>
<td>The implementation was able to achieve speedup of 11.8x against an Intel Core i7 CPU clocked at 3.20 GHz.</td>
</tr>
<tr>
<td>Geometric Algebra in colour Edge Detection (2013) [312]</td>
<td>• Image acquisition; Background generation; Segmentation; Presentation of the results</td>
<td>Xilinx Virtex-6</td>
<td>60 fps for colour images with a resolution of 1920 pixel × 1080 pixel. As a comparison, it took 1.7 s to process a single frame with a C++ code in an Intel Core i7 CPU (speedup of 102-fold)</td>
</tr>
<tr>
<td>Real-time background generation and foreground object segmentation for high-definition colour video stream (2014) [301]</td>
<td>• Accelerating the probability boundary (Pb) detector algorithm (Pb is a gradient-based algorithm)</td>
<td>Xilinx Virtex-7</td>
<td>The execution time was 0.0063 s in the FPGA as oppose to 72.494 s in a 2.1 GHz CPU (speedup of 11507-fold)</td>
</tr>
<tr>
<td>Image boundary detection (2014) [302]</td>
<td>• Run-based algorithm with a Monte Carlo update. (The algorithm includes parallel convolutions units, delays and LUTs)</td>
<td>Xilinx Virtex-7</td>
<td></td>
</tr>
</tbody>
</table>
A.4.7 Summary and conclusion for FPGAs

FPGAs are the most flexible hardware accelerators for the implementation of customised computer vision and image processing algorithms. However, good knowledge in digital logic design, hardware architecture, HDLs, and programming tools are essential for implementing efficient complex algorithms in FPGAs. Most of the computer vision and image processing algorithms are designed for sequential processors. Hence, achieving an acceptable performance in FPGAs is only possible when the algorithm is modified and optimised for parallel processing. For instance, the standard sequential algorithms for corner detection and frontal face detection were modified to be optimised for FPGA implementation by Lim et al. [313]. As a result, the optimised code could be implemented in an Altera Cylone-4 FPGA (a relatively inexpensive, low-end FPGA with few hardware resources). Lim et al. in [313] showed that even though they had used a low-end FPGA, their optimised algorithm could achieve a similar or higher speed compared with similar non-optimised algorithms implemented in high-end FPGAs.

In summary, some applications for which FPGAs are suitable options include:

- FPGAs are the best option for algorithms with high computational demands in a portable PC-independent device. FPGAs are low-power, can be used in embedded systems, and are designed for high performance tasks.

- For designs that will be mass produced, FPGAs are suitable options, since an ASIC can easily be designed and produced based on an FPGA design. ASICs substantially reduce the costs for mass production.

- Because of their high data throughput, FPGAs are the most suitable option for capturing and processing high-frame-rate data from high-speed cameras. The image data can be processed in the FPGA at a high speed.
However, FPGAs are expensive, and development times using traditional methods are usually extensive. For instance, the average time between the introduction of an FPGA and its publication year was \((5.2 \pm 1.8)\) years for the papers in Table A-6, and \((5.3 \pm 2.0)\) years for the papers in Table A-7. Even though FPGAs are not introduced each year (Tables 3 and 4), and it takes further time to develop and release a commercially available FPGA board, this duration highlights the fact that, in most publications, the FPGA technology used was approximately 5 years old. However, if the FPGA was chosen from the latest technologies (i.e. Virtex-6, and Virtex-7), good performance could be achieved. For example, an impressive speedup of 11507-fold was achieved with a Virtex-7 in [302].

A.5 Graphics processing units (GPUs)

The first graphics accelerators were built for professional graphics workstations, such as the Infinite Reality for the Onyx series [314]. GPUs consist of many processing cores, and are accelerators that are optimised for performing fast matrix calculations in parallel (images are in the form of 2D matrices). These devices are typically very affordable, since their development is motivated by the gaming industry. GPUs are thus cost-effective hardware accelerators for massively parallel algorithms. GPUs have been used in a wide range of applications, other than games, over the last ten years.

NVidia is the most widely known vendor of GPUs. AMD and Intel are other major producers of GPUs. In this review, only NVidia GPUs were evaluated, since they are widely used by the research community.

A.5.1 NVidia GPU series

NVidia GPU series have a range of core microarchitectures, and can be used for various image processing applications [315] [316]. Some of the main features to consider when selecting a suitable GPU for a specific application include (more details about the GPU hardware architecture can be found in [224]):
- **GPU microarchitecture technology.** An important feature of a GPU is the generation of its microarchitecture. The four most recent NVidia microarchitectures were named Tesla [317], Fermi [318], Kepler [319], and Maxwell [320], and were introduced in 2008, 2010, 2012, and 2014, respectively. Each microarchitecture technology has its own features, and specifications. For example, the Kepler microarchitecture has a high computational power, and the Maxwell microarchitecture has a power-efficient design and an improved scheduler that provides higher delivered performance per core compared to the earlier GPU microarchitectures [321].

- **Memory.** GPUs have varying levels of memory. The internal memory of GPUs is typically used for storing image data. This memory is of DRAM type, but is based on different technologies in different GPUs. The main DRAM technologies are DDR2, DDR3, GDDR3, and GDDR5, and each has a different speed and bandwidth. Even though an adequate amount of DRAM memory is necessary for storing the image data for the algorithm, DRAM memory is costly, and GPUs of the same generation that have more memory are more expensive. Therefore, a suitable GPU should be chosen by considering the memory demands of the algorithm.

- **Compute unified device architecture (CUDA) cores.** CUDA cores or stream processors are the smallest processing units of NVidia GPUs, and each task can be assigned to one of them. NVidia microarchitectures have different numbers of CUDA cores. A group of CUDA cores forms a streaming multiprocessor (SM). For instance, the NVidia Fermi architecture has 512 CUDA cores, while Nvidia Kepler architecture has 2880 CUDA cores. Massively parallel computer vision and image processing algorithms are suitable to be implemented in GPUs with large numbers of CUDA cores.

- **CUDA compute capability.** CUDA compute capability version indicates some of the main features for programming in GPUs, including the maximum shared memory and
the maximum 2D and 3D array size in CUDA cores. NVidia has released four major versions of CUDA compute capability up to 2015 (1.x, 2.x, 3.x, and 5.x).

- **Processing power.** The maximum number of floating point operations (single or double precision) per second that a GPU can perform is defined as the processing power of that GPU in FLOPs (an acronym for floating-point operations per second). The performance of a GPU in high-performance computing is often measured based on its capability in performing multiplication and addition or fused multiply-add (FMA), which is two FLOPs per instruction.

- **Bus interface.** All modern GPUs use a PCIe interface for data communication with PCs. However, the generation of the PCIe interface, and the number of data lanes that the GPU can use, will determine the maximum data transfer rate.

Among NVidia GPUs, the Tesla series with Kepler microarchitecture was particularly designed for performing technical and scientific computing [322], hence they are the most common GPUs in high performance applications. GPUs are inherently suitable for performing computer vision and image processing tasks, and many different GPU series can be used for this purpose. NVidia has introduced some examples of computer vision and image processing tasks in [323].

### A.5.2 Tools and utilities for NVidia GPUs

CUDA and OpenCL are the two main programming languages for GPUs. Fang et al. [324] published a detailed performance comparison of CUDA and OpenCL, in 2011.

CUDA was created by NVidia, and is a parallel computing platform and programming model [325]. Not all NVidia GPUs support CUDA, but all the GPUs that are released after 2007 have CUDA support (the list of NVidia GPUs with CUDA support is available in [326]). OpenCL is a programming language which can be used in many different platforms, such as GPUs, CPUs, DSPs, and recently some FPGAs and ARMs (an example is Altera’s OpenCL SDK, which was introduced in Section 3.2.1). One of the main advantages of OpenCL is its
portability across different platforms. OpenCL has the potential to have the same performance as CUDA under a fair comparison, but it requires a higher level of programming skill [324]. CUDA is the preferred programming language for NVidia GPUs. CUDA has a wide range of support from NVidia, it does not require advanced programming skills, and has many libraries and development tools.

The list and description of the libraries and tools available for CUDA can be found in [327]. Some of the basic CUDA libraries for image processing algorithms include:

- **CUBLAS** was developed for linear algebra calculations. According to NVidia, CUBLAS was 6 to 17 times faster than Intel’s math kernel library (MKL) BLAS for CPUs [328].

- **CUFFT** is a library for performing FFT. According to NVidia, CUFFT was also faster than Intel’s MKL FFT function for CPUs [329].

- **NVidia performance primitives (NPP)** is a library of basic image, video, and signal processing functions [330].

CUDA supports different programming languages, of which C/C++ is one of the most widely used. NVidia has introduced some tools to help programmers develop GPU codes in C/C++ language, such as Nvidia Nsight for debugging, building, profiling, and tracing NVidia GPU codes with CUDA and C/C++. NVidia Nsight is available under both Windows (in the Microsoft visual studio edition [331]) and Linux (the Eclipse edition [332]).

In addition to the C/C++ programming language, NVidia GPUs can be programmed in MATLAB using the MATLAB parallel computing toolbox [333][334] (NVidia has suggested using the Tesla family for this purpose [335]). ArrayFire is another software library available for GPU programming [336]. ArrayFire acts as an application program interface (API) and simplifies programming with CUDA-capable NVidia GPUs and some other OpenCL devices.
ArrayFire functions (the list and details are available in [337]) are designed to perform calculations on arrays to speed up the process, while maintaining a simple interface for programming. There are also some other tools for accelerating MATLAB codes using GPUs (refer to [338] for a brief review and comparison).

CUDA-capable NVidia GPUs can also be programmed in NI LabView using the NI-GPU analysis toolkit [339]. In addition to the basic CUDA functions, CUBLAS and CUFFT functions can be wrapped for use in NI LabView with the NI-GPU analysis toolkit [340].

A.5.3 Development time

The development time for GPUs is shorter than for FPGAs and DSPs. CUDA and NVidia Nsight facilitate the process of developing and debugging complex GPU codes. It was estimated in a 2012 study that developing algorithms in a GPU for extracting dense optical flow, stereo and local image features will take 2 months for one post-doctoral employee, while developing the same algorithms in an FPGA will take 12 months for two post-doctoral employees [287]. Thus, in this specific application, the development time for GPUs may be 12 times faster than for FPGAs. This shorter development time results from both easier programming, and the simpler architecture of GPUs compared to FPGAs. The development time in GPUs can be made even shorter using the Matlab parallel computing toolbox or ArrayFire. However, writing optimised codes in GPUs for high-throughput algorithms in high-speed applications requires understanding of the GPU hardware architecture and optimisation techniques. Often, algorithms should be modified in a way to suit the particular GPU hardware which is chosen for implementations. Furthermore, careful memory management and data transfer between the host (PC) and the GPU card is necessary.

A.5.4 Advantages of using GPUs

Important advantages of using GPUs over other hardware accelerators include:
- GPUs are mass produced (primarily for the entertainment industry). Hence they are relatively inexpensive compared to FPGAs, and have the best processing power to price ratio among hardware accelerators [222].

- GPUs are specially designed for performing image and video processing.

- GPUs are programmed with high-level programming languages. Developing and debugging code in GPUs is faster and easier than in FPGAs.

- The PCIe interface between the GPU cards and the host PC can be easily used by programmers.

- NVidia GPUs can be programmed using NI LabView, which is an advantage for using GPUs in instrumentation projects.

- GPU technologies are rapidly advancing and, despite new technologies having higher capabilities, they are often not much more expensive.

A.5.5 Disadvantages of using GPUs

Important disadvantages of using GPUs over other hardware accelerators include:

- GPUs usually consume more power compared to FPGAs, and are thus generally unsuitable for portable systems that include complicated image processing algorithms.

- GPUs are designed for problems that have massive data parallelism. The performance of GPUs will decrease considerably if they have to wait for data, or if the processing of data is time-consuming and slow.

- The main speed bottleneck in using GPUs in PC-based systems is the data transfer time between the host PC and the GPU. A non-optimised GPU code might not help
to increase the processing speed. Thus, it is very important to minimise the GPU data access to the host PC.

- Even though some tools are available for managing the memory in NVidia GPUs (e.g. MATOG [341]), the management of memory and the choice between shared memory, local memory, global memory, constant memory, and texture memory are not straightforward for high performance applications.

- Double precision calculations are theoretically around two times slower than single precision calculations inside GPUs [342]. Although the actual speed reduction of double precision calculations is less than two times in practice (since they require less data throughput), it is important to consider this limitation before deciding about the type of calculation in GPUs.

- The development of many low-level functions in GPUs often requires using the assembly language custom codes. The GPU’s hardware is pre-structured and has a lower flexibility compared to that of FPGAs.

### A.5.6 Review of applications that use GPUs

A survey of the use of GPUs in medical image registration was published by Shams et al. [223], in 2010. They investigated a number of different registration criteria algorithms in GPUs. Some of these algorithms include (details are in [223]):

- Sum of square differences (SSD);
- Sum of absolute differences (SAD);
- Normalized cross correlation (NCC);
- Correlation coefficient;
- Gradient correlation;
- Mutual information (MI);
- Normalized mutual information;
- Correlation ratio.
Among these algorithms, SSD was the fastest in GPUs. They also investigated different optimisation methods. Some of these methods include (details are in [223]):

- Powell;
- Simplex;
- Soblex;
- Gradient descent;
- Quasi-Newton;
- Levenberg-Marquardt;
- Simulated annealing;
- Genetic.

To compare the performance of GPUs with FPGAs, Shams et al. [223] defined a normalised performance value, which was the average execution time in milliseconds for a single iteration of the optimisation algorithm and for processing 1 million voxel pairs. The average normalised performance value reported by Shams et al. [223] was higher for GPUs than FPGAs.

In 2011, Fluck et al. [222] published a related survey for 3D and 2D medical image registration on GPUs. They divided the registration transformations into rigid and non-rigid, and they investigated SSD, SAD, NCC, and MI as image registration similarity metrics for the registration criteria. Fluck et al. [222] did not carry out any performance analysis between the registration algorithms, but they summarised the strengths and weaknesses of each. They also investigated programming models and interfaces. For the programming language, they concluded that CUDA has established itself as a popular platform for image registration and other image processing tasks, while OpenCL is an emerging standard for parallel programming.

In another review paper, Eklund et al. [220] published a survey of GPU accelerated medical image processing. The algorithms were divided into basic image processing operations
(filtering, interpolation, histogram estimation, and distance transforms), and commonly used algorithms (image registration, image segmentation, and image denoising). The majority of the image registration methods investigated in their review paper was based on image intensity rather than phase-based optical flow techniques. Phase-based optical flow approaches are computationally complex, since they require using a filter bank to decompose the image into components, and for every component the temporal phase gradient needs to be calculated. For this reason, it is difficult to implement such algorithms in GPUs.

Castaño-Díez et al. [343] evaluated the performance of some image processing algorithms such as FFT, matrix algebra, geometrical operations, image reconstruction, and principal component analysis (PCA), in GPUs. They implemented these algorithms in GPUs to enable a comparison with an ordinary CPU. The typical speedup ratio of their GPU implementation compared to the CPU implementation of the same algorithm was between 10 times and 20 times. However, the speedup ratio was very dependent on the type of algorithm, and the image size. For example, the speedup ratio of the FFT algorithm using the NVidia CUFFT library was on average 15 for various image sizes, whereas it was 33 at its maximum for image sizes of approximately 1000 pixel × 1000 pixel.

Table A-8 provides a summary of some selected GPU implementations of computer vision and image processing algorithms in recent literature. The most commonly used GPUs were the NVidia GeForce series. Table A-9 lists the microarchitecture of the NVidia GeForce series, and other GPUs which were used in the literature of for implementation of the algorithms.

Table A-8: A summary of some selected implemented computer vision and image processing algorithms on GPUs in the recent literature.
<table>
<thead>
<tr>
<th>Application(s)</th>
<th>Algorithm(s) Implemented</th>
<th>Hardware (GPU) (the year of introduction)</th>
<th>Performance/ Data Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stereoscopic scene flow computation for 3D motion analysis (2011) [150]</td>
<td>• Image rectification • Residual processing • Stereo matching • Disparity map calculation</td>
<td>GTX 480 (2010)</td>
<td>20 fps for an image size of 320 pixel × 240 pixel</td>
</tr>
<tr>
<td>Stereo matching with slanted surface modelling (2011) [344]</td>
<td>• Coarse stereo matching • Disparity plane fitting • Subpixel stereo matching • Disparity map calculation</td>
<td>GTX 480 (2010)</td>
<td>8 M disparities/s with subpixel accuracy (i.e. 5 fps for an image size of 384 pixel × 288 pixel and 16 levels of disparity)</td>
</tr>
<tr>
<td>Real-Time Surface Curvature Estimation (2012) [345]</td>
<td>• Computation of principal curvatures, principal directions of curvatures, and the derivative of curvature</td>
<td>GTX 480 (2010) and Quadro FX 5800 (2008)</td>
<td>Speedup of 6 to 8-fold in a Quadro FX 5800 (only), and speedup of 18 to 20-fold in a GTX 480 (only) compared to the multithreaded CPU algorithm</td>
</tr>
<tr>
<td>Tomographic reconstruction (2012) [346]</td>
<td>• WBP and SIRT methods (the two most common methods for electron tomography reconstruction)</td>
<td>Two hybrid system of CPUs and GPUs:</td>
<td>The hybrid system could achieve speedup of 2-fold with a Tesla C2050 GPUs, and speedup of 1.5-fold with a GTX 285</td>
</tr>
<tr>
<td>Real-time 3D range video encoding and decoding (2012) [347]</td>
<td>• The Holovideo technique, which is devised from the digital fringe projection technique</td>
<td>GeForce 9400M (2008)</td>
<td>18 fps for an image size of 512 pixel × 512 pixel</td>
</tr>
<tr>
<td>Colour Stereo Matching (2013) [348]</td>
<td>• Calculation of SAD, and the arm-length-differences (ALD)</td>
<td>GTX 570 (2010)</td>
<td>The implementation could generate matching results for each pair of images in less than 100 milliseconds for an image size of 450 pixel × 357 pixel</td>
</tr>
<tr>
<td>A stereo vision system for real-time tracking (2014) [349]</td>
<td>• Stereo matching with symmetric dynamic programming stereo (SDPS) • Image rectification • SDPS disparity generation • Joint colour disparity filter • Foreground 3D reprojection • Post-processing (speckle filtering)</td>
<td>GTX 680 (2012)</td>
<td>11475 M disparities/s (i.e. 1 14 fps for 1024 pixel × 768 pixel images and 128 levels of disparity)</td>
</tr>
<tr>
<td>Large-size VHR images registration (2014) [350]</td>
<td>• Coarse registration (using SIFT and RANSAC algorithms) • Fine registration • Image rectification based on the triangulated Images</td>
<td>GTX 650 (2012)</td>
<td>The image rectification speed was increased by 16 times compared to CPU implementation</td>
</tr>
<tr>
<td>Digital volume correlation (2014) [133]</td>
<td>• Coarse search and registration using FFT and IFFT • Subpixel registration based on an optimisation process with Broyden-Fletcher-Goldfarb-Shanno algorithm</td>
<td>3 × Tesla M2090 (2012)</td>
<td>Speedup of 8-fold by using three Tesla M2090 GPUs in comparison to the CPU alone</td>
</tr>
</tbody>
</table>
Application(s) | Algorithm(s) Implemented | Hardware (GPU) (the year of introduction) | Performance/ Data Throughput
--- | --- | --- | ---
Digital image correlation (2015) [352] | • FFT and IFFT for finding the integer shift • Inverse compositional Gauss–Newton (IC-GN) algorithm for finding the sub-pixel shift | GTX 760 (2013) | Speedup of 57 to 76-fold over sequential implementation on a CPU, with the same accuracy
Fingerprint identification (2015) [353] | • Feature matching (global and local) for fingerprint images | 2 × Tesla K20 (2012) 2 × Tesla M2090 (2012) | Speedups compared to the multi-threaded CPU implementations are: 15.69 for 1 × Tesla K20 14.79 for 1 × Tesla M2090 30.49 for 2 × Tesla K20 28.71 for 2 × Tesla M2090 54.20 for all 4 GPUs

Table A-9: The microarchitecture of some of NVidia GPUs used in the literature

<table>
<thead>
<tr>
<th>NVidia GPUs</th>
<th>Microarchitecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeForce 200 series (GTX 280, GTX 285, GTX 295)</td>
<td>Tesla</td>
</tr>
<tr>
<td>GeForce 300 series</td>
<td>Tesla</td>
</tr>
<tr>
<td>Quadro FX 5000 series (FX 5800)</td>
<td>Tesla</td>
</tr>
<tr>
<td>GeForce 9 series (GeForce 9400M)</td>
<td>Tesla</td>
</tr>
<tr>
<td>Tesla C1060</td>
<td>Tesla</td>
</tr>
<tr>
<td>Tesla C2050</td>
<td>Tesla</td>
</tr>
<tr>
<td>GeForce 400 series (GTX 480)</td>
<td>Fermi</td>
</tr>
<tr>
<td>GeForce 500 series (GTX 570, GTX 580)</td>
<td>Fermi</td>
</tr>
<tr>
<td>Tesla M2090</td>
<td>Fermi</td>
</tr>
<tr>
<td>GeForce 600 series (GTX 650, GTX 660, and GTX 680)</td>
<td>Kepler</td>
</tr>
<tr>
<td>GeForce 700 series (GTX 760)</td>
<td>Kepler</td>
</tr>
<tr>
<td>Tesla K20</td>
<td>Kepler</td>
</tr>
</tbody>
</table>

In Table A-8, neither the speedup ratio, nor the data throughput, was remarkable for implementation of algorithms in GPUs with Tesla or Fermi microarchitectures (Table A-9). However, the GPUs with Kepler microarchitecture (Table A-9) showed superior performance.
For example, 11475 M disparities/s were achieved in a GTX 680 in [349], and a speedup ratio of up to 76, compared to the CPU implementation, was achieved in a GTX 760 in [352]. However, the speedup ratio of the fingerprint identification algorithm in Table A-8 was not much higher in a Tesla K20 with Kepler microarchitecture than in a Tesla M2090 with Fermi microarchitecture in [353].

A.5.7 Summary and conclusion for GPUs

In recent years, there has been increasing interest in using GPUs to perform scientific computations. GPUs are relatively inexpensive, have high processing speeds in their latest generations, and it is relatively simple to develop complicated codes for them. The development time for computer vision and image processing applications for GPUs is less than for FPGAs, and GPUs can provide a good performance in applications where no data acquisition is required.

The speedup that one can obtain by implementing the algorithms in a GPU varies depending on the algorithm type, the GPU microarchitecture, and programming techniques. However, Brodtkorb et al [354] suggested that considering the latest technologies in GPUs and CPUs, when the algorithms are not being adapted for GPUs, the performance speedup in the GPU can be approximately seven times compared to the CPU implementation [354].

Even though recent NVidia GPUs for laptops show higher performance, they are mainly designed for gaming not scientific computing. Because of this, desktop GPUs are still a more suitable option for scientific computing.

In summary, there are some applications in which GPUs are a more suitable option compared to other hardware accelerators. Some examples include:

- The programming language of GPUs is high-level, and their development time is shorter than FPGAs, hence they are more suitable for fast prototyping.
• GPUs are suitable hardware-accelerators in PC-based systems in which capturing of high throughput data is not required.

• GPUs are less expensive than FPGAs, hence GPUs are more suitable for cost-sensitive applications.

• It is usually easier to transfer codes to new hardware in GPUs than in FPGAs. GPUs are thus more suitable hardware accelerators for algorithms which need to be frequently updated.

The GPUs being used in the literature are typically selected from recent technologies at the time of publication. The codes could even be developed before the release date of the GPUs, and then quickly modified for that specific GPU. For instance, the average interval between the introduction year of a GPU and its use in a publication (the publication year), for the papers in Table A-8, was (2.4 ± 1.0) years. Note that this time includes the duration of the manuscript review process.

NVidia has a plan for producing a new generation of GPUs with integrated CPU and GPU cores based on the ARM architecture [354]. This technology will make GPUs a more suitable option for mobile applications, and will help to decrease the need for transferring data between the GPU and the CPU, which is currently one of the main performance bottlenecks for GPUs. However, those GPUs are not designed for high-performance applications.

GPUs and FPGAs are more commonly used compared to DSPs for implementing image processing and computer vision algorithms. It is thus interesting to compare implementations in an FPGA and a GPU. However, an accurate performance comparison between FPGAs, and GPUs is not practical since each application has different demands, and the performance depends on the level of programmer proficiency and the hardware being used. In most of the published papers, the comparison between FPGAs and GPUs is not performed for comparable technology levels, with GPUs usually belonging to a more recent generation compared to FPGAs. This may be a consequence of cheaper costs of GPUs over FPGAs, or their relative ease of programming. However, comparing some examples of implementation
of the same algorithm in both FPGAs and GPUs can help developers to achieve an approximate performance comparison. The next section provides a brief comparison between FPGAs and GPUs that are being used as hardware accelerators for computer vision and image processing applications.

A.6 Comparison of FPGAs and GPU for implementing image processing, and computer vision algorithms

Figure A-1 shows the search trend in the Google search engine for the word “GPU” (in blue) and the word “FPGA” (in orange), over the last 11 years (2004 to 2015). The values are extracted from Google Trends [355], and the graphs are normalised to the highest number of searches. Since 2007 GPUs have been searched for more than FPGAs. One of the most important examples of using NVidia GPUs for high performance applications in recent years, is the second fastest supercomputer in the world, named Titan [356]. Titan includes 18,688 NVidia Tesla GPUs, and has a processing power of more than $2 \times 10^{16}$ calculations per second [356].

![Figure A-1: Search trends for GPU (in blue colour) and FPGA (in orange colour)](image)
Among computer vision and image processing algorithms, stereo vision algorithms are the most common application implemented in hardware accelerators. Tippetts et al. [217] reviewed the implementation of various stereo vision algorithms in CPUs, GPUs, and FPGAs. Among the reported publications in their review, the maximum disparities/s for algorithms were 6 million in CPUs, 7,247 million in GPUs, and 6,062 million in FPGAs. Among the papers reviewed by Tippetts et al. [217], the fastest GPU and FPGA implementations were approximately 1000 times faster than the fastest CPU implementation. However, the CPU implementation was in a single core Pentium 4 CPU (introduced in 2000), while the latest FPGAs being used were Xilinx Virtex-5 and Altera Stratix-3 (both introduced in 2006), and the latest GPU was NVidia GeForce GTX 280 (GeForce 200 series, introduced in 2008 (Table A-9)). Thus, all of these hardware technologies reviewed by Tippetts et al. [217] were far from state-of-the-art, being outdated even at the publication time of Tippetts et al.’s review [217] in 2013.

Brodtkorb et al. [224] published a comprehensive review paper, and described hardware and software tools in heterogeneous computing. They summarised the differences between FPGAs and GPUs in architecture, performance, power consumption, cost, programming languages, and debugging tools. In addition, they investigated the implementation of some basic algorithms. For example, Brodtkorb et al. [224] reviewed the implementation of FFT with NVidia CUFFT library in a GTX 280 GPU (Table A-9), which they reported to be 8 times to 40 times faster than Intel’s MKL on a 3 GHz Intel Core2 Quad [224]. Unlike the review of Tippetts et al. [217], in the review by Brodtkorb et al. [224] the hardware technologies were relatively recent in 2010 when the paper was published. However, for implementation of FFT in FPGAs, even though Brodtkorb et al. [224] had introduced Xilinx and Altera IP-cores, there was no performance comparison in the FPGA section of that paper.

Gao et al. [219] reviewed the applications of parallel computing in experimental mechanics and optical measurements using hardware accelerators (DSPs, FPGAs and GPUs). They mentioned that it is difficult to increase the speed of such algorithms in CPU-based systems, whereas using data parallelism techniques makes them suitable choices for parallel hardware,
such as FPGAs and GPUs. As an example, they reported a 47 times speedup using an NVidia GeForce 6600 GPU in comparison to a Pentium 4 CPU. However, both of these were old technologies even at the publication date of the review paper in 2012 (GeForce 6600 was introduced in 2004). In total, Gao et al. [219] reviewed 38 papers from 2008 to 2012, of which 42% used GPUs, 29% used FPGAs, and 29% used others. Based on their survey, GPUs were the most popular hardware accelerators in experimental mechanics and optical measurements.

Asano et al. [357] reviewed the performance of FPGAs, GPUs, and CPUs in three different applications in image processing. The algorithms they investigated included two-dimensional filters, stereo-vision (feature matching, and 3D projection), and $k$-means clustering [357]. The GPU used in their study was an NVidia GTX 280 (Table A-9), and the FPGA they used was a Xilinx Virtex-4 (introduced in 2005 (Table A-3)). In their comparisons, the GPU performed better for 2D filter sizes less than 11, and the FPGA performed better for stereo vision (feature matching, and 3D projection), and $k$-means clustering. Asano et al. [357] concluded that the GPU is preferable for simple tasks where pixels can be processed independently, whereas for more sophisticated algorithms FPGAs perform better.

Fowers et al. (2012) [218] compared the performance and power consumption of FPGAs, GPUs, and multicore processors for sliding-window applications. The GPU and FPGA used for the sliding-window applications in [218] were a GTX 295 (introduced in 2008 (Table A-9)), and an Altera Stratix-3 (introduced in 2006 (Table A-4)). They investigated SAD, 2D convolution, and correntropy [358] (i.e. a measure of similarity based on information theoretic learning) as sliding-window applications [218]. In their implementations, the FPGA implementation gave the best performance for SAD, and was the only device which was able to do real-time computations up to $50 \times 50$ kernel size. The FPGA implementation could also reach approximately 80 fps for a 720p image size, while the frame rate was approximately 10 fps for the GPU implementation, and less than 1 fps for the CPU sequential C++ implementation.
Fowers et al. [218] also analysed the 2D convolution performance for the FPGA and GPU implementations in the time domain, as well as the GPU implementation in the frequency domain using FFT [218]. For the 2D convolution, the frequency domain GPU code (GPU-FFT) showed the best performance where, for an image size of 1280 pixel × 720 pixel and a kernel size of 25, a frame rate of approximately 120 fps was achieved in the GPU, whereas the FPGA implementation in the time domain could achieve 80 fps, and the GPU implementation in the time domain could achieve 60 fps. However, their comparison of the frequency domain implementation in GPU and the time domain implementations in FPGA is unfair, since implementation of 2D convolution in frequency domain for large kernel sizes is faster than time domain. As can be seen, the FPGA implementation of Fowers et al. for 2D convolution in the time domain could achieve 20 fps more than the one implemented in the GPU in the time domain.

Fowers et al. [218] improved their methodology for comparing FPGA and GPU implementations in their next paper [359] by adding the FPGA frequency domain implementation of 1D convolution to their comparisons. The FPGA in their study was a Gidel ProcSTAR III board, which contains four Altera Stratix-3 FPGAs, and the GPU was an NVidia GeForce 295 GTX (Table A-9). They considered two scenarios in the FPGA performance analysis: first where the data is sent to the FPGA board via PCIe; and the second was a standalone FPGA, where the data source was directly connected to the FPGA (similar to the embedded systems). From these two scenarios, the standalone FPGA showed better performance for the implementation of 1D convolution in the frequency domain compared to the PCIe FPGA board. For instance, when performance was compared to the frequency domain implementation in the CPU, the speedup ratios of the standalone FPGA and the PCIe FPGA were approximately 10 and 5, respectively. The standalone FPGA could perform better than the GPU for large signal sizes with small convolution kernel sizes. However, in other cases (such as small or large signal sizes with large convolution kernel sizes) the GPU had better performance.
Pauwels et al. [287] compared the performance of FPGAs and GPUs for real-time phase-based optical flow, stereo, and local image feature matching [287]. The FPGAs they used were Xilinx Virtex-4 and Virtex-5, and the GPUs were GeForce GTX 280 (Table A-9) and GTX 580 (introduced in 2010 (Table A-9)). A comparison was performed for accuracy, speed, power consumption, cost, and design time. They found that GPUs and FPGAs were suitable for different applications. For example, FPGAs were more suitable than GPUs for local feature matching, median filtering, and embedded platforms. FPGAs also had advantages over GPUs for their low power consumption, and level of flexibility. On the other hand, Pauwels et al. [287] concluded that GPUs were more suitable for image warping and applying spatial filters on image pyramids. In their study they suggested that the advantages of GPUs over FPGAs are their low cost and short design time.

In another comparison between FPGAs and GPUs, Cortie et al. [309] compared an FPGA implementation in Xilinx Spartan-3 for parallel convolutions to an Intel Xeon CPU, and an NVidia Tesla C1060 GPU (introduced in 2008 (Table A-9)) [309] (refer to Table A-7 for more details about FPGA implementation). They showed that, although they could reach higher speeds with both the FPGA and GPU in comparison to the CPU, the FPGA was faster in smaller systems and the GPU performed better in larger systems.

Very few papers have evaluated the performance of the same technology level for FPGAs and GPUs. In one recent paper, Birk et al. [360] performed an efficiency comparison for algorithms in 3D ultrasound computer tomography in 40 nm and 28 nm fabrication technology generations of FPGAs and GPUs. The hardware accelerators from the 40 nm fabrication technology were Xilinx Virtex-6 (Table A-3) for the FPGA and NVidia GTX 580 (Table A-9) for the GPU. The hardware accelerators from the 28 nm fabrication technology were Xilinx Virtex-7 (Table A-3) for the FPGA and NVidia Tesla K20 (Table A-9) for the GPU. This selection of hardware accelerators from the same fabrication technology enabled a rigorous and fair comparison between FPGAs and GPUs. Birk et al. concluded that the GPU and the FPGA using the 40 nm fabrication technology can provide similar performance
and efficiency, if the power consumption is not considered. In contrast, for the 28 nm fabrication technology, the FPGA implementation had a speedup of 1.86 compared to its GPU counterpart. This implies that FPGAs have substantially improved in their latest generation. One of the main differences between FPGAs and GPUs is in using multiple hardware accelerators at the same time to implement complex algorithms. NVidia GPUs use PCIe as their interface to PCs, and the interface is fully developed by NVidia. It is thus simple to use multiple GPU cards at the same time in one PC. In contrast, developing PCIe interfaces for multiple FPGA cards is difficult. Using multiple GPU cards is thus more common compared to multiple FPGA boards when implementing complex algorithms (some examples of using multiple GPUs were introduced in Table A-8).

A.7 Summary and conclusions

In this review, practical information was provided for selecting suitable hardware accelerators for computer vision and image processing algorithms. The hardware architectures of the most recent DSPs, FPGAs, and GPUs were discussed, and the important features of these hardware accelerators for computer vision and image processing algorithms. For each hardware accelerator, available tools and utilities, development time, advantages, and disadvantages were discussed in an attempt to help developers to choose the most appropriate hardware for their application. Examples from the literature were reviewed in separate sections for applications of DSPs, FPGAs, and GPUs in accelerating computer vision and image processing algorithms. Details of the implemented algorithm, the hardware type, and the hardware introduction year were included. Among hardware accelerators, FPGAs and GPUs are widely used in computer vision and image processing applications. Thus, a specific comparison of the performances of FPGAs and GPUs was provided.

The average time between the introduction of an FPGA and its publication year was (5.3 ± 1.9) years in 19 publications (Tables 6 and 7), and (2.4 ± 1.0) years for GPUs in 12 publications (Table A-8). This illustrates that the FPGA technologies being used in recent publications were on average twice as old as the GPU technologies.
Among the hardware accelerators, DSPs are the least commonly used for computer vision and image processing tasks, and GPUs are the most commonly used. GPUs are suitable hardware accelerators for applications that need short development times or fast prototyping, require good processing speed, or need significant on-board memory. In comparison, FPGAs are the most suitable hardware accelerators for algorithms that require significant processing speed, need to be used in low-power applications, or include customised algorithms. DSPs are suitable hardware accelerators for very low-power applications, or cheap portable devices.

Even though GPUs are frequently used for computer vision and image processing algorithms, and it is often believed that they are the most suitable choice, recent developments have made FPGAs an attractive option for many applications. The tools and utilities from Xilinx and Altera have simplified the code development process, and shortened the development time when using FPGAs. Furthermore, the latest FPGA technologies provide a good level of performance.

Processing speed is an important factor in most computer vision and image processing applications, and hardware technologies are rapidly growing to address this demand. Selection of a suitable hardware accelerator could thus have a great impact on the performance of the system. In this review, practical information was provided about the state-of-the-art hardware accelerators to assist researchers and developers in selecting suitable hardware accelerators for their specific applications.
Appendix B: FPGA Implementation of 2D Cross-Correlation

Portions of this appendix were published in:


B.1 Abstract

Two dimensional (2D) and three dimensional (3D) deformation measurement have been used in various industrial and medical applications. While often a large number of images are taken to perform dynamic 2D or 3D deformation measurements, processing this amount of data is very time-consuming. A significant portion of the computation time for tracking the deformations is spent performing cross-correlations (CC) of images. Computation time can be reduced by parallel processing of the 2D CC. This appendix describes a parallel implementation of 2D CC in field programmable gate arrays (FPGAs) in order to increase the processing speed compared to its implementation on a central processing unit (CPU).

A variable size 2D CC was implemented in the Xilinx Virtex-6 LX240T FPGA using the Xilinx System Generator tool. A hierarchical approach was proposed for finding the CC peak to efficiently use this method for different image sizes. Furthermore, in this design, FPGA
random access memory (RAM) blocks were used instead of shift registers to lower the resource requirements compared to other FPGA implementations.

Results indicated that better than 200 times speed up was achieved using this design compared to a CPU implementation on an Intel Xeon E5620 CPU (2.4 GHz clock speed, 4 cores and 8 threads) and 12 GB double data rate 3 (DDR3) RAM, and 190 times speed up was achieved in comparison to an NVidia GForce GT 525M graphics processing unit (GPU) [361].

B.2 Introduction

The CC algorithm is a widely recognised approach to determine the displacements between two images and tracking changes [119]. However, the large number of multiplications and accumulations has made 2D CC a computationally expensive algorithm. Equation B.1 shows the computations required to calculate 2D CC of an \( M \times N \) pixel image \( (I_1) \) and a \( P \times Q \) pixel image \( (I_2) \).

\[
CC (k, l) = \sum_{m=1}^{M} \sum_{n=1}^{N} I_1(m, n) \times \overline{I_2(m + k, n + l)} \quad \left( - (P - 1) \leq k \leq M - 1, \quad - (Q - 1) \leq l \leq N - 1 \right)
\]

where the bar over \( I_2 \) denotes complex conjugation.

In many applications, real-time or fast systems are needed to track the deformations dynamically. FPGAs and GPUs are both good candidates for real-time systems, as they offer high speed parallel processing capabilities, and thus have been used for real-time image processing tasks [307, 343, 362]. The performance evaluations of FPGAs and GPUs are highly dependent on the hardware choice and the application. However, FPGAs have the advantage of architectural flexibility that can be used to design efficient algorithms for a specific application. To take advantage of this flexibility, an FPGA was selected for a parallel implementation of a variable size 2D CC. Moreover, a hierarchical approach was proposed for finding the CC peak in large image sizes to increase the speed of calculating 2D CC for real-time applications.
The FPGA implementation is described in Section B.3. In Section B.4, a detailed analysis of the design is provided, and speed performance is compared to the tested CPU and GPU implementations. Section B.5 discusses and concludes this study.

B.3 Method

The Xilinx System Generator tool (SysGen)[363] was used to implement the CC algorithm. This tool has been designed for the bit- and cycle-accurate implementation of signal and image processing algorithms in Xilinx FPGAs, and has helped to reduce considerably the implementation time of such algorithms. Furthermore, this tool enables performing software-hardware co-simulation [363]. The spatial domain implementation of the 2D CC in FPGAs is more efficient than its frequency domain implementation in resource allocation, especially for small image sizes [364]. For this reason, the spatial domain implementation was chosen for the FPGA implementation of 2D CC.

Huber et al. [365] implemented one dimensional (1D) autocorrelation based on asynchronous delay elements and counters. This technique uses delay lines to compute correlation. The use of a large number of shift registers is a common approach for implementing CC [365–367]. However, instead of using a large number of shift registers to implement a 2D CC, Xilinx FPGA block RAMs were used to store the data. The use of FPGA block RAMs considerably reduces the hardware resources occupied by the implementation. Furthermore, a novel hierarchical approach has also been suggested to efficiently use our method on images of various sizes, and to take advantage of the parallel processing architecture of the FPGA.

The implementation of this design is simulated based on a Xilinx ML605 board that contains the Xilinx Virtex-6 LX240T FPGA hosted on PCIe interface [368]. This board was chosen since Virtex-6 and PCIe are suitable for image processing applications and can represent a realistic cost-efficient choice for this application. The width of images can be varied
from 1 pixel to 32 pixel, and the image height can be varied from 1 pixel to 512 pixel. Two 512 pixel × 512 pixel single precision integer images (8-bit), and 32 pixel × 32 pixel subimages were selected for the 2D CC tests here. Since the memory size in the Xilinx Virtex-6 LX240T is 36 bit × 512 row [369], the size of 512 pixel for the height of the images provides the optimum FPGA block-RAM allocation.

The algorithm was designed in a block-based format, and was implemented using Xilinx SysGen. The implementation of block-based algorithms is easier and faster using SysGen than VHDL coding. This tool has previously been used for implementing some real-time algorithms [370, 371].

The purpose of performing CC is to find the best match between two images. For example, the CC algorithm was used in [119] to measure the displacement, and track material points between initial and deformed subimages of grey scale images. The starting point of the CC algorithm was from a position where the subimages of the deformed image had a complete overlap (were aligned) with the initial image, and the end point was when they reached to the end of the initial image (Figure B-2).

![Figure B-2: The implemented CC algorithm for two images. Dashed lines show the subimage and solid lines show the initial image. The first position of the algorithm was where the top of the two images were aligned, and the last position was where they were aligned at the bottom.](image)

The multiplication part of the CC algorithm was applied in parallel to all of the intensity values of every row of the images. The output of the multiplication was stored in an accumulator for all of the rows of the subimage (corresponding to the image height). This
accumulator stores the output of the $\Sigma$ operator in Equation B.1, which quantifies the similarity between the subimages of the deformed image and the initial image.

After calculating the value of CC for one row, the subimage was moved one pixel down and the value of the accumulator was reset to calculate the value of the CC for the new position. Figure B-3 illustrates this concept.

Figure B-3: Calculation of the CC value for each relative position of the subset and the initial images

The design implemented in SysGen had three main stages:

- loading and initialisation: the intensity values of the two images was loaded and stored in block RAMs. The initial parameters were also set in this stage;
- synchronisation: the outputs of each set of block RAMs were synchronised to have a correct pixel correspondences between the subimages of the deformed and initial images for the specified position of the subimages;
- calculating the CC value: at each specified position of subimages, the intensities of the corresponding pixels were multiplied together, and multiplied intensities were summed to calculate the CC value for that position.

These three stages are described in more detail in the following three sections.
B.3.1 Loading and initialisation

Four 64 bit \( \times \) 512 row block RAMs were used to store the intensity data of the initial and deformed images. Since the intensity value of each pixel was 8 bit, 64 bit of block RAMs could store the intensities of 8 image pixels.

The 64 bit width of block RAMs was compatible with the standard width used for other interfaces which are typically available on the FPGA boards, such as the peripheral component interconnect express (PCIe) interface, and DDR RAMs. The intensity values of 8 pixels were concatenated to form a 64 bit data block, as shown in Figure B-4.

![Figure B-4: Concatenating the intensity values for 8 pixels to form a 64 bit digit.](image)

The heights of the initial and the subimage were set during this stage.

B.3.2 Synchronisation

The correct corresponding pixels were read from the FPGA block RAMs to have pixel correspondences between the two subimages. The flow diagram shown in Figure B-5 was implemented to achieve this and to synchronise the data read process.
Figure B-5: The flow diagram of the procedure for synchronisation between reading the image data of two subimages.

The whole procedure in Figure B-5 comprises of two main loops, one for the inner $\Sigma$ and one for the outer $\Sigma$ in Equation B.1. The variable “address increase” is used to control the upper limits of the outer $\Sigma$. Hence, address increase was set to zero at the starting position of the two subimages, and was automatically incremented by one when calculation of the CC
was completed at a specified position of the two subimages. This ensures that the calculation of the CC value restarts at each new position of the deformed image subimage on the initial image. To have a variable image sizes, the criteria for both loops were checked based on the sizes, which were set in the initialisation stage. Several flags were used in the process of checking and controlling these conditions, to loop through all the lines of data, and to synchronise the parallel computations.

B.3.3 Calculating the CC value

The corresponding pixel positions of two images were multiplied to calculate the CC value. Before multiplying the pixels, a register with an enable port was placed to control sending the appropriate data to the multiplication unit. The use of this register enabled achieving the maximum possible calculation speed, as the multiplication was synchronised with the main clock cycles. Three main clock cycles (delays) were chosen for the multiplication unit of the pipelined operation. Adjacent to the multiplication unit was an accumulator to store the CC data of the rows of subimages. The accumulator was implemented using DSP48E1 modules [372] to increase the speed and reduce the use of logic gates compared to using FPGA logic cells. The multiplication unit and the accumulator are shown in Figure B-6.

![Figure B-6: The multiplication unit and the accumulator for calculating CC values](image)

A reset signal was sent to the accumulator to reset its value at each new placement of two subimages. The reset signal of the accumulator was generated based on the size of the subimage set in the initialisation state. Therefore, in this implementation, the subimage size could be selected between 1 pixel to 512 pixel (Figure B-2).
To synchronise the data valid signals (enable signal) with the data, the delays of any operation on the data should be considered before applying the data valid signals to the enable port of the logic gates. To address this, the enable signal at the end of this stage was delayed by the same time as was observed in the multiplication and accumulation (Figure B-7). 32 multiplication and 32 accumulation units were required to perform parallel multiplication for every position of subimages, and to calculate one CC value.

![Diagram showing registers to synchronise the output data and the data valid signal.]

**Figure B-7: Using registers to synchronise the output data and the data valid signal**

At the last step of this stage, the outputs of the CC for each position of two subimages were summed, and the final CC value was calculated. The total number of CC values at each position was equal to the height of the subimage. Pipelined addition units operating at the main clock cycle were used in summing these CC values. The data valid signal was applied to the output of the addition unit using the “register” at the end of this process (Figure B-7). The CC value could thus be sent to any other module, or could be written to an external or internal RAM using the data valid signal.

**B.3.4 A hierarchical approach to calculate CC**

Even though the implementation described in Sections B to B.7.3 calculates the CC value in parallel, it is designed for 32 pixel × 512 pixel images. However, feature matching and displacement computation usually involves finding the peak CC value over a large initial image.
To achieve the best performance in finding the peak of CC using this implementation for image sizes larger than 32 pixel × 512 pixel, a hierarchical approach with a number of CC modules in parallel can be used. This hierarchical approach is described below for the width size of 256 pixel × 512 pixel images, as an example.

- At the first step, the initial image was divided into a number of 32 pixel width slices with half size overlap (16 slices in this example), and the CC value was found for all of the slices in parallel (Figure B-8).

- In the next step, two neighbour slices that had the highest CC values were chosen to form a new image data block (i.e. 32 + 16 = 48 pixel), and the procedure in the first step was repeated for this new data block, but with a larger overlap.

- Step two was repeated until the size of the data block was smaller than (32 + number of slices) (i.e. 48 pixel for this example). At this step, the maximum CC value was found simultaneously in parallel modules of CC (because the shift between modules will be one pixel, and the maximum value (peak) of the CC can be found with one pixel resolution in parallel modules).

**16 pixel (Overlapped area)**

![Diagram of 16 pixel overlapped area](image)

**Figure B-8: Two sample neighbour slices in the hierarchical approach**
B.4 Results

The CC results were tested and validated using inputs generated from a Matlab program. To test the algorithm, counters with values from 0 to 512 were stored in 8 block RAMs to be used as the inputs. The height of the subimage of the displaced image was 32 pixel, and the height of the initial image was 512 pixel. Part of the final result is shown in Figure B-9. The output demonstrates a complete synchronisation between the data and data valid signals (Figure B-9). The value of the CC for the next row of data was shifted up in the output at each new row because the input data values were counters stored in the rows.

![Diagram](image)

**Figure B-9: Part of the final result. The data and data valid signal were synchronised, and the CC value was shifted up for each new row.**

The Xilinx ISE software [373] was used to provide more details of the implementation. The SysGen output was imported into a Xilinx ISE project to extract information about the details of the implementation. The device utilisation summary of the ISE project showed that there was a total of 543 occupied slices for this design (approximately 1% of the 37,680 available slices in the Virtex-6 LX240T). However, since the number of logic gates inside the FPGA slices varies across FPGAs, the utilisation report of the Xilinx PlanAhead [272] was also generated to provide more detail (Figure B-10).
The DSP48E1 blocks occupied in this implementation (Figure B-10) were all used for calculating the CC values. For each pair of pixels (from the two images), one DSP48E1 block was used for multiplication, and one was used for accumulation. Therefore, 64 DSP48E1 blocks were required for 32 pixels (width of images) (Figure B-10).

This design used eight 64 bit × 512 row memory units. Each memory unit occupies 2 FPGA block RAMs, which is the optimum number for the width of 64 bit. Hence, in total, 16 FPGA block RAMs were used for the implementation of this design (Figure B-10).

The maximum achievable frequency of this design was reported to be 252.8 MHz (minimum period: 3.95 ns) by the synthesis report of the ISE project. The resources occupied by this design were less than other implementations in the FPGA for the same size of the CC [365–367]. As the FPGA type of the design will affect the maximum possible frequency, a reliable comparison of the maximum frequency of this implementation with other implementations was not practicable. However, the maximum frequency of this design (i.e. 252 MHz) was around half the speed of DSP48E1 blocks [23].

The total number of delays in this implementation was 45 main clock cycles. Therefore, for the subimage size of 32 pixel and initial image of size 256 pixel, it will take
(45 + (256 - 32) \times 32) = 7213 \text{ main clock cycles to compute all the CC values. The main clock frequency can be chosen up to 252 MHz according to the ISE synthesis report. Hence, the total time to calculate the CC value for these two images will be } 7213/252 \text{ MHz} = 28.6\mu s. \text{ However, this execution time is not exact because the time required to find the maximum value of the CC and loading the data for a new set of CC calculations has not been considered.}

The total time spent to find the correlation peak for 32 pixel \times 32 pixel subimages of 256 pixel \times 512 pixel images using the hierarchical method described in Section B.3 was approximately } 3 \times 28.6\mu s = 85.8\mu s.

The execution time of this implementation was compared to an equivalent Matlab program computing CC for 32 pixel subimages of 256 pixel \times 512 pixel images using the \textit{conv2} command [374]. The Matlab program execution time was 19.3 ms in an Intel Xeon E5620 CPU (2.4 GHz clock speed, 4 cores and 8 threads) using 12 GB DDR3 RAM — a rate that is approximately 225 times slower than the minimum time required for the FPGA implementation (85.8\mu s). Nevertheless, for larger images, the ratio of FPGA-to-CPU execution time would decrease, as more parallel modules can be used in the FPGA implementation. Another advantage of using the FPGA over the CPU is that the whole image can be divided into smaller regions of interest (ROIs) for which the CC can be calculated in parallel for all the ROIs and subimages.

In addition to the CPU implementation, parallel computing toolbox of Matlab [375] was used to implement the Matlab’s \textit{conv2} command in parallel in an NVidia GForce GT 525M GPU. The execution time in this GPU was 16.3 ms, which is faster than the CPU, but still 190 times slower than the FPGA implementation.
B.5 Discussion and conclusion

In this appendix, a new method for implementation of 2D CC of variable size in FPGAs was proposed. The utilisation report of this implementation (Figure B-10) showed this approach is well-designed for resource allocation, and occupies fewer resources in comparison to FPGA designs that use shift registers [365–367]. The utilisation report also showed that this implementation uses less than 1 % of the available registers and LUTs of the Virtex-6 LX240T. The large number of unoccupied slices will make it possible to implement many CC calculation modules at the same time.

The use of SysGen for this design will help with code changes for other configurations, import data to this algorithm in Matlab for debugging, and perform hardware and software co-simulation.

Several factors contribute to the differences between the execution times of 2D CC implementations in the FPGA, GPU, and CPU of this study. The parallel implementation of the CC algorithm and the hierarchical approach (Section B.3) reduced the execution time of the 2D CC in FPGA. Nevertheless, the time needed for transferring the data to the FPGA board was not included in the time estimations. One way to reduce the data load time in the FPGA board is to use DDR RAMs. Only one core of the CPU was used in computation of CC in the CPU in this study. Therefore, the CPU has not taken advantage of multi-cores and multi-threads. The comparison against the GPU will be more meaningful if a more powerful GPU, with more cores, is used for the GPU implementation.

The CC algorithm is an essential part of many 2D and 3D image registration methods, such as the methods described in Chapter 2 and Chapter 6. The FPGA implementation of CC in this appendix illustrates that hardware implementation of this algorithm can significantly decrease the computation time of image registration methods, making it possible to use these algorithms in computationally demanding applications, or applications that require fast computations.


86. Grédiac, M., Sur, F.: Effect of sensor noise on the resolution and spatial resolution of displacement and
strain maps estimated with the grid method. Strain. 1–27 (2013).


99. Fan, Z., Samuel, Q., Ilana, B.S., William, N., Allison, R.P.: Sensory substitution using 3-Degree-of-


   #findchessboardcorners.

   #stereocalibrate.


204. Schwiegerling, J., Greivenkamp, J.E., Miller, J.M.: Representation of videokeratoscopic height data with


Suite HLx Editions.


342. NVIDIA: Cuda C Programming Guide.


351. Akgün, D., Erdoğmuş, P.: GPU accelerated training of image convolution filter weights using genetic


357. Asano, S., Maruyama, T., Yamaguchi, Y.: Performance Comparison of FPGA, GPU and CPU in Image


FPGAs and GPUs in the 40 and 28nm generations for algorithms in 3D ultrasound computer
313


