http://researchspace.auckland.ac.nz

ResearchSpace@Auckland

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 recognise 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.

To request permissions please use the Feedback form on our webpage.
http://researchspace.auckland.ac.nz/feedback

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.
Digital Surface Curvature

PhD Thesis
by John Rugis

Submitted in Partial Fulfillment of the Requirements of the
Degree of Doctor of Philosophy of Science

Department of Computer Science
The University of Auckland, New Zealand
April 2008
Abstract

The theoretical basis for this thesis can be found in the subject of differential geometry where both line and surface curvature is a core feature. We begin with a review of curvature basics, establish notational conventions, and contribute new results (on n-cuts) which are of importance for this thesis. A new scale invariant curvature measure is presented.

Even though curvature of continuous smooth lines and surfaces is a well-defined property, when working with digital surfaces, curvature can only be estimated. We review the nature of digitized surfaces and present a number of curvature estimators, one of which (the 3-cut mean estimator) is new.

We also develop an estimator for our new scale invariant curvature measure, and apply it to digital surfaces. Surface curvature maps are defined and examples are presented. A number of curvature visualization examples are provided.

In practical applications, the noise present in digital surfaces usually precludes the possibility of direct curvature calculation. We address this noise problem with solutions including a new 2.5D filter.

Combining techniques, we introduce a data processing pipeline designed to generate surface registration markers which can be used to identify correspondences between multiple surfaces. We present a method (projecting curvature maps) in which high resolution detail is merged with a simplified mesh model for visualization purposes.

Finally, we present the results of experiments (using texture projection merging and image processing assisted physical measurement) in which we have identified, characterized, and produced visualizations of selected fine surface detail from a digitization of Michelangelo’s David statue.
Acknowledgements

The author would like to acknowledge and thank:

- Professor Reinhard Klette, for opening the door to greater possibilities,
- Professor Vladimir Kovalevski, for introducing me to digital geometry,
- Dr. Andrew Chalmers, for sharing his knowledge and experience,
- The Department of Electrical and Computer Engineering at Manukau Institute of Technology, for research and study support,
- The Department of Computer Science at the University of Auckland, for conference travel support,
- Stanford University, for access to The Digital Michelangelo Project data-set.
To Anna,

Beauty,

and Truth.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>v</td>
</tr>
<tr>
<td>1 Thesis Introduction</td>
<td>3</td>
</tr>
<tr>
<td><strong>Part 1 - Theory</strong></td>
<td>7</td>
</tr>
<tr>
<td>2 Curvature and Differential Geometry</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Curves and Arcs</td>
<td>7</td>
</tr>
<tr>
<td>2.1.1 Curvature of Plane Curves</td>
<td>10</td>
</tr>
<tr>
<td>2.1.2 Curvature and Planar Regions</td>
<td>13</td>
</tr>
<tr>
<td>2.2 Surfaces and Surface Patches</td>
<td>15</td>
</tr>
<tr>
<td>2.2.1 Surface Curvature and Solid Regions</td>
<td>17</td>
</tr>
<tr>
<td>2.2.2 Surface Curvature Computation</td>
<td>25</td>
</tr>
<tr>
<td>2.3 Scale Invariant Curvature</td>
<td>26</td>
</tr>
<tr>
<td>2.3.1 Geometric Invariants</td>
<td>27</td>
</tr>
<tr>
<td>2.3.2 A Scale Invariant Curvature Measure</td>
<td>28</td>
</tr>
<tr>
<td>2.3.3 The Similarity Curvature Space</td>
<td>29</td>
</tr>
<tr>
<td>2.4 Curvature Visualization</td>
<td>30</td>
</tr>
<tr>
<td>3 Digital Surface Curvature</td>
<td>39</td>
</tr>
<tr>
<td>3.1 Digitized Surfaces</td>
<td>39</td>
</tr>
<tr>
<td>3.2 Curvature Estimators</td>
<td>41</td>
</tr>
<tr>
<td>3.2.1 Triangle Umbrella</td>
<td>42</td>
</tr>
<tr>
<td>3.2.2 Surface Cut</td>
<td>47</td>
</tr>
</tbody>
</table>
Contents

3.3 Estimators Summary .............................................. 51

Part 2 - Applications .............................................. 55

4 Similarity Curvature Experiments ............................... 55
  4.1 Introduction .................................................... 55
    4.1.1 Curvatures .................................................. 55
  4.2 A Similarity Curvature Estimator ............................. 56
  4.3 Experiments ..................................................... 57
    4.3.1 Similarity Curvature Histograms ......................... 58
    4.3.2 Shading Coded Similarity Curvature ....................... 61
    4.3.3 3D Object Detection ....................................... 63
  4.4 Conclusion ..................................................... 63

5 Curvature Maps for Surface Analysis ............................ 65
  5.1 Introduction .................................................... 66
  5.2 The David Data-Set ............................................. 67
    5.2.1 Data Structures ............................................ 68
    5.2.2 Data Geometry ............................................. 68
  5.3 Curvature Maps .................................................. 71
    5.3.1 Surface Curvature ........................................... 71
    5.3.2 Results ..................................................... 73
  5.4 Conclusion ..................................................... 73

6 2.5D Filtering of Noisy Data ....................................... 77
  6.1 Introduction .................................................... 77
  6.2 Curvature Maps .................................................. 78
  6.3 Curvature Noise Filtering ..................................... 79
    6.3.1 2D Gaussian Filter ......................................... 79
    6.3.2 2.5D Gaussian Filter ...................................... 80
  6.4 Experiments ..................................................... 82
  6.5 Conclusion and Further Work .................................. 86

7 Surface Registration Markers ...................................... 89
  7.1 Introduction .................................................... 89
    7.1.1 Curvature and Curvature Estimation ....................... 90
    7.1.2 Curvature Maps, Filtering and Segmentation ............. 91
  7.2 Local Feature Identification .................................. 93
  7.3 Performance Evaluation ....................................... 94
Contents

7.3.1 Synthetic Data ........................................... 94
7.3.2 Performance Measures .................................... 95
7.3.3 Feature Identification Evaluation ......................... 96
7.4 Experiments and the David Dataset ........................ 97
  7.4.1 Data Structure ........................................... 99
  7.4.2 Curvature Estimators, Filtering and Segmentation ...... 100
  7.4.3 Feature Identification ..................................... 103
  7.4.4 Some Feature Mapping Results .......................... 104
7.5 Conclusions .................................................. 105

8 Projecting Surface Curvature .......................... 107
  8.1 Introduction ............................................... 107
  8.2 The Processing Pipeline .................................... 110
  8.3 Seamless Projection ........................................ 113
  8.4 Experiments .................................................. 114

9 Identifying and Visualizing Surface Detail ........... 119
  9.1 Introduction ............................................... 119
  9.2 Processing Tools and Pipeline ......................... 120
  9.3 Characterizing Chisel Marks ............................. 123
  9.4 Summary and Conclusions .................................. 128

Appendices .................................................. 131

A The Digital Michelangelo Project ..................... 131
  A.1 The David Data-Set ...................................... 132

B Publications .................................................. 139

C CD-ROM ....................................................... 141

Bibliography .................................................. 143
Chapter 1

Thesis Introduction

Taken most broadly, this thesis combines ideas from the disciplines of differential geometry and 3D computer graphics. The core concepts in differential geometry date back to original work by Gauss in the early 19th century, see, for example (Pesic 2005), whereas, of course, computer graphics is a product of the late 20th century.

In this thesis, more specifically, we will draw on ideas from disciplines including topology (e.g. books (Carlson 2001) and (Crossley 2005)), set theory (e.g. books (Shen and Verschagin 2002) and (Levy 2002)), complex analysis (e.g. books (Stewart and Tall 1983) and (Needham 1997)), computational geometry (e.g. books (de Berg et al. 2008) and (Klette and Rosenfeld 2004)), and image analysis (e.g. books (Soille 2003) and (Umbaugh 2005)).

We make extensive use of concepts and data structures from computer graphics (e.g. books (Hill and Kelley 2007), (Lengyel 2004), and (Foley et al. 2002)), especially in computations. Visualization plays an important part throughout this thesis.

Part 1 covers theory, beginning with a two section summary of differential geometry and surface curvature. The next two sections introduce a new scale invariant curvature measure as well as curvature visualization schemes. The following chapter defines digital surface curvature. Sections in this chapter discuss surface digitization and curvature estimators.

Part 2 contains six chapters dedicated to digital surface curvature applications. The first of these chapters reports on experiments using the previously defined scale invariant curvature. The remaining chapters each use data from The Digital Michelangelo Project (which is described in the Appendix). These chapters introduce curvature maps (for curvature visualization), explore and solve problems associated with noisy data, demonstrate a feature based scan registration method, present a curvature map projection technique, and report the results of work identifying and characterizing fine surface detail.
In writing this thesis, a challenge encountered, because of synthesizing multiple disciplines, was differences in standard notation, sometimes subtle, but none the less significant. In this thesis, we choose to employ vector notation and homogeneous coordinates\(^1\) for both points and directions. Point and directions are thus instances of an abstract vector class in which the fourth coordinate for points is one, and the forth coordinate for directions is zero.

We assign a right-handed \(y\)-up orthogonal coordinate system to 3D space as shown in Figure 1.1. By convention, the \(ijk\) unit basis vectors align with the positive direction of the \(xyz\) axes respectively.

\(\text{I hope that this thesis is as enjoyable to read as it was to write.}\)

\textit{John Rugis}\br
\emph{Maraetai, New Zealand 2008.}\n
\(^1\text{The use of homogeneous coordinates facilitates the general use of transformation matrices.}\)
Part 1
Theory
Chapter 2

Curvature and Differential Geometry

A discussion of both line and surface curvature is a core feature in the subject of differential geometry. See, for example, books (Oprea 2007), (Gray 1998), (Davies and Samuels 1996), and (do Carmo 1976). Studies on surface curvature can be traced back to original work by Gauss; see, for example, (Pesic 2005). This chapter begins with a review of curvature basics, establishes notational conventions, and adds a few new results (e.g., on n-cuts) which are of importance for this thesis. The chapter continues with a new scale invariant curvature measure and concludes with a presentation of some curvature visualization examples.

2.1 Curves and Arcs

We consider closed simple curves in 3D space. Such curves can be specified parametrically as a set of points

\[
A = \left\{ \mathbf{p}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} : t_{\text{min}} \leq t < t_{\text{max}} \right\}
\]

(2.1)

where \( p(t) \) is a continuous function, \( p(t_1) \neq p(t_2) \) for all \( t_1 \neq t_2 \), and \( \lim_{t \to t_{\text{max}}} p(t) = p(t_{\text{min}}) \). A point in the curve can be referred to as point \( \mathbf{p} \).

Example (a) in Figure 2.1 is a closed, simple curve. Note that the points in (a) form a closed loop. Curve (b) does not meet the \( \lim_{t \to t_{\text{max}}} \) criterion and is thus not closed. Curve (c) is self intersecting (i.e., \( p(t_1) = p(t_2) \) for some \( t_1 \neq t_2 \)) and is thus not simple.

The points in a curve are ordered in the sense that if \( t_1 < t_2 < t_3 \) then \( p(t_1) < p(t_2) < p(t_3) \). Thus we can construct sequences of points in a curve as well as consider limits such as \( \lim_{\mathbf{p} \to \mathbf{p}_0} \).

1In this thesis, we use vector notation for both points and directions.
A curve is called smooth if the function \( p(t) \) is continuously differentiable. We define differentiability at the point \( p(t_{\text{min}}) \) by considering a shifted parameterization with \( (t_{\text{min}} - \delta) \leq t < (t_{\text{max}} - \delta) \) for some \( \delta > 0 \), thus extending the domain of the function \( p(t) \), such that \( p(t_{\text{min}} - \varepsilon) = p(t_{\text{max}} - \varepsilon) \) whenever \( 0 < \varepsilon \leq \delta \). Curve (d) in Figure 2.1 contains some corner points and is thus not smooth.

Using the notational convention \( \dot{x} = dx/dt \), the speed of a curve parameterization at the point \( p \) is given by the following:

\[
s(t) = ||\dot{p}|| = \sqrt{(\dot{x})^2 + (\dot{y})^2 + (\dot{z})^2}
\]

A curve parameterization \( p(t) \) is said to be regular if its speed is never zero. Any point \( p(t) \) at which the speed of a curve parameterization is zero is said to be a singular point under that parameterization. Some singularities are removable under a change of parameterization; others are not removable.

By default, in this thesis, when we refer to a curve, we mean a smooth closed simple curve that has a regular parameterization.

We define an arc as any subset of a curve where that subset is parameterized by \( t \) with \( a \leq t \leq b \) and \( t_{\text{min}} \leq a < b < t_{\text{max}} \). The end-points of the arc are \( p(a) \) and \( p(b) \). Arcs that include the point \( p(t_{\text{min}}) \) can be constructed using a shift in parameterization as given in the previous discussion of differentiability.

In general, the arc-length of an arc is given by the following:

\[
\int_{a}^{b} s(t) \, dt
\]
2.1. Curves and Arcs

Arc-length can alternatively be considered as a function of $t$ where the length is that from some starting point in a curve, parameterized by $t_0$, to an arbitrary point in the curve parameterized by $t$:

$$l(t) = \int_{t_0}^{t} s(t) \, dt$$  \hspace{1cm} (2.4)

Note that arc-length is independent of the particular curve parameterization that is used. We will additionally restrict our consideration of curves to only those not containing infinite length arcs. This eliminates so-called space filling curves.\(^2\)

The following is a basic theorem from differential geometry and can be found in (Davies and Samuels 1996).

1. **Theorem.** A regular curve reparameterized by arc-length has unit speed at all points.

This reparameterization can be written as

$$A = \left\{ \mathbf{p}(t(l)) = \begin{bmatrix} x(t(l)) \\ y(t(l)) \\ z(t(l)) \end{bmatrix} : 0 \leq l < l(t_{max}) \right\}$$ \hspace{1cm} (2.5)

Note that, to write this reparameterization explicitly, we need to solve Equation (2.4) for $t$, and this is not always possible.

1. **Definition.** The (unit length) tangent direction vector to a curve at point $p$ is given by the following:

$$\hat{t} = \frac{\dot{p}}{||\dot{p}||}$$ \hspace{1cm} (2.6)

There is a unique tangent line associated with each point $p_0$ on a curve. The tangent line is the set of points given by

$$L = \{ \mathbf{p}(t) = p_0 + \hat{t} t : -\infty < t < +\infty \}$$ \hspace{1cm} (2.7)

The tangent line to a curve at a given point is independent of the parameterization used.

Whenever a line is specified in the form of Equation (2.7), both a reference point $p_0$ and a direction vector $\hat{t}$ are given. Note that, in general, for any specific line (set of points), the reference point $p_0$ could be any point on the line (one of an infinite

\(^2\)An example is the Peano curve which fills the entire unit square; see, for example, (Klette and Rosenfeld 2004).
number of possibilities) and direction vector could be $\pm \hat{t}$ (one of exactly two possibilities).

2. DEFINITION. The (unit length) normal direction vector to a curve at point $p$ is given by the following:

$$\hat{n} = \frac{\hat{t}}{||\hat{t}||}$$

(2.8)

Note that the normal vector is undefined both at points where the tangent function is not differentiable and (because of a divide by zero) within any arc where the tangent vector is constant. Also, a basic result from differential geometry shows that, at each point in a curve, the normal vector, where defined, is orthogonal to the associated tangent vector.

2.1.1 Curvature of Plane Curves

Even though, in this section, we focus on plane curves, we will work in $\mathbb{R}^3$ because we seek properties that are generally applicable in 3D space.

3. DEFINITION. Consider a plane curve with arc-length $l$ from the point $p_0$ to the point $p$, and the (non-negative) angle $\alpha$ between the tangent lines at $p_0$ and $p$ as illustrated, for example, in Figure 2.2. Then the curvature of the curve at point $p_0$ is defined to be

$$\kappa = \lim_{p \to p_0} \frac{\alpha}{l} = \frac{d\alpha}{dl}$$

(2.9)
2.1. Curves and Arcs

Figure 2.3: Curvature is not defined at the join point of two half circles.

Note that curvature as defined here is always non-negative. Since neither the angle between the tangent lines, nor the arc-length, are altered by translation and rotation, we conclude that curvature is a translation and rotation independent property.

Curvature is not necessarily defined at all points on a smooth curve. A well-known result gives constant curvature $1/r$ at all points of a circle having radius $r$. However, for example, curvature is not defined at the joining point of the arc constructed from two half circles, even when both have the same radius, as shown in Figure 2.3. The existence of curvature at a point is conditional on a parameterization having a continuous second derivative at that point.

We show, in the following example, that the second derivative for this curve is not continuous.

Consider the arc, constructed from joining two unit radius half-circles as shown in Figure 2.4 parameterized by:
When 0 ≤ t ≤ π we have:

\[ \dot{\mathbf{p}}(t) = \begin{bmatrix} -\sin(t - \pi) \\ \cos(t - \pi) \end{bmatrix} \]  

(2.11)

\[ \ddot{\mathbf{p}}(t) = \begin{bmatrix} -\cos(t - \pi) \\ -\sin(t - \pi) \end{bmatrix} \]  

(2.12)

Making use of the identities \( \cos(-t) = \cos(t) \) and \( \sin(-t) = -\sin(t) \), we have, when \( \pi < t ≤ 2\pi \):

\[ \dot{\mathbf{p}}(t) = \begin{bmatrix} -\sin(t) \\ -\cos(t) \end{bmatrix} \]  

(2.13)

\[ \ddot{\mathbf{p}}(t) = \begin{bmatrix} -\cos(t) \\ \sin(t) \end{bmatrix} \]  

(2.14)

When \( t = \pi \), both of the first derivatives evaluate to \( \begin{bmatrix} 0 \\ 1 \end{bmatrix} \). Thus, the first derivative of the arc, as a whole, is continuous at the join-point. However, again when \( t = \pi \), the second derivatives evaluate to \( \begin{bmatrix} -1 \\ 0 \end{bmatrix} \) and \( \begin{bmatrix} 1 \\ 0 \end{bmatrix} \) respectively. Since the second derivative of the arc parameterization is not continuous at the join point, curvature is not defined there.

Note that there is also a related discontinuity in the direction of the normal vector \( \mathbf{\hat{n}} \) at the join point.

The following theorem from differential geometry summarizes a useful relationship involving curvature (Davies and Samuels 1996).

2. Theorem. The Frenet formula,

\[ \begin{bmatrix} \dot{\mathbf{\hat{t}}} \\ \dot{\mathbf{\hat{n}}} \end{bmatrix} = \begin{bmatrix} 0 & ks \\ -ks & 0 \end{bmatrix} \begin{bmatrix} \mathbf{\hat{t}} \\ \mathbf{\hat{n}} \end{bmatrix} \]  

(2.15)

describes the relationship between the curvature, speed, the tangent and the normal at each point in a planar curve.
Since any plane curve in $\mathbb{R}^3$ can be transformed by rotation and translation such that it lies in the $xy$ coordinate plane, it is meaningful, for curvature calculation purposes, to consider $xy$ plane curves.

Curvature for an $xy$-plane curve can be computed directly, from the curve parameterization, as

$$k = \frac{|\dot{x}\ddot{y} - \ddot{x}\dot{y}|}{(x'^2 + y'^2)^{3/2}}$$  \hspace{1cm} (2.16)

or for a unit speed parameterization as simply

$$k = |\dot{x}\ddot{y} - \ddot{x}\dot{y}|$$  \hspace{1cm} (2.17)

Curvature for $xy$-plane curves, given as a function $y(x)$ in rectangular coordinates, can be computed as

$$k = \frac{|\frac{d^2y}{dx^2}|}{(1 + (\frac{dy}{dx})^2)^{3/2}}$$  \hspace{1cm} (2.18)

and for plane curves given as $r(\theta)$ in polar coordinates as

$$k = \frac{r^2 + 2(\frac{dr}{d\theta})^2 - r^2(\frac{d^2r}{d\theta^2})}{(r^2 + (\frac{dr}{d\theta})^2)^{3/2}}$$  \hspace{1cm} (2.19)

Note that curvature is dependent in general on both first and second order derivatives.

### 2.1.2 Curvature and Planar Regions

A closed simple planar curve separates the plane into two regions: a bounded interior and an unbounded exterior. Figure 2.5 shows two curves, each with its associated interior region shaded.
In this thesis, we associate a (closed simple) planar curve, as well as each of its arcs, with its interior region.

A preliminary definition is required before we reconsider curvature.

**4. Definition.** The (planar) $\varepsilon$-neighborhood of a point $p$ in a plane is the open disk of all points in the plane whose distance to $p$ is less than $\varepsilon$.

Figure 2.6 illustrates the $\varepsilon$-neighborhood of a point in an arc.

The association of an interior region with a curve allows us to refine our definition of curvature, from Section 2.1.1, with an assignment of a polarity sign.

**5. Definition.** Consider the non-zero curvature at a point $p$ on a curve and the tangent line at that point. A negative polarity is assigned to that curvature if there exists an $\varepsilon$-neighborhood of $p$ such that, within that $\varepsilon$-neighborhood, all of the points of the tangent line, except for $p$ itself, are part of the associated interior region. The curvature has a positive polarity otherwise.

Figure 2.7 illustrates the definition of curvature polarity. It shows example arcs, interior regions, $\varepsilon$-neighborhoods, and tangents lines for each of the two curvature polarity possibilities.
2.2 Surfaces and Surface Patches

Throughout the remainder of this thesis, a polarity will always be assigned to (non-zero) curvature.

2.2 Surfaces and Surface Patches

We consider closed simple surfaces in 3D space. Such a surface can be specified parametrically as a set of points

\[
S = \left\{ p(u, v) = \begin{bmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{bmatrix} : (u_{\text{min}} \leq u < u_{\text{max}}) \land (v_{\text{min}} \leq v < v_{\text{max}}) \right\}
\]

where \( p(u, v) \) is a continuous function, \( p(u_1, v_1) \neq p(u_2, v_2) \) for all \( (u_1, v_1) \neq (u_2, v_2) \),

\[ \forall (v) \lim_{u \to u_{\text{max}}} p(u, v) = p(u_{\text{min}}, v) \]

A point in the surface can be referred to as point \( p \).

Example (a) in Figure 2.8 is a closed, simple surface. Surface (b) can be thought of as having started out like surface (a), but with one side pushed back through the opposite side. So, surface (b), with its self intersection, is not simple, i.e. \( p(u_1, v_1) = p(u_2, v_2) \) for some \( (u_1, v_1) \neq (u_2, v_2) \).

A surface is called smooth if the function \( p(u, v) \) is continuously differentiable. We define differentiability at the points \( p(u_{\text{min}}, v) \) by considering a shifted parameterization with \( (u_{\text{min}} - \delta) \leq u < (u_{\text{max}} - \delta) \) for some \( \delta > 0 \), thus extending the domain of the function \( p(u, v) \), such that \( p(u_{\text{min}} - \varepsilon, v) = p(u_{\text{max}} - \varepsilon, v) \) whenever \( 0 < \varepsilon \leq \delta \). We define differentiability at the points \( p(u, v_{\text{min}}) \) similarly. Surface (c) in Figure 2.8 composed of triangle facets, is not smooth where the facets meet.
A surface contains families of curves associated with its parameterization variables. Consider the family of curves given by

\[
A_v = \left\{ \mathbf{p}(u, v) = \begin{bmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{bmatrix} : u_{\min} \leq u < u_{\max} \right\}
\]

(2.21)

for some finite set of discrete fixed values of \( v \) such that \( v_{\min} \leq v < v_{\max} \). Families given by \( B_{u,v} \), with a fixed set of values of \( u \), can be constructed similarly. Each point on a surface is contained in two parameterization curves, one from each of the possible families \( A_u \) and \( B_v \).

Of course, for each curve in the families \( A_u \) and \( B_v \), parameterization speed, arc-length, regularity, singular points, and tangent lines are defined as in Section 2.1.

By default, in this thesis, when we refer to a surface, we mean a smooth closed simple surface which does not contain any singular points.

We define a (closed) surface patch as any subset of a surface where that subset is parameterized by \( u \) and \( v \) where \( a \leq u \leq b \) with \( u_{\min} \leq a < b < u_{\max} \), and \( c \leq v \leq d \) with \( v_{\min} \leq c < d < v_{\max} \). Surface patches that include points given \( \mathbf{p}(u_{\min}, v) \) and \( \mathbf{p}(u, v_{\min}) \) can be constructed using a shift in parameterization as given in the previous discussion of differentiability.

The area of a surface patch is given by the following:

\[
\int_a^b \int_c^d s(u, v) \, dv \, du
\]

(2.22)

Note that surface patch area is independent of the particular surface parameterization that is used.

Consider a point in a surface, and the two tangent lines associated, one each, with the two parameterization curves at that point as shown, for example, in Figure 2.9. The unique plane which includes these two tangent lines is called the surface tangent plane. Note that, for any given parameterization, the two tangent lines are not necessarily orthogonal to each other.

In general, a plane can be specified as the set of points

\[
P = \{ \mathbf{p} : (\mathbf{p} - \mathbf{p}_0) \cdot \hat{\mathbf{n}} = 0 \}
\]

(2.23)
2.2. Surfaces and Surface Patches

where the reference point \( p_0 \) can be any point on the plane (one of an infinite number of possibilities) and the normal direction vector \( \hat{n} \), could just as well be \( -\hat{n} \) (one of exactly two possibilities).

In the specific case of a surface tangent plane, a normal direction vector can be calculated simply as

\[
\hat{n} = \hat{t}_1 \times \hat{t}_2 \quad (2.24)
\]

where \( \hat{t}_1 \) and \( \hat{t}_2 \) are surface tangent line directions.

Once we have a surface tangent plane, we can construct a surface tangent line in any direction orthogonal to the tangent plane normal direction. We are thus not limited to the parameterization directions when considering surface tangent lines.

2.2.1 Surface Curvature and Solid Regions

A closed simple surface separates \( \mathbb{R}^3 \) into two solid regions: a bounded interior and an unbounded exterior. Figure 2.10 shows a surface patch, with a portion of its associated interior region.

---

3This is an example of a separation theorem. Separation theorems are of great importance in topology, and have a rich history that dates back to (Jordan 1887) and (Veblen 1905). For further discussion, see (Klette and Rosenfeld 2004).
In this thesis, we associate a (closed simple) surface, as well as each of its surface patches, with its interior region.

We build up to a definition of the surface normal vector with some preliminaries.

6. **DEFINITION.** The $\varepsilon$-neighborhood of a point $p$ in $\mathbb{R}^3$ is the open sphere of all points whose distance to $p$ is less than $\varepsilon$.

A ray is the (half-line) set of points given by

$$R = \{ p(t) = p_0 + \hat{r} t : 0 \leq t < +\infty \}$$

(2.25)

where $p_0$ is the ray starting point and $\hat{r}$ is the (unit length) ray direction vector.

Consider a point $p$ on a surface, the tangent plane at that point, and the rays associated with the two possible tangent plane normal vectors. One of those normal vectors is exterior pointing in that, for its associated ray, there exists an $\varepsilon$-neighborhood of $p$ such that, within that $\varepsilon$-neighborhood, all of the points of the ray, except for $p$ itself, are part of the surface exterior region. See Figure 2.11 for examples.

7. **DEFINITION.** The surface normal vector at a point in a surface is the exterior pointing tangent plane normal vector at that point.

A surface cutting plane through the point $p$ in a surface, is any plane, other than the tangent plane, containing $p$. A normal cutting plane through a point in a surface, is a surface cutting plane that includes the surface normal at that point. Note that there is a unique normal cutting plane that includes any given surface tangent line. Figure 2.12 illustrates examples of cutting planes.
2.2. Surfaces and Surface Patches

Figure 2.12: Cutting planes: (a) aligned with a surface tangent line, (b) aligned with the surface normal, (c) the unique plane aligned with both a surface tangent line and the surface normal.

Figure 2.13: Cutting plane intersections: (a) single curve, (b) two curves, (c) self-intersecting curve, (d) concentric curves.

The intersection of a surface and a surface cutting plane associated with a point in the surface, is (locally) a smooth simple arc which includes that point. Globally, the intersection may include singular points, additional curves, and the curves may self-intersect. Examples are shown in Figure 2.13. Note that in Figure 2.13 (d), the surface interior region is an annulus which lies between two concentric intersection curves.

8. **Definition.** The normal curvature $\kappa_n$ at a point in a surface, along the direction of a surface tangent line, is the (signed) curvature of the curve which is the intersection of that surface and the normal cutting plane aligned with the tangent line at that point.

With this definition, curvature polarity is assigned based on surface interior points within the cutting plane subset thought of as planar curve interior points.

9. **Definition.** The principal curvatures, $\kappa_1$ and $\kappa_2$, at a point in a surface are the respective maximum and minimum values of the normal curvature at that point.
The following Theorem is a fundamental result from differential geometry and can be found in (do Carmo 1976).

3. **Theorem.** The relationship between normal curvature and the principal curvatures is given by

\[ \kappa_n = \kappa_1 \cos^2 \theta + \kappa_2 \sin^2 \theta \]  

(2.26)

where \( \theta \) is the angle between an arbitrary normal curvature (\( \kappa_n \)) cutting plane (or direction) and the maximum curvature (\( \kappa_1 \)) cutting plane (or direction).

Theorem 3 implies that the principal curvature cutting planes (and directions) are orthogonal to each other. (This can be verified by setting \( \theta \) in Equation (2.26) to the value 0 and then to the value \( \pm \pi/2 \).)

We are now in a position to define two frequently used surface curvature measures.

10. **Definition.** The mean curvature is defined as

\[ H = \frac{\kappa_1 + \kappa_2}{2} \]  

(2.27)

11. **Definition.** The Gaussian curvature is defined as

\[ K = \kappa_1 \kappa_2 \]  

(2.28)

Often, in practice, as we see later in this thesis, normal curvatures can be directly calculated, but the principal curvatures cannot. Fortunately, using the following theorem, mean curvature can be calculated without knowing the principal curvatures.

4. **Theorem.** (Two-Cut Mean Curvature) Using any two normal cutting planes which are orthogonal to each other, the mean curvature at a point on a surface is given by

\[ H = \frac{\kappa_n_1 + \kappa_n_2}{2} \]  

(2.29)

where \( \kappa_n_1 \) and \( \kappa_n_2 \) are normal curvatures associated with those cutting planes.

**Proof.** Consider the normal curvatures, \( \kappa_n_1 \) and \( \kappa_n_2 \), associated with two orthogonal normal cutting planes, with the first cutting plane oriented at an arbitrary angle.

---

We provide a proof of this known theorem for reasons of later generalization.
with the maximum principal curvature direction, and the second cutting plane oriented at the angle $\alpha + \pi/2$. From Equation (2.26) we have that

$$\kappa_n = \kappa_1 \cos^2 \alpha + \kappa_2 \sin^2 \alpha$$  \hspace{1cm} (2.30)

and

$$\kappa_{n_2} = \kappa_1 \left( \cos(\alpha + \pi/2) \right)^2 + \kappa_2 \left( \sin(\alpha + \pi/2) \right)^2$$  \hspace{1cm} (2.31)

$$= \kappa_1 (- \sin \alpha)^2 + \kappa_2 (\cos \alpha)^2$$  \hspace{1cm} (2.32)

$$= \kappa_1 \sin^2 \alpha + \kappa_2 \cos^2 \alpha$$  \hspace{1cm} (2.33)

Combining Equations (2.30) and (2.33), we have that

$$\kappa_{n_1} + \kappa_{n_2} = \left( \kappa_1 \sin^2 \alpha + \kappa_2 \cos^2 \alpha \right) + \left( \kappa_1 \cos^2 \alpha + \kappa_2 \sin^2 \alpha \right)$$  \hspace{1cm} (2.34)

$$= \kappa_1 \sin^2 \alpha + \kappa_1 \cos^2 \alpha + \kappa_2 \sin^2 \alpha + \kappa_2 \cos^2 \alpha$$  \hspace{1cm} (2.35)

$$= \kappa_1 (\sin^2 \alpha + \cos^2 \alpha) + \kappa_2 (\sin^2 \alpha + \cos^2 \alpha)$$  \hspace{1cm} (2.36)

$$= \kappa_1 + \kappa_2$$  \hspace{1cm} (2.37)

$$= \kappa_1 + \kappa_2$$  \hspace{1cm} (2.38)

It follows that

$$\frac{\kappa_{n_1} + \kappa_{n_2}}{2} = \frac{\kappa_1 + \kappa_2}{2} = H$$  \hspace{1cm} (2.39)

This proves the theorem. \hfill \Box

In this thesis, we generalize the two-cut mean curvature theorem to $n$-cuts. But first, some preliminary results are required.

1. **Lemma.** For $n \geq 2$, and a non-zero integer $c$,

$$\sum_{m=0}^{n-1} \cos(2c\pi m/n) = \sum_{m=0}^{n-1} \sin(2c\pi m/n) = 0$$  \hspace{1cm} (2.40)

**Proof.**
Recall the following identity for the partial sums of a geometric series where $z$ is a complex number:

$$z^0 + z^1 + z^2 + \ldots + z^{n-1} = \frac{z^n - 1}{z - 1}$$  \hspace{1cm} (2.41)
Consider the following summation, where \( n \geq 2 \), and \( c \) is a non-zero integer:

\[
\sum_{m=0}^{n-1} e^{i2c\pi\frac{m}{n}} = (e^{i2c\pi/n})^0 + (e^{i2c\pi/n})^1 + (e^{i2c\pi/n})^2 + \cdots + (e^{i2c\pi/n})^{n-1} \quad (2.42)
\]

\[
= \frac{(e^{i2c\pi/n})^n - 1}{e^{i2c\pi/n} - 1} \quad (2.43)
\]

\[
= \frac{e^{i2c\pi} - 1}{e^{i2c\pi/n} - 1} \quad (2.44)
\]

\[
= \frac{1 - 1}{e^{i2c\pi/n} - 1} \quad (2.45)
\]

\[
= \frac{0}{e^{i2c\pi/n} - 1} = 0 \quad (2.46)
\]

But we also have that

\[
\sum_{m=0}^{n-1} e^{i2\pi\frac{m}{n}} = \sum_{m=0}^{n-1} \left( \cos \left(2\pi \frac{m}{n} \right) + i \sin \left(2\pi \frac{m}{n} \right) \right) \quad (2.47)
\]

\[
= \sum_{m=0}^{n-1} \cos \left(2\pi \frac{m}{n} \right) + i \sum_{m=0}^{n-1} \sin \left(2\pi \frac{m}{n} \right) \quad (2.48)
\]

Taking Equations (2.46) and (2.48) together gives

\[
\sum_{m=0}^{n-1} \cos \left(2\pi \frac{m}{n} \right) + i \sum_{m=0}^{n-1} \sin \left(2\pi \frac{m}{n} \right) = 0 \quad (2.49)
\]

which implies that

\[
\sum_{m=0}^{n-1} \cos \left(2\pi \frac{m}{n} \right) = \sum_{m=0}^{n-1} \sin \left(2\pi \frac{m}{n} \right) = 0 \quad (2.50)
\]

This proves the lemma. \( \square \)

2. Lemma. For \( n \geq 2 \),

\[
\sum_{m=0}^{n-1} \cos^2 \left(2\pi \frac{m}{n} \right) = \sum_{m=0}^{n-1} \sin^2 \left(2\pi \frac{m}{n} \right) = \frac{n}{2} \quad (2.51)
\]
2.2. Surfaces and Surface Patches

**Proof.**

\[
\sum_{m=0}^{n-1} \cos^2 \left( \frac{2\pi m}{n} \right) = \sum_{m=0}^{n-1} \frac{1 + \cos \left( \frac{4\pi m}{n} \right)}{2} = \frac{n}{2} + \frac{1}{2} \left( \frac{1}{2} \right) = \frac{n}{2}
\]  

(2.52)

\[
\sum_{m=0}^{n-1} \sin^2 \left( \frac{2\pi m}{n} \right) = \sum_{m=0}^{n-1} \frac{1 - \cos \left( \frac{4\pi m}{n} \right)}{2} = \frac{n}{2} - \frac{1}{2} \left( \frac{1}{2} \right) = \frac{n}{2}
\]  

(2.54)

This proves the lemma.

\[\square\]

5. **Theorem.** \((n\text{-Cut Mean Curvature})\) Using any \(n \geq 2\) equally spaced (by angle) normal cutting planes, the mean curvature at a point on a surface is the mean of the associated normal curvatures.

This can be written as:

\[
H = \frac{1}{n} \sum_{m=0}^{n-1} \kappa_{n,m}
\]  

(2.58)

where \(\kappa_{n,m}\) is the normal curvature associated with the \((m+1)\)-th cutting plane.

**Proof.** Consider \(n \geq 2\) equally spaced normal cutting planes, with an arbitrary constant angle \(\alpha\) between the first cutting plane and the maximum principal curvature direction.

Let \(\beta = 2\pi \frac{m}{n}\). From Equation (2.26) we have:

\[
\sum_{m=0}^{n-1} \kappa_{n,m} = \kappa_1 \sum_{m=0}^{n-1} \cos^2 \left( \alpha + \beta \right) + \kappa_2 \sum_{m=0}^{n-1} \sin^2 \left( \alpha + \beta \right)
\]  

(2.68)
Consider the $\kappa_1$ summation term from Equation (2.59):
\[
\sum_{m=0}^{n-1} \cos^2(\alpha + \beta) = \sum_{m=0}^{n-1} (\cos \alpha \cos \beta - \sin \alpha \sin \beta)^2 \quad (2.60)
\]
\[
= \sum_{m=0}^{n-1} (\cos^2 \alpha \cos^2 \beta - 2 \cos \alpha \sin \alpha \cos \beta \sin \beta + \sin^2 \alpha \sin^2 \beta) \quad (2.61)
\]
\[
= \cos^2 \alpha \sum_{m=0}^{n-1} \cos^2 \beta - \cos \alpha \sin \alpha \sum_{m=0}^{n-1} \sin 2\beta + \sin^2 \alpha \sum_{m=0}^{n-1} \sin^2 \beta \quad (2.62)
\]
\[
= (\cos^2 \alpha) \left( \frac{n}{2} \right) - (\cos \alpha \sin \alpha) \left( 0 \right) + (\sin^2 \alpha) \left( \frac{n}{2} \right) \quad (2.63)
\]
\[
= (\cos^2 \alpha + \sin^2 \alpha) \frac{n}{2} \quad (2.64)
\]
\[
= \frac{n}{2} \quad (2.65)
\]
Note that Equation (2.63) employs Lemmas [1] and [2].

Consider the $\kappa_2$ summation term from Equation (2.59):
\[
\sum_{m=0}^{n-1} \sin^2(\alpha + \beta) = \sum_{m=0}^{n-1} (\sin \alpha \cos \beta + \cos \alpha \sin \beta)^2 \quad (2.66)
\]
\[
= \sum_{m=0}^{n-1} (\sin^2 \alpha \cos^2 \beta + 2 \cos \alpha \sin \alpha \cos \beta \sin \beta + \cos^2 \alpha \sin^2 \beta) \quad (2.67)
\]
\[
= \sin^2 \alpha \sum_{m=0}^{n-1} \cos^2 \beta + \cos \alpha \sin \alpha \sum_{m=0}^{n-1} \sin 2\beta + \cos^2 \alpha \sum_{m=0}^{n-1} \sin^2 \beta \quad (2.68)
\]
\[
= (\sin^2 \alpha) \left( \frac{n}{2} \right) + (\cos \alpha \sin \alpha) \left( 0 \right) + (\cos^2 \alpha) \left( \frac{n}{2} \right) \quad (2.69)
\]
\[
= (\sin^2 \alpha + \cos^2 \alpha) \frac{n}{2} \quad (2.70)
\]
\[
= \frac{n}{2} \quad (2.71)
\]
Substituting these results back into Equation (2.59) gives:
\[
\sum_{m=0}^{n-1} \kappa_{nm} = \kappa_1 \left( \frac{n}{2} \right) + \kappa_2 \left( \frac{n}{2} \right) \quad (2.72)
\]
\[
= n \left( \frac{\kappa_1 + \kappa_2}{2} \right) \quad (2.73)
\]
\[
\frac{1}{n} \sum_{m=0}^{n-1} \kappa_{nm} = \frac{\kappa_1 + \kappa_2}{2} = H \quad (2.74)
\]
2.2. Surfaces and Surface Patches

This proves the theorem. □

2.2.2 Surface Curvature Computation

In our discussion of surface curvature so far, we have taken a geometric view in the sense that we have used cutting planes in defining normal curvature and, hence, the principal curvatures. (The motivation for this approach becomes clear later in this thesis when we consider digitized surfaces.) Alternatively, given an explicit surface parameterization (i.e., a set of parametric equations), it is also possible to take an analytical approach to surface curvature and its computation.

To this end, a number of standard working variables are defined in differential geometry:

Given a surface parameterization \( p(u, v) \), the first fundamental form variables are

\[
E = p_u \cdot p_u \quad (2.75)
\]

\[
F = p_u \cdot p_v \quad (2.76)
\]

\[
G = p_v \cdot p_v \quad (2.77)
\]

and the second fundamental form variables are

\[
l = \hat{n} \cdot p_{uu} \quad (2.78)
\]

\[
m = \hat{n} \cdot p_{uv} \quad (2.79)
\]

\[
n = \hat{n} \cdot p_{vv} \quad (2.80)
\]

where we have used the notational convention, for example, \( p_u = \partial p(u, v)/\partial u \) and \( p_{uu} = \partial p_{u}/\partial u \).

The Weingarten matrix \( L \) is defined as the unique \( 2 \times 2 \) matrix, such that, for every point in a surface, we have

\[
\begin{bmatrix}
\hat{n}_u & \hat{n}_v
\end{bmatrix} = \begin{bmatrix} p_u & p_v \end{bmatrix} L \quad (2.81)
\]

where

\[
\hat{n} = \frac{p_u \times p_v}{||p_u \times p_v||} \quad (2.82)
\]

Remarkably, the eigenvalues of the matrix \( L \) give the principal curvatures (as \(-\kappa_1\) and \(-\kappa_2\)), and the eigenvectors of \( L \) are the principal curvature directions. Note that a curvature direction corresponds to the direction of the surface tangent line
that is coincident with the associated normal curvature cutting plane.

One of the most important theoretical results in differential geometry is an explicit formulation for the Weingarten matrix in terms of the first and second fundamental form variables:

\[
L = \frac{1}{EG - F^2} \begin{bmatrix}
Fm - Gm & Fn - Gm \\
Fl - Em & Fm - En
\end{bmatrix}
\]  

(2.83)

It is well worth noting that, with the assistance of modern symbolic math computer software that can compute eigenvalues and eigenvectors, the Weingarten matrix can be put to practical use, giving explicit symbolic formulations for the principal curvatures and the principal curvature directions.

Additionally (or alternatively), mean curvature can be given in terms of the first and second fundamental form variables as

\[
H = \frac{En + Gl - 2Fm}{2(EG - F^2)}
\]  

(2.84)

and Gaussian curvature is given as

\[
K = \frac{ln - m^2}{EG - F^2}
\]  

(2.85)

And then, using the definitions of mean and Gaussian curvature given in Equations (2.27) and (2.27), we can extract the principal curvatures as

\[
\kappa_1 = H + \sqrt{H^2 - K}
\]  

(2.86)

and

\[
\kappa_2 = H - \sqrt{H^2 - K}
\]  

(2.87)

For derivations of Equations (2.83), (2.84) and (2.85) see, for example (Davies and Samuels 1996) or (do Carmo 1976).

### 2.3 Scale Invariant Curvature

It is generally useful to seek *invariant* properties when characterizing 3D objects. At a minimum, translation and rotation invariant characterization are desired as clearly

\textsuperscript{5}Symbolic math software encourages users to engage in symbolic formulations that might otherwise be considered intractable!
2.3. Scale Invariant Curvature

neither of these transformations alters the essential shape property of an object. Surface curvature, a rotation and translation invariant property, meets this requirement (Davies and Samuels 1996).

Any characterization that is additionally scaling invariant enables determining the equivalence of shapes independent of size. Perhaps not so obvious, practically speaking, this scale invariance would also enable the use of uncalibrated measurement units in 3D digitization (e.g. scanning).

None of the curvature measures defined so far in this chapter are scale invariant.

In this section we introduce a new scale invariant curvature measure, similarity curvature. We also define a similarity curvature space which consists of the set of all possible similarity curvature values.


2.3.1 Geometric Invariants

As previously noted, with existing surface curvature definitions, we already have translation and rotation invariance. What we now seek is scaling invariance. Shape characterization based on moments has been studied since (Hu 1962), with varying emphasis on invariance with respect to translation, rotation, reflection, or scaling.

Related work (Sapiro 2001), among other things, generalizes and extends the invariance concepts contained in affine differential geometry. Affine invariance is stronger than what we seek in that it includes, for example, squash and stretch transformations.

A number of authors touch on scaling invariant properties in their exploration of multi-scale properties. For example, in (Mokhtarian and Bober 2003), firstly surface feature points such as maximum curvature locations are identified. Then triples of feature points are combined using a geometric hashing algorithm in a way that is scaling invariant. Hash tables for various objects of interest are statistically com-
pared to check for similarity matches between different objects.

### 2.3.2 A Scale Invariant Curvature Measure

In this thesis, we present a scale invariant curvature measure that can be assigned at every point on a surface. We keep in mind that any definition of scale invariant surface curvature must be related to geometric similarity in which it is well-known that 1) angles are preserved and 2) ratios of lengths are preserved.

We start with some definitions.

12. **Definition.** The curvature ratio \( \kappa_3 \) is defined as

\[
\kappa_3 = \frac{\min(|\kappa_1|, |\kappa_2|)}{\max(|\kappa_1|, |\kappa_2|)}
\]

In the case when \( \kappa_1 \) and \( \kappa_2 \) are both equal to zero, \( \kappa_3 \) is defined as being equal to zero. Note that \( 0 \leq \kappa_3 \leq 1 \).

13. **Definition.** The curvature measure \( R \) at a point in a surface is defined as

\[
R = \begin{cases} 
(\kappa_3, 0) & \text{if signs of } \kappa_1 \text{ and } \kappa_2 \text{ are both positive,} \\
(-\kappa_3, 0) & \text{if signs of } \kappa_1 \text{ and } \kappa_2 \text{ are both negative,} \\
(0, \kappa_3) & \text{if signs of } \kappa_1 \text{ and } \kappa_2 \text{ differ, and } \kappa_1 \geq |\kappa_2|, \\
(0, -\kappa_3) & \text{if signs of } \kappa_1 \text{ and } \kappa_2 \text{ differ, and } \kappa_1 < |\kappa_2|.
\end{cases}
\]

Note that \( R \in \mathbb{R}^2 \).

6. **Theorem.** For every closed, simple, smooth surface, the curvature measure \( R \) is (positive) scaling invariant.

**Proof.** Consider a point on a surface and any associated normal curvature \( \kappa \), as well as the resultant normal curvature \( \kappa' \) after scaling the surface by a factor \( s \). Since, by definition, \( \kappa = d\alpha/dl \), and scaling alters length but not angle, we have \( \kappa' = \kappa/s \).

Therefore, after scaling, both of the principal curvatures change by the same factor, and the ratio of the principal curvatures is unchanged. Also, neither the signs, nor the relative magnitudes of the principal curvatures are changed by scaling.

This proves the theorem. \( \square \)

Henceforth, we will refer to the curvature measure \( R \) as the *similarity curvature*. 
2.3. Scale Invariant Curvature

2.3.3 The Similarity Curvature Space

Since the set of all possible values of the similarity curvature $R$ is a subset of $\mathbb{R}^2$, it is natural to consider a two-dimensional plot representation. Also recall, from differential geometry, that all surface patches on continuous smooth surfaces are locally either elliptic, hyperbolic (saddle-like), parabolic or planar.

We introduce the similarity curvature plot template in Figure 2.14. The horizontal $e$-axis is for curvature values at locally elliptic surface points. The vertical $h$-axis is for curvature values at locally hyperbolic surface points. Both parabolic and planar surface points have similarity curvature value $(0, 0)$ and are thus plotted at the origin. Plotted similarity curvature values will never be off the axes.

When considering examples, it is clear that the similarity curvature at every point on all spheres is constant and equal to $(1, 0)$. The similarity curvature on every planar surface, every cylinder and every cone is constant and equal to $(0, 0)$. Note that this is exactly where the Gaussian curvature is equal to zero.

Continuous motion through points on a smooth surface, taking the similarity curvature at each point, results in a related continuous motion in the similarity plot space. We observe, for example, that it is not possible to traverse from similarity curvature $(-1,0)$ to similarity curvature $(1,0)$ without going through similarity curvature $(0,0)$.

However, it is possible to traverse from $(0,1)$ directly to $(0,-1)$. This can be described by saying that the $h$-axis wraps around. An example where this wrapping occurs is given later in this thesis.

---

Figure 2.14: $eh$-plot space for similarity curvature.

---

*Excluding the cylinder and cone edges and the cone apex.
2.4 Curvature Visualization

We can visualize surface curvature magnitude in illustrations by coloring or grayscale shading the surface points. With this visualization technique, we can see the patterns of curvature variation across a surface. The goal in this section is to gain insight into the nature of, and differences between, curvature measures.

Given an explicit surface parameterization, Equations (2.84) and (2.85) can be used respectively to calculate exact values for mean and Gaussian curvature.

Consider the following sphere parameterization:

\[
S = \left\{ \mathbf{p}(u, v) = \begin{bmatrix} r \sin u \\ r \cos u \cos v \\ r \cos u \sin v \end{bmatrix} : \left(-\frac{\pi}{2} \leq u < \frac{\pi}{2}\right) \land \left(0 \leq v < 2\pi\right) \right\} \quad (2.88)
\]

Figure 2.15 shows results for this sphere parameterization when the radius \( r = 2 \). The color coded sphere object itself is shown on the left, and curvature versus parameterization is shown on the right. For spheres, in general, both the mean curvature and the Gaussian curvature are always constant. For the sphere shown, the mean curvature is \( \frac{1}{2} \) and the Gaussian curvature is \( \frac{1}{4} \).

Consider the following cylinder parameterization:

\[
S = \left\{ \mathbf{p}(u, v) = \begin{bmatrix} h v \\ r \sin u \\ r \cos u \end{bmatrix} : (0 \leq u < 2\pi) \land (0 \leq v < 1) \right\} \quad (2.89)
\]

Figure 2.16 shows results for this cylinder parameterization when the radius \( r = 1 \) and the height \( h = 2 \). For cylinders, in general, both the mean curvature and the Gaussian curvature are always constant, as is the case for spheres. However, in contrast with spheres, all cylinders have Gaussian curvature zero. For the cylinder shown, the mean curvature is \( \frac{1}{2} \).
2.4. Curvature Visualization

Consider the following truncated cone parameterization:

\[ S = \left\{ p(u, v) = \begin{bmatrix} hv \\ r(1 - v) \sin u \\ r(1 - v) \cos u \end{bmatrix} : (0 \leq u < 2\pi) \land (0 \leq v < 0.8) \right\} \]  (2.90)

Figure 2.17 shows results for this cone parameterization when the base radius \( r = 1/2 \) and the height \( h = 1 \), with the apex cut off at 0.8 times the height. As is the case for cylinders, all cones have Gaussian curvature zero. For the cylinder shown, the mean curvature is 1 at the base, increasing to 4.5 towards the apex.
Figure 2.16: Mean (top) and Gaussian (bottom) curvature for a parameterized cylinder of radius 1 and height 2.

Consider the following torus parameterization:

\[
S = \left\{ p(u, v) = \begin{bmatrix} (a + b \cos u) \cos v \\ (a + b \cos u) \sin v \\ b \sin u \end{bmatrix} : (0 \leq u < 2\pi) \land (0 \leq v < 2\pi) \right\}
\] (2.91)

Figure 2.18 shows results for this torus parameterization when radius \( a = 1 \) and radius \( b = 1/3 \). In this figure, the color range for each of the curvatures is separately set to span the range of curvature values. Thus, the range (and offset) of the color scale for each of the curvatures is not the same. However, with this color scale, the
2.4. Curvature Visualization

Figure 2.17: Mean (top) and Gaussian (bottom) curvature for a (truncated) parameterized cone having base radius $1/2$ and height 1.

curvature visualizations for Gaussian and mean curvature appear almost identical. In this sense, at least for the given torus, the shape of the curvatures is nearly the same.

Of course, for the given torus, the absolute values of the curvatures are clearly not the same. The mean curvatures are non-negative, while the Gaussian curvatures span both positive and negative values.
Consider the following ellipsoid parameterization:

\[
S = \left\{ \mathbf{p}(u, v) = \begin{bmatrix} a \sin u \\ b \cos u \cos v \\ c \cos u \sin v \end{bmatrix} : (-\pi/2 \leq u < \pi/2) \land (0 \leq v < 2\pi) \right\}
\] (2.92)

Figure 2.19 shows results for this ellipsoid parameterization when \(a = \sqrt{3}, b = \sqrt{2},\) and \(c = 1\). In this figure, the color range for each of the curvatures is separately set to span the range of curvature values. Thus, the range (and offset) of the color coding scale for each of the curvatures is not the same. However, with this color scale, as was the case for the torus, the curvature visualizations for Gaussian and
2.4. Curvature Visualization

Figure 2.19: Mean (top) and Gaussian (bottom) curvature for a parameterized ellipsoid.

mean curvature appear similar. Again, the shape of the curvatures is similar.

Interestingly, at least for the given ellipsoid, the actual values of the Gaussian and mean curvatures are similar as well.

For any given surface parameterization, Equations (2.84) and (2.85) can also be used to develop explicit formulae for Gaussian and mean curvature respectively. However, performing the algebraic operations required to simplify these formulae can be rather tedious!

The resultant formula for Gaussian curvature of the ellipsoid given by Equa-
2. Curvature and Differential Geometry

Figure 2.20: EH-plot space color assignment for the ellipsoid example.

The curvature function (2.92) is

\[ K = \frac{a^2b^2c^2}{(b^2c^2 - ((b^2 - c^2)a^2 \cos^2 v + b^2(c^2 - a^2)) \cos^2 u)^2} \]  

(2.93)

and the resultant formula for mean curvature is

\[ H = \left( \frac{abc}{2} \right) \frac{b^2 + c^2 + (a^2 - c^2 + (c^2 - b^2) \cos^2 v) \cos^2 u}{(b^2c^2 - ((b^2 - c^2)a^2 \cos^2 v + b^2(c^2 - a^2)) \cos^2 u)^{3/2}} \]  

(2.94)

Using the same ellipsoid parameterization, we can also visualize an example of similarity curvature.

Figure 2.21 shows results for this ellipsoid parameterization when \( a = \sqrt{3}, b = \sqrt{2}, \) and \( c = 1 \) at a 1x scale and a 100x scale. Note that, since similarity curvature for an ellipsoid is always on the \( e \)-axis in the similarity curvature space, we have assigned the color range to that axis only as shown in Figure 2.20. As expected, the similarity curvature is the same at both scales. It is interesting to note that, with this ellipsoid example, similarity curvature takes the sphere curvature value \((1, 0)\) at exactly four points.

Figure 2.22 shows results for the same ellipsoid parameterization when \( a = \sqrt{3}, b = \sqrt{3}, \) and \( c = 1 \). It is interesting to note that, in comparison with the previous example, since this ellipsoid has two equal length axes, similarity curvature takes the sphere curvature value \((1, 0)\) only at exactly two points.
2.4. Curvature Visualization

Figure 2.21: Similarity curvature for a parameterized ellipsoid scaled at 1x (top) and 100x (bottom).
Figure 2.22: Similarity curvature for a parameterized ellipsoid with two equal axes.
Chapter 3

Digital Surface Curvature

The surface curvature of continuous smooth lines and surfaces is a well-defined property, however, when working with point set data, mesh surfaces or 3D digital images, line and surface curvatures can only be estimated (Kalogerakis et al. 2007) (Klette and Rosenfeld 2004) (Mitra and Nguyen 2003). In this chapter we review the nature of digitized surfaces and present a number of curvature estimators, one of which (the 3-cut mean estimator) is new.

3.1 Digitized Surfaces

Digitized surfaces arise in numerous practical applications and the form that the digitization takes is highly application dependant. For a range of different applications, see, for example, (Ikeuchi et al. 2003), (Allen et al. 2002), (Levoy et al. 2000), (Bloomenthal 1985), and (Rogers and Satterfield 1980). The data acquisition methods associated with surface digitization can be separated into two categories: volume-based and range-based. Consider, for example, the surface of a spherical object as illustrated in Figure 3.1(a).

Voxel digitizations, as shown, for example, in Figure 3.1(b), are volume-based, with the resultant data-set often being built up from multiple 2D scan slice layers. In volumetric applications, surfaces are usually defined as the boundary associated with a physical density discontinuity within an otherwise solid object (e.g. the medical application of locating bone surface within the volume of an arm). The voxel grid is usually regular, but the voxels are not necessarily cubes. This situation can arise, for example, from the fact that it is not uncommon for the distance between scan slices to be greater than the resolution within each 2D scan. Voxels are inherently volume elements, and as such, can only estimate actual surface location. See, for example, (Srámek 1994) and (Gargantini et al. 1993). However, it could be reasonably expected that each voxel would intersect the actual surface. Because voxel data is associated with positions in a fixed grid structure, a voxel data-set can be
3. Digital Surface Curvature

stored as integers which index position within that grid.

For examples of techniques which employ volumetric data structures, see (Fournier et al. 2006) and (Montani and Scopigno 1990). For a comprehensive theoretical treatment of voxel digitization that draws on concepts from topology, see (Kovalevsky 2008) and (Kovalevsky 2003).

Point cloud digitizations are range-based (e.g., laser triangulation or laser time-of-flight) and the data consists of 3D space points, presented possibly in no particular sequence, and generally with unequal point separation distance. In Figure 3.1(c), for example, the concentration of points is greater at the top and bottom poles of the sphere. Unlike a voxel digitization, each point in a point cloud digitization is considered (ideally) to lie on the actual surface. Point cloud data is usually stored as a collection of real number 3D space coordinates.

For a curvature estimation technique that works directly with point cloud data as input, see (Tong and Tang 2005). This work builds on previous work in which the concept of tensor voting is developed (Tang et al. 1999). For an approach in which the local surface geometry in point cloud data is modeled by a set of quadratic curves see (Agam and Tang 2005b), (Agam and Tang 2005a) and (Tang and Agam 2004).

Triangle mesh digitizations, as shown, for example, in Figure 3.1(d), approximates an actual surface as a set of adjoining flat triangular surface patches. It seems reasonable to assume that at least some part of each triangle patch would intersect the actual surface, but, in practice, there is no specific general rule. The triangle mesh data structure is often thought of as consisting of points, edges and faces. The
face adjacency pattern is an essential characteristic of a triangle mesh, and this adjacency is exploited by many triangle mesh processing algorithms.

Triangle meshes are the standard data structure used in 3D graphics modeling; see, for example, (Foley et al. 2002). 3D graphics modeling is a very active research area resulting in varied and numerous techniques. See, for example, books (Lengyel 2004) and (Hill and Kelley 2007). Triangle meshes can be derived from both voxel based and point cloud digitizations (e.g. (Cooper et al. 2003), (Kobbelt et al. 2001), (Curless 2000), (Curless and Levoy 1996)), and in this sense can be thought of as having higher order structure. Alternatively, a triangle mesh can also arise naturally when a range scan is acquired using a controlled rasterization pattern.

Most real-world digitization processes introduce noticeable noise into the acquired data-set. This noise results in some uncertainty as to whether or not, for example, a voxel actually intersects the real surface, or a point in a point cloud actually lies on the real surface. Fortunately, it is possible to mathematically model noise and to reduce its negative impact through filtering. Several filtering techniques are presented later in this thesis.

In this thesis, in addition to surface digitizations of real-world objects, we will consider synthetic objects and their digitizations for the purposes of characterizing the performance of processing algorithms, especially under controlled noise conditions.

Henceforth, we will refer to the data-set associated with a surface digitization as a digital surface.

### 3.2 Curvature Estimators

Each of the curvature estimators presented in this thesis requires known point adjacency neighborhood information. This adjacency information is inherent in a triangle mesh, and, for this reason, we limit our consideration to triangle meshes. In practice, this restriction to triangle meshes is not particularly limiting in that conversion from other digitization types is usually possible.

---

1For any given set of surface points, more than one triangulation is possible, however, this does not affect the validity of the curvature estimators which are presented.
The estimators used in this thesis are all local in the sense that only knowledge of the nearby neighboring points is used in the computation of curvature.

3.2.1 Triangle Umbrella

First we review existing Gaussian and mean curvature estimators used by other authors, for example, (Alboul and van Damme 1996), (Dyn et al. 2000), and (Meek and Walton 2000), which rely on knowledge of a complete triangle umbrella neighborhood around a point of interest.

With reference to Figure 3.2(a), consider a point in the surface and, say, six adjacent points. The points are thought to be connected by edges, and edges enclose, in this case, six faces. We also identify an area $A(f_n)$ and a central angle $\alpha_n$ associated with each face $f_n$.

In Figure 3.2(b), we identify a surface normal vector associated with each face from an edge-on view point. The dihedral angle between adjacent face normals is designated as $\beta$. Angle $\beta$ is positive if the faces form a convex surface (when viewed from the exterior) and $\beta$ is negative if the faces form a concave surface (again, when viewed from the exterior).
3.2. Curvature Estimators

The interior angles of an arbitrary triangle as shown in Figure 3.3.

The Gaussian curvature at the point \( p_0 \) is estimated by

\[
\tilde{K}(p_0) = \frac{3(2\pi - \sum \alpha_n)}{\sum A(f_n)}
\]  

(3.1)

Interestingly, we will show, in the following derivation, that this estimator is generally valid, without change, in the case of any adjacency point count greater than or equal to three.

This estimator can be derived as follows:

Consider, the interior angles of an arbitrary triangle as shown in Figure 3.3. Of course, for a flat triangle, the sum of the interior angles is \( \pi \), but for a triangle inscribed on a surface, this is not generally true. The well-known Gauss-Bonnet theorem tells us that, for any triangle inscribed on a surface, the integral Gaussian curvature (also known as the total curvature), over the area of the inscribed triangle, is equal to the interior angle excess. This can be expressed as:

\[
\int Kda = (\theta_1 + \theta_2 + \theta_3) - \pi
\]  

(3.2)

In order to extend this idea to polygons, consider an arbitrary inscribed polygon having \( n \geq 3 \) sides, and \( n \) interior angles \( \phi \), as illustrated in Figure 3.4(a). This polygon can be subdivided into \( (n - 2) \) inscribed triangles as shown in Figure 3.4(b). The total curvature over the area of the polygon can be expressed as the sum of total
curvatures associated with each of those \((n-2)\) triangles:

\[
\int Kda = \sum_{T=1}^{n-2} \left((\theta_1 + \theta_2 + \theta_3) - \pi\right)
\]

\[
= \left(\sum_{T=1}^{n-2} (\theta_1 + \theta_2 + \theta_3)\right) - ((n-2)\pi)
\]

\[
= \left(\sum_{T=1}^{n-2} (\theta_1 + \theta_2 + \theta_3)\right) + 2\pi - n\pi
\]  

But, observe that, again with reference to Figure 3.4, the sum of all the interior angles \(\theta\) in all of the \((n-2)\) triangles is equal to the sum of the \(n\) interior angles \(\phi\) of the polygon:

\[
\sum_{T=1}^{n-2} (\theta_1 + \theta_2 + \theta_3) = \sum_{k=1}^{n} \phi_k
\]

Substitution into Equation (3.5) gives

\[
\int Kda = \left(\sum_{k=1}^{n} \phi_k\right) + 2\pi - n\pi
\]  

Now consider a triangle mesh digitization of the same (arbitrary) inscribed \(n\)-sided polygon as illustrated in Figure 3.5. Note that we have separately identified a central angle \(\alpha\) associated with each of the \(n\) triangles. Because these are flat triangles, we have

\[
\sum_{T=1}^{n} (\alpha + \theta_1 + \theta_2) = n\pi
\]
3.2. Curvature Estimators

which can be rewritten as

$$\sum_{T=1}^{n} (\theta_1 + \theta_2) = n\pi - \sum \alpha_n$$

(3.9)

Note that the left hand side approximates the sum of the inscribed polygon interior angles. Substitution into Equation (3.7) gives

$$\int K \, da \approx \left( n\pi - \sum \alpha_n \right) + 2\pi - n\pi$$

(3.10)

$$\approx 2\pi - \sum \alpha_n$$

(3.11)

To convert the total (integral) curvature into a curvature value that will be assigned to the central vertex in the triangle umbrella, we need to divide both sides of Equation (3.11) by the area over which the integration takes place. Keeping in mind that a curvature value is ultimately assigned at every vertex, and that the overall total area over which we integrate should be equal to the total surface area, we need to somehow apportion the area of each triangle face in the umbrella to each of its three vertices.

The required apportioning of area has been done by other authors in various ways; for examples of different area operators used in this context, see (Meyer et al. 2003) and (Jagannathan and Miller 2007). Perhaps the simplest approach is to assign $1/3$ of the area of each face $A$ to each of its vertices, and, of course, to sum the contributions from each face at common vertices. Dividing both sides of Equation (3.11) by this integration area gives

$$K \approx \frac{(2\pi - \sum \alpha_n)}{\sum A(f_n)/3} = \frac{3(2\pi - \sum \alpha_n)}{\sum A(f_n)} = \tilde{K}$$

(3.12)

This completes the derivation. □
Referring again to Figure 3.6, mean curvature at the point \( p_0 \) is estimated by

\[
\tilde{H}(p_0) = \frac{3}{4} \left( \frac{\sum \beta_n ||e_n||}{\sum A(f_n)} \right)
\]  

(3.13)

As was the case with the Gaussian curvature estimator, this estimator is also generally valid in the case of any adjacency point count greater than or equal to three.

This mean curvature estimator can be derived as follows:

We begin with the observations that, with any triangle mesh, the curvature of each (flat) face is zero, and the curvature along edges and vertices is undefined because the mesh is not smooth there. So, we consider the effects of smoothing each edge of the mesh using a small arbitrary radius cylinder placed such that mesh faces at the edge are tangent to the cylinder. Figure 3.6 shows an edge corner, in cross-section, smoothed by a circular arc (in bold) with length \( l \). Note that, because of the tangential placement of the cylinder, the central angle associated with this smoothing arc is exactly equal to the previously identified dihedral angle \( \beta \) between adjacent face normals. By elementary geometry we have

\[
l_n = \beta_n r
\]  

(3.14)

Now, since the mean curvature of a cylinder is everywhere \( 1/2r \) and the area of a smoothed edge is \( l \) times the length of the edge, the integral (total) mean curvature over a complete triangle umbrella is estimated by

\[
\int Hda \approx \sum_{k=1}^{n} \left( \frac{1}{2r} l_n ||e_n|| \right) = \sum_{k=1}^{n} \left( \frac{l_n ||e_n||}{2r} \right)
\]  

(3.15)
3.2. Curvature Estimators

Substitution using Equation (3.14) gives

\[ \int H \, da \approx \sum_{k=1}^{n} \left( \frac{\beta_n r \|e_n\|}{2r} \right) = \sum_{k=1}^{n} \left( \frac{\beta_n \|e_n\|}{2} \right) \]  

(3.16)

Similarly to the case of Gaussian curvature, we need to divide both sides of Equation (3.16) by the area over which the integration takes place. As before, we use the simple \(1/3\) area operator. However, with mean curvature, edge length also needs to be apportioned between vertices. Consistent with our area operator justification, we assign \(1/2\) the length of every edge to its adjacent vertices, which results in

\[ H \approx \frac{\sum_{k=1}^{n} \left( \frac{\beta_n \|e_n\|/2}{2} \right)}{\sum_{k=1}^{n} A(f_n)/3} = \frac{3}{4} \left( \frac{\sum_{k=1}^{n} \beta_n \|e_n\|}{\sum_{k=1}^{n} A(f_n)} \right) = \tilde{H} \]  

(3.17)

This completes the derivation.

Generally, in practical applications, only the point data is given. So, for Gaussian curvature, the angles \(\alpha_n\) and the face areas \(A(f_n)\) need to be calculated. Additionally, for mean curvature, the dihedral angles \(\beta_n\) and the edge lengths \(e_n\) need to be calculated. With reference to Figure 3.2, this can be done using vector operations as follows:

\[ \alpha_n = \cos^{-1} \left( \frac{(p_n - p_0) \cdot (p_{n+1} - p_0)}{\|p_n - p_0\| \|p_{n+1} - p_0\|} \right) \]  

(3.18)

\[ A(f_n) = \frac{\| (p_n - p_0) \times (p_{n+1} - p_0) \|}{2} \]  

(3.19)

\[ \beta_n = \cos^{-1} \left( \frac{((p_{n-1} - p_0) \times (p_n - p_0)) \cdot ((p_n - p_0) \times (p_{n+1} - p_0))}{\| (p_{n-1} - p_0) \times (p_n - p_0) \| \| (p_n - p_0) \times (p_{n+1} - p_0) \|} \right) \]  

(3.20)

\[ e_n = \| p_n - p_0 \| \]  

(3.21)

### 3.2.2 Surface Cut

The development of the estimators presented in this section starts with curvature estimation for (discrete) digitized planar curves.

Consider a planar curve contained in a surface, along with three points on the curve, as shown in Figure 3.7(a). Note that the unique plane determined by the three points qualifies as a cutting plane as defined in Section 2.2.1. Of course, for example, when working with a digitization that includes these three points, we can only estimate the position of the intermediate connecting points. One very straightforward
way to estimate the location of the missing detail is to connect the points with line
segments as illustrated in Figure 3.7(b).

Recall, from Section 2.1.2, that (signed) planar line curvature for smooth curves
is defined as incremental change in angle divided by incremental change in length.
This definition can be applied to a discrete digitization as follows:

With reference to Figure 3.7(b), the curvature at the point $p_2$ is estimated by

$$\tilde{k}(p_2) = \frac{\alpha}{(d_1 + d_2)/2}$$

(3.22)

where $d_1$ and $d_2$ are the distances from $p_2$ to $p_1$ and $p_3$ respectively. Note that the
total length $(d_1 + d_2)$ has been divided by two because some of this total will be
apportioned to curvature estimation at the points $p_1$ and $p_3$. The reasoning behind
this length operator is similar to the reasoning used for the face area and edge length
operators used in Section 3.2.1.

To make use of the $n$-cut mean curvature theory from Section 2.2.1 we require
digitizations in which equally spaced (by angle) cutting planes can be found. Fortunately,
surface digitizations are often acquired mechanically in a regular stepped
fashion that results in a symmetry which can be exploited. A common example is
illustrated in Figure 3.8 where the acquisition sequence results in a symmetrical
hexagonal point adjacency pattern.
3.2. Curvature Estimators

Figure 3.8: A regular acquisition pattern that results in hexagonal adjacency.

Figure 3.9: Estimator point selection for: (a) two cut, (b) three-cut.

Given hexagonal adjacency, both two and three equally spaced cutting planes can be placed by selecting points as illustrated in Figure 3.9. The curvatures associated with these cutting planes can then be used to estimate surface curvature. For the two-cut mean curvature estimator, we have

$$\bar{H}_{(2\text{-cut})} = \frac{\bar{\kappa}_1 + \bar{\kappa}_2}{2}$$

and, for the three-cut mean curvature estimator,

$$\bar{H}_{(3\text{-cut})} = \frac{\bar{\kappa}_1 + \bar{\kappa}_2 + \bar{\kappa}_3}{3}$$

These estimators have been derived under the assumption that the intersection of the cutting planes aligns with the surface normal as shown, for example, in Figure 3.10(a). When the cutting planes do not align with the surface normal, a correction, based on Meusnier’s theorem (see e.g. (Hermann and Klette 2007)), can be
applied, resulting in the following general form for the n-cut estimator:

\[
\tilde{H}_{(n\text{-cut})} = \cos \theta \left( \sum_{n} \tilde{\kappa}_{n} \right)
\]

(3.25)

where \(\theta\) is the misalignment angle as shown, for example, in Figure 3.10(b).

As was stated previously, in practical applications, only the point data is given. So, for the surface cut estimator, the angular advance \(\alpha\) and the segment lengths \(d_{n}\) need to be calculated. With reference to Figure 3.7(b), this can be done using vector operations as follows:

\[
\alpha = \cos^{-1} \left( \frac{(p_2 - p_1) \cdot (p_3 - p_2)}{||(p_2 - p_1)|| \cdot ||(p_3 - p_2)||} \right)
\]

(3.26)

\[
d_{n} = ||(p_{n+1} - p_{n})||
\]

(3.27)

Additionally, for misaligned cutting planes, \(\theta\) needs to be calculated. We present this calculation in several steps, beginning with an estimation of the surface normal vector as a weighted sum (by face area) of the adjacent face normals. With reference to Figure 3.2, this sum is

\[
v_1 = \sum 2 A(f_n) n_n
\]

(3.28)

\[
= \sum \left( ||(p_n - p_0) \times (p_{n+1} - p_0)|| \cdot ((p_n - p_0) \times (p_{n+1} - p_0)) \right)
\]

(3.29)
3.3. Estimators Summary

Note that, to minimize computation steps, the weighting used is double the face area. This is acceptable because normalization of $v_1$ occurs later.

Next, working with any two of the cutting planes, we calculate a cutting plane intersection vector. With reference to Figure 3.7 and using additional subscripts $a$ and $b$ to distinguish between the two cutting planes, we calculate

$$v_2 = \left( (p_2 - p_{a1}) \times (p_{a3} - p_2) \right) \times \left( (p_2 - p_{b1}) \times (p_{b3} - p_2) \right)$$  \hspace{1cm} (3.30)

Finally, since what’s required in Equation 3.25 is not $\theta$, but rather $\cos \theta$, with reference to Figure 3.10 we calculate

$$\cos \theta = \frac{v_1 \cdot v_2}{||v_1|| \cdot ||v_2||}$$  \hspace{1cm} (3.31)

3.3 Estimators Summary

The estimators presented in this chapter can be divided into three groups:

1) n-neighborhood triangle umbrella Gaussian,
2) n-neighborhood triangle umbrella mean,
3) n-cut mean, with or without compensation.

In practice, the application of these estimators will depend on identifying point neighborhoods and, for the n-cut mean estimator, identifying points which give equally spaced cutting planes. We note that the n-cut uncompensated mean estimator calculation has the least number of operations and would thus have the fastest execution time.
Part 2
Applications
Chapter 4

Similarity Curvature Experiments

In this chapter we further develop our new scale invariant curvature measure, similarity curvature. An estimator for the similarity curvature of digital surfaces is presented. Experiments and results applying similarity curvature to synthetic data are also presented.

4.1 Introduction

Recall, from Part 1 in this thesis, that similarity curvature is well-motivated by numerous well known practical 3D shape applications in computer vision including 3D scan matching, alignment and merging, 3D object matching, and 3D object classification and recognition. 3D object databases are also an active research area; see, for example, (Bustos et al. 2005), (Ichida et al. 2004) and (Assfalg et al. 2004).

We generally seek invariant properties when characterizing 3D objects. Any characterization that is rotation, translation and scaling invariant enables determining the equivalence of shapes independent of size. Again, as mentioned in Part 1, scale invariance would also enable the use of uncalibrated measurement units in 3D digitization (e.g. scanning).

4.1.1 Curvatures

Recall that a number of different curvature measures are defined in differential geometry. Curvature is well defined for continuously differentiable lines and surfaces (Davies and Samuels 1996). Planar lines have only a single curvature measure whilst surfaces have a number of curvature measures, all of which are based on normal curvature.
As given in Chapter 2 on surfaces, the two principal curvatures are $\kappa_1$ and $\kappa_2$, where $\kappa_1$ is the maximum normal curvature and $\kappa_2$ is the minimum normal curvature at a given point.

And, again, historically, the mean curvature has been given as

$$H = (\kappa_1 + \kappa_2)/2 \quad (4.1)$$

and the Gaussian curvature given as

$$K = \kappa_1\kappa_2 \quad (4.2)$$

More recently additional surface curvature measures have appeared. In (Jagannathan and Miller 2007) for example, the curvature measure *curvedness* has been defined as

$$C = \sqrt{(\kappa_1^2 + \kappa_2^2)/2} \quad (4.3)$$

None of the above curvature measures are scale invariant. Also, recall that, with digital data, there is inherent discontinuity and curvature can only be estimated.

### 4.2 A Similarity Curvature Estimator

From Section 2.3.2 similarity curvature is, by definition, determined by the principal curvatures. But, because a digitization acquisition pattern will not, in general, align with the principal curvature directions, we cannot directly compute the principal curvatures. However, recall, from Section 3.2.1 and Section 3.2.2 that we do already have estimators for mean and Gaussian curvature.

Building on this previous work, similarity curvature is estimated using the following process. First the mean and Gaussian curvatures are estimated, then the principal curvatures are calculated from the mean and Gaussian curvatures as follows:

$$\tilde{\kappa}_1 = \tilde{H} + \sqrt{\tilde{H}^2 - K} \quad (4.4)$$

$$\tilde{\kappa}_2 = \tilde{H} - \sqrt{\tilde{H}^2 - K} \quad (4.5)$$

Finally, the (estimated) similarity curvature is calculated from these principal curvatures using Definition 12 and Definition 13 as was given in Section 2.3.2.
4.3. Experiments

The validity of Equations (4.4) and (4.5) is easily verified. Substitution using Equations (4.1) and (4.2) gives

\[
H + \sqrt{H^2 - K} = \left(\frac{\kappa_1 + \kappa_2}{2}\right) + \sqrt{\left(\frac{\kappa_1 + \kappa_2}{2}\right)^2 - \kappa_1 \kappa_2} \quad (4.6)
\]

\[
= \left(\frac{\kappa_1 + \kappa_2}{2}\right) + \sqrt{\left(\frac{\kappa_1^2 + 2\kappa_1 \kappa_2 + \kappa_2^2}{4}\right) - \left(4\kappa_1 \kappa_2\right)} \quad (4.7)
\]

\[
= \left(\frac{\kappa_1 + \kappa_2}{2}\right) + \sqrt{\left(\frac{\kappa_1^2 - 2\kappa_1 \kappa_2 + \kappa_2^2}{4}\right)} \quad (4.8)
\]

\[
= \left(\frac{\kappa_1 + \kappa_2}{2}\right) + \frac{\kappa_1 - \kappa_2}{2} \quad (4.9)
\]

\[
= \frac{\kappa_1}{2} + \frac{\kappa_2}{2} + \frac{\kappa_1}{2} - \frac{\kappa_2}{2} = \kappa_1 \quad (4.10)
\]

and

\[
H - \sqrt{H^2 - K} = \left(\frac{\kappa_1 + \kappa_2}{2}\right) - \sqrt{\left(\frac{\kappa_1 + \kappa_2}{2}\right)^2 - \kappa_1 \kappa_2} \quad (4.11)
\]

\[
= \left(\frac{\kappa_1 + \kappa_2}{2}\right) - \sqrt{\left(\frac{\kappa_1^2 + 2\kappa_1 \kappa_2 + \kappa_2^2}{4}\right) - \left(4\kappa_1 \kappa_2\right)} \quad (4.12)
\]

\[
= \left(\frac{\kappa_1 + \kappa_2}{2}\right) - \sqrt{\left(\frac{\kappa_1^2 - 2\kappa_1 \kappa_2 + \kappa_2^2}{4}\right)} \quad (4.13)
\]

\[
= \left(\frac{\kappa_1 + \kappa_2}{2}\right) - \left(\frac{\kappa_1 - \kappa_2}{2}\right) \quad (4.14)
\]

\[
= \frac{\kappa_1}{2} + \frac{\kappa_2}{2} - \frac{\kappa_1}{2} + \frac{\kappa_2}{2} = \kappa_2 \quad (4.15)
\]

In this chapter, we have chosen (somewhat arbitrarily) to employ the triangle umbrella mean and triangle umbrella Gaussian curvature estimators. Recall that the Gaussian curvature is estimated by

\[
\tilde{K} = \frac{3(2\pi - \sum \alpha_n)}{\sum A(f_n)}
\]

and mean curvature is estimated by

\[
\tilde{H} = \frac{3 \sum \beta_n ||e_n||}{4 \sum A(f_n)}
\]

4.3 Experiments

Synthetic digital data has been created for a number of simple and compound objects. Each object was digitized by orthogonally scanning from above (in simulation) using the offset rasterization pattern shown in Figure 4.1(a), which results in
4. Similarity Curvature Experiments

Figure 4.1: Scan grid.

6-neighborhood (hexagonal) point adjacency as shown in Figure 4.1(b). The scan grid has a pitch dimension as illustrated in the same Figure. Note that, with this scanning method, only the portion of an object which faces towards the scanning source direction gets digitized.

4.3.1 Similarity Curvature Histograms

Experimental reference shapes included a sphere, cylinder, ellipsoid and torus. To evaluate the similarity curvature estimation, we considered each reference shape in turn with an initial scan pitch of 1.0, then a 10x scaling, and finally a 10x scan resolution. For the 10x scaling, both the scan pitch and the object size were increased by a factor of ten. For the 10x resolution, the scan pitch was decreased by a factor of ten and the initial object size was retained. Of course, this finer scan pitch increases the total number of scan points by a factor of approximately 100x.

Initial object sizes where chosen such that the resultant number of digitized points was somewhere in the range of sixty to one hundred. Subsequently, of course, the 10x scan resolution resulted in digitized point counts that were more than one hundred times greater.

Similarity curvature values where calculated at each scanned object point. Recall, from Section 2.3.3, that similarity curvature values can be uniquely placed on $e$-axis and $h$-axis positions in the similarity curvature space.

In this first analysis, the resultant similarity curvature values have been accu-
4.3. Experiments

Figure 4.2: Sphere $\varepsilon h$-histograms for reference, 10x scale, 10x resolution.

mulated for summary in associated $\varepsilon$-axis and $h$-axis histograms. Consistent with common practice, curvature values are indicated on each horizontal axis and point counts are indicated on each vertical axis.

Sphere results are shown in Figure 4.2. The reference sphere that was used has a radius of 5. Observe that the similarity curvature has the constant value of 1 in the $\varepsilon$ histograms regardless of scale. For a sphere, we would not expect any $h$-axis values. However, due to limited precision arithmetic, there are a small number (approximately 0.5 percent of the total count) of erroneous values in the high resolution $h$ histogram.

Cylinder results are shown in Figure 4.3. The reference cylinder that was used has a radius of 5 and a height of 4. As expected, in all cases the $\varepsilon$ and $h$ values are constant at zero. Recall that this zero similarity is expected whenever the Gaussian curvature is zero, as is the case for a cylinder.

Ellipsoid results are shown in Figure 4.4. The reference ellipsoid that was used has axes equal to 6 and 12. As expected, the $\varepsilon$ values are bounded by 1 and $(1/12)/(1/6) = 0.5$. As was the case for a sphere, we would not expect any $h$-axis similarity curvature values for an ellipsoid. However, again due to finite precision arithmetic, this
4. Similarity Curvature Experiments

Figure 4.3: Cylinder $eh$-histograms for reference, 10x scale, 10x resolution.

Figure 4.4: Ellipsoid $eh$-histograms for reference, 10x scale, 10x resolution.
time an error count equal to approximately 2 percent of the total number of points has accumulated in each of the $h$ histograms.

Torus results are shown in Figure 4.5. The reference torus that was used has an inner radius of 6 and an outer radius of 14, which implies an interior radius of $(14 - 6)/2 = 4$. As expected, based on the associated minimum and maximum curvatures of $1/14$ and $1/4$ respectively along the outer circumference, the $e$ values are bounded by 0 and $(1/14)/(1/4) \approx 0.29$. The associated minimum and maximum curvatures of $1/6$ and $1/4$ respectively along the inner circumference, result in $h$ values which are bounded by 0 and $(1/6)/(1/4) \approx 0.67$.

### 4.3.2 Shading Coded Similarity Curvature

In this section we assign color coding to similarity curvature values as was done in Section 2.4. An example color coding scheme for the $eh$-plot axis is shown in Figure 4.6 where negative $e$ values are pure green, positive $e$ values are pure red, negative $h$ values are pure blue and positive $h$ values are pure yellow. These similarity curvature shading coded values can be used to color each surface point on test objects for visualization purposes.
4. Similarity Curvature Experiments

Figure 4.6: Shading coded $eh$-axis.

Figure 4.7: Shading coded similarity curvature reference images: sphere, cylinder, ellipsoid, torus.

Shading coded similarity curvature images for each of the test reference shapes used in the previous section are shown in Figure 4.7. The constant curvature of the sphere and the cylinder as well as the smooth transition through curvature values in the curvature maps of the ellipsoid and the torus are readily apparent.

Figure 4.8 shows the case of a torus having an inner radius of zero. Note the $h$-axis wrapping near the center of the torus. The bright yellow color transitions change abruptly to bright blue when the $h$-axis similarity curvature wraps around, changing sign.
4.4 Conclusion

Similarity curvature measure has been refined with the introduction of an estimator. The estimator has been verified by testing a number of synthetic reference shapes. 

4.3.3 3D Object Detection

Similarity curvature can be used to identify and extract 3D shapes from within complex 3D scan scenes. We may wish to, for example, identify all (convex) spheres and spherical patches within a scene no matter what the sphere size or scan resolution.

We have constructed a test scan scene containing a surface with five each spherical, ellipsoid and toroid bumps, as well as five pits each having those same shapes, with some of the pits and bumps overlapping. Several representations of the scene are shown in Figure 4.9. Figure 4.9(a) is a shading coded depth image in which points closer to the scan source are white color shaded, and more distant points are shaded black. A color coded similarity curvature image is shown in Figure 4.9(b), in which spherical bumps are pure red and spherical pits are pure green. Finally, all of the spherical bump surface patches have been identified and color coded as white in Figure 4.9(c).

4.4 Conclusion

Similarity curvature measure has been refined with the introduction of an estimator. The estimator has been verified by testing a number of synthetic reference shapes. $eh$-histograms as well as a color shaded images have been presented. Experiments have demonstrated that similarity curvature can be used to characterize and identify simple synthetic digitized shapes within a complex synthetic test scene. Further work is anticipated to include applying similarity curvature measure to real world scans and addressing the issue of noisy data.
4. Similarity Curvature Experiments

Figure 4.9: Color shaded test scene images illustrating (a) depth, (b) curvature, (c) extracted spherical patches.
Chapter 5

Curvature Maps for Surface Analysis

In 1998 - 1999 a team of researchers from the Computer Science Departments at the University of Stanford and the University of Washington digitized a number of Michelangelo’s sculptures, including the David statue, using a custom designed laser triangulation scanner. The resultant data has been made available to the research community. This chapter begins with an exploration of the data structures and the inherent geometry associated with the David data-set. An estimation of surface curvature that exploits the structure and geometry of the data-set is described. Finally surface curvature maps are defined and several curvature maps of David are presented.

Figure 5.1: The large statue scanner acquiring data (Levoy 1999).
5. Curvature Maps for Surface Analysis

5.1 Introduction

Marc Levoy and his team have reported the extensive logistical and technical effort that went into The Digital Michelangelo Project\(^1\) (Levoy et al. 2000). They developed both significant customized hardware and software to achieve their goal (Pulli et al. 2002), (Rusinkiewicz and Levoy 2001b). The resultant data set has historical, cultural and technical significance (Bracci et al. 2004). Their large statue scanner is shown acquiring data in Figure 5.1.

A photograph of David’s face is shown in Figure 5.2 and a rendering (by the author of this thesis) of a 2mm resolution composite triangle mesh model from the Stanford archive is shown in Figure 5.3. Surface curvature detail is clearly visible in both Figures 5.2 and 5.3. Note however, that the visual cues in photographs and rendered images, from which apparent depth and surface curvature detail are discerned, are highly dependant on lighting intensity and lighting position as well as the reflectance properties of materials. In this chapter we seek to overcome the visualization limitations which are imposed by the dependency on lighting in standard

\(^1\)See Appendix A for additional detail on The Digital Michelangelo Project.
graphics rendering models.

As a practical application, the techniques and methods that we have developed were to a certain extent driven by the size, organization and structure of the David data-set.

5.2 The David Data-Set

To gain an appreciation of the magnitude of the David data-set, we observe that the height of the David statue is 5.17 meters and that the scanning was done with a sample spacing of 0.29mm. The resultant 1.93 giga-bytes of data, has been made available by Stanford in nine separate compressed files. The uncompressed data set represents approximately 1.1 billion 3D space points. There are a total of 6540 raw scan files collected into 515 groupings.

2Refer to Section A.1 for more detail on the David data-set.
Each scan was acquired over a fixed width of approximately 140mm and a variable height. A rendered image of the points from two adjacent overlapping scans is shown at the end of this chapter as Figure 5.10.

5.2.1 Data Structures

The David data-set provided by Stanford includes very little explanatory documentation. However, the C++ source code for the software developed by Stanford for the project (Pulli et al. 2002) was made available. This program code includes class definitions and data structures from which much can be discerned. Consider the following code fragment:

```cpp
class SDfile {
public:
    unsigned int pts_per_frame;
    unsigned int n_frames;
    unsigned short *z_data;
};
```

We see from this code listing, for example, that data is acquired line-by-line, that each scan line of acquired data is referred to as a frame, and that there is a constant number of points in each frame. Scan files 13 and 14 from the scan group Face1, for example, both have 486 points per frame and have 1480 frames 1515 frames respectively. The z_data array holds raw depth values.

The Stanford team report that, because the scanner uses an interlaced CCD sensor, the values in each frame actually represent an alternating acquisition pattern which effectively results in two interlaced horizontal scan lines in each frame (Levoy et al. 2000). This pattern and its implications are explored in the next section.

5.2.2 Data Geometry

The scanning system's physical geometry is illustrated in Figure 5.4. The Stanford software converts z_data depth values into 3D points using a concatenated sequence of transformation matrices modeled on the system geometry.

Thoughtful visual exploration of the scan points in 3D space (using software tools developed by the author of this thesis) reveals close local alignment with four
5.2. The David Data-Set

Figure 5.4: The scanner’s imaging volume (Levoy et al. 2000).

Figure 5.5: Two different 3D views of extracted surface points which intersect four (invisible) cutting planes.

As an example of this, we arbitrarily selected a point in scan 14 and extracted points from a subset of the scan which are intersected by four cutting planes through that same point. Figure 5.5 shows the extracted points from two different viewing directions. The cutting plane alignment
clearly visible in Figure 5.5(b) occurs from the viewing location which corresponds to the scanning acquisition position.

A close up view of the extracted surface points is shown in Figure 5.6. The dark black points are those associated with the frame that includes the center point.

Now note that the alternating pattern of frame acquisition results in a local hexagonal point adjacency. For data visualization purposes, this hexagonal pattern is somewhat problematic in that it cannot be directly displayed on the square grid associated with conventional computer displays. This problem can be overcome by using a slightly squashed array of smaller dots as shown in Figure 5.7.

This squashed dot mapping can be used to correctly generate displayable 2D arrays of the raw \(z\) data depth values. Depth maps of scans 13 and 14 are shown in
5.3. Curvature Maps

Successful production of depth map images from scans of the David statue, as was done in the previous section, suggests that similar images might be generated using curvature information.

5.3.1 Surface Curvature

Recall that surface curvature is a well-defined property for continuous smooth surfaces (Davies and Samuels 1996) but that, when working with point set data, surface curvature can only be estimated (Klette, Kozera, Noakes and Weickert 2006), (Klette and Rosenfeld 2004).

The curvature estimation approach used in this application is as follows:

For each point,
1) select points associated with two orthogonal cuts as illustrated in Figure 5.8;
2) for each cut, calculate an estimated signed planar line curvature;
3) take the mean of these curvatures as an estimation of the mean surface curvature;

In this application we have chosen the simplest (2-cut) curvature estimator to optimize for execution speed.
4) using squashed dot mapping (because of the hexagonal adjacency), convert the results into a displayable 2D curvature map;
5) apply standard 2D Gaussian image filtering to the curvature map.

In Step 2, with reference to Figure 5.9, the planar line curvature was estimated (as was introduced in Section 3.2.2) as the incremental angular advance divided by the incremental change in length

\[ \tilde{\kappa} = \frac{a}{(d_1 + d_2)/2} \]  \hspace{1cm} (5.1)

and calculated as

\[ \tilde{\kappa} = \frac{2}{||v_1|| + ||v_2||} \cos^{-1}\left(\frac{v_1 \cdot v_2}{||v_1|| ||v_2||}\right) \] \hspace{1cm} (5.2)

where \( v_1 = p_2 - p_1 \) and \( v_2 = p_3 - p_2 \).

In this curvature estimation off-axis correction has not been applied. Note that this curvature estimator corresponds to an uncompensated version of the 2-cut mean estimator presented earlier in Section 3.2.2.

Curvature map filtering techniques are described more fully, and extended, in the next chapter.
5.3.2 Results

Estimated mean surface curvature for scans 13 and 14 is shown in the curvature maps in Figure 5.12. Positive surface curvature is defined as that which bends away from the acquisition source or, equivalently in this case, curvature associated with viewing a convex hull from the outside. Maximum positive curvature is shading coded as white. Zero curvature is shading coded as medium grey and maximum negative curvature is encoded as black. Out of range points are shading coded as medium grey.

It is noteworthy that the curvature maps extract topological detail (such as the chip in the lower eyelid of David’s right eye!) that is certainly not as readily apparent in either the photographic image shown earlier in this chapter as Figure 5.2 or the rendered image shown in Figure 5.3.

5.4 Conclusion

Curvature maps have been introduced and applied in practice to important real-world data. The images produced using the curvature map technique in this chapter have been shown to be capable of displaying surface detail that is not otherwise as clearly visible when lighting dependant image creation is used (e.g. photographs and rendered images).

---

4The correction could have been done using Meusnier’s theorem (Hermann and Klette 2007).
Figure 5.10: Points from scans 13 and 14, rendered with lighting.
Figure 5.11: Depth maps of scans 13 and 14.
5. Curvature Maps for Surface Analysis

Figure 5.12: Curvature maps of scans 13 and 14.
In general, the noise that is present in real-world 3D surface scan data prevents accurate curvature calculation. In this chapter we show how curvature can be extracted from noisy data by applying filtering after a noisy curvature calculation. To this end, we extend the standard Gaussian filter (as used in 2D image processing) by taking adjacent point distances along the scanned surface into account. A brief comparison is made between this new 2.5D Gaussian filter and a standard 2D Gaussian filter using data from the Digital Michelangelo Project.

6.1 Introduction

Three dimensional objects are often digitized in a way that results in surface point data-sets (Visintini et al. 2007), (Hu et al. 2005), (El-Hakim et al. 2004). In general, real-world scan data is noisy due to inaccuracies accumulated in the scanning process. Curvature, being a second derivative property, is particularly sensitive to corruption by noise.

A common approach towards solving this problem is to smooth the point data prior to attempting curvature calculations. This approach has a shortcoming in that surface detail can easily be lost. Alternatively, in this thesis, we firstly calculate noisy curvatures and subsequently apply filtering to these curvature values.

Although the standard 2D Gaussian filter can be used in our approach (as was done in Chapter 5) it does have the undesirable effect of smoothing edges as well as noise. Edge preserving variants of the 2D Gaussian filter have been developed by others. For example, the bilateral filter, developed by (Tomasi and Manduchi 1998), is a 2D filter that employs an edge preserving term that decreases pixel weighting

\footnote{In this chapter, we will consider noise which has primarily a Gaussian distribution.}
based on pixel intensity differences.

Smoothing of the 3D shape itself is the goal of other noise reduction filters. A curvature and Laplacian operator based diffusion approach was introduced by (Desbrun et al. 1999). Work by (Fleishman et al. 2003) consists of a mesh de-noising algorithm that operates on a surface predictor geometric component of the mesh.

In contrast, even though the approach that is presented in this chapter uses 3D surface point position information, it does not alter the position of those points.

6.2 Curvature Maps

In this chapter we generalize by introducing some variation into the process of generating curvature maps that was given in the previous chapter. In any case, 1) curvature estimation, 2) dot mapping, and 3) filtering will be required.

In this chapter, we use the triangle umbrella mean curvature estimator as was given in Chapter 3. With reference to the left side of Figure 3.2 in Chapter 3, mean curvature at the point \( p_0 \) is estimated by

\[
\tilde{H}(p_0) = \frac{3}{4} \left( \frac{\sum \beta_n \|e_n\|}{\sum A(f_n)} \right)
\]  

Recall that this estimator is generally valid, without change, in the case of any adjacency point count \( n \) greater than or equal to three.

For 2D visualization purposes it is useful to convert the mean curvature values at surface scan points into a (2D) curvature map (see Chapter 5). If the 3D point data has been acquired in a 3D orthogonal grid, then the curvature mapping is straightforward (defined by orthogonal cuts parallel to coordinate planes, see (Hermann and Klette 2007)). For data that has been acquired in a hexagonal grid, a squashed dot mapping is used. Other mappings are possible. However, these are the only mappings considered in this thesis.

Mappings for both cases, either orthogonal or hexagonal, are shown in Figure 6.1. The second mapping has the expense of quadrupling the number of pixels, with the benefit of no loss in curvature detail.
6.3 Curvature Noise Filtering

Consider sampling the planar surface, which has zero curvature everywhere. Noisy sampled data points will exhibit a symmetrical distribution of positive and negative curvatures centered around zero, with the limiting mean value for many points being zero. This suggests filtering that includes a mean calculation.

In this chapter we describe and compare two different weighted mean based curvature noise filtering approaches.

6.3.1 2D Gaussian Filter

In this approach, we start by converting the noisy curvature values, associated with 3D space points, into a 2D curvature map as described in the previous chapter. This conversion into 2D enables the use of standard 2D image processing techniques (e.g. blur, sharpen). The reduction in dimension (from 3D to 2D) implies higher performance in subsequent processing.

Next we apply a standard 2D image processing Gaussian filter which performs the desired smoothing based on the now fixed adjacency (and distance) relationships in the curvature map. The standard 2D Gaussian filter is implemented as a convolution process with the terms in an \((2m + 1) \times (2m + 1)\) Gaussian convolution kernel centered at \((0, 0)\) being determined using the formula

\[
h(n_1, n_2) = h_g(n_1, n_2) / \sum_{n_1=-m}^{m} \sum_{n_2=-m}^{m} h_g
\]

with \(h_g(n_1, n_2) = e^{-(n_1^2 + n_2^2)/2\sigma^2}\) (6.2)
where, as usual, the standard deviation $\sigma$ acts as a pixel area smoothing factor. Note that the $(n_1^2 + n_2^2)$ term can be thought of as the adjacency distance (squared) in a fixed adjacency grid.

6.3.2 2.5D Gaussian Filter

As a preliminary motivation, note that the 2D Gaussian filter, with its innate fixed distance adjacency, has the undesirable effect of smoothing edges which may be present due to silhouettes, occlusions, and surface folding in the scan. For example, Figure 6.2 shows a cross-section view of a surface fold in which the distances between adjacent scan points are not equal.

We introduce a 2.5D Gaussian filter which gives consideration to edges. We will apply this filter to the noisy curvature values assigned to each scan point in the 3D domain before generating a final 2D curvature map.

In the 3D point space, we define adjacency point neighborhood rings and assign subscripts as shown in Figure 6.3 for the case of hexagonal adjacency. Note that only the inner two rings are shown, but that additional rings may be used. The first subscript identifies the ring and the second subscript identifies each point within a ring. The concept of neighborhood rings applies similarly to orthogonal adjacency where, of course, the number of points in each respective ring will be greater.

We calculate the 2.5D Gaussian filtered mean curvature $\tilde{H}_{2.5}$ at each point $p$ on
6.3. Curvature Noise Filtering

Figure 6.3: Adjacency point neighborhood rings.

The surface (indexed in turn as $p_{0,0}$) as follows:

$$\hat{H}_{2.5}(p_{0,0}) = \frac{\sum_{m} \sum_{n} (w_{mn} \tilde{H}_{mn})}{\sum_{m} \sum_{n} w_{mn}}$$

where the values of $H_{mn}$ are the unfiltered mean curvatures. The summations are computed over $m$ neighborhood rings with $n$ points in each ring.\(^2\) Note that $||p_{mn} - p_{0,0}||$ is the physical (Euclidean) distance from the center point to a point in a neighborhood ring, and that the values of $w$ can be thought of as representing filter weights. The standard deviation $\sigma$ is a smoothing factor that, in contrast with the 2D Gaussian filter, is now associated with (estimated) space distances. The standard deviation retains its usual meaning in that we would expect, for example, the sum of the filter weights assigned to points within a two-sigma radius to be 95.5% of the total filter weight.

We refer to this as a 2.5D filter because it is computed on a 2D surface which is embedded in 3D space. Although there is some similarity to the standard 2D filter, note that, strictly speaking, this is not a convolution process, and we have lost the computational efficiency associated with a fixed convolution kernel.

\(^2\) Including the center point $p_{0,0}$ as the single element in ring zero.
6.2.5D Filtering of Noisy Data

6.4 Experiments

We have performed experiments using scan data of the David statue from the Digital Michelangelo Project (Levoy et al. 2000). This data set was acquired with hexagonal point adjacency, and thus neighborhood rings as illustrated in Figure 6.3 were assigned for the computations. The data contains a moderate level of noise, with the equivalent Gaussian noise level standard deviation being approximately equal to the minimum scan point adjacency distance.

We concentrate on a curl of hair just above David’s right eye which is visible in the upper left of the photograph shown in Figure 6.4. Figure 6.5 is a reference closeup of the curl rendered as a mesh surface constructed from the raw scan data points. The lighting direction in this rendering has been chosen to highlight certain contours. The holes in the rendered mesh surface are due to occlusions in the scan. The black bar with ends marked A and B identifies a cut through a folded section of the surface.

Figure 6.6 shows a cross-section slice of the scan points which indicates a folding edge between locations A and B. Traversing from A to B, the curvature starts as slightly positive, is distinctly positive at the first bend, is then zero, is distinctly negative at the second bend, and then slightly negative.
6.4. Experiments

Figure 6.5: The curl: rendered mesh.

Figure 6.6: Cross-section of curl scan points.

Figure 6.7 shows an unfiltered (noisy) shading encoded mean curvature map of the curl. Maximum positive curvature is shading coded as white. Zero curvature is shading coded as medium grey and maximum negative curvature is encoded as black. Figure 6.8 shows the curvature map with 2D filtering. Note that, because
the filtering was done after the squashed dot mapping, there is some spread into regions where there is otherwise insufficient point data for curvature calculation.
Figure 6.9 shows a shading encoded mean curvature map of the curl with 2.5D filtering. Equivalent values for the smoothing factor $\sigma$ were used in both filters.

In Figure 6.9 it does appear that, as expected, there exists sharper transitions between black and white at the sharp fold edges. To confirm this, we extracted pixels associated with the previously illustrated A-B cut from both the 2D and the 2.5D...
filtered curvature maps. The extracted pixels are shown in Figure 6.10.

We averaged the pixel values across the narrow cut direction and then plotted the resultant values as shown in Figure 6.11. In the center section of the plot, we see that, in the 2.5D trace, 1) the maximum slope is greater, 2) the magnitudes of both the positive and the negative peaks are greater, and 3) there is a distinct zero curvature shelf.

6.5 Conclusion and Further Work

In the experiment presented, 2.5D filtering results in more representative curvature at a fold edge than does 2D filtering. Further work is anticipated to include additional noise models (such as highly impulsive), additional filtering methods, and additional visualization techniques.

Finally, to illustrate the total size and scale of the scan, Figure 6.12 shows two curvature maps of the entire scan, one with 2D filtering and the other with 2.5D filtering. Close examination reveals that other regions, such as the eyelid near the corner of the right eye, for example, appear to benefit from the improved edge preservation associated with 2.5D filtering.
Figure 6.12: David’s face: 2D (left) and 2.5D (right) filtering.
6. 2.5D Filtering of Noisy Data
Chapter 7

Surface Registration Markers

In this chapter, we introduce a data processing pipeline designed to generate registration markers from range scan data. This approach uses curvature maps and histogram-templates to identify local surface features. The noise associated with real-world scans is addressed using a 2D Gaussian filter and expansion segmentation. Experimental results are presented for data from The Digital Michelangelo Project.

7.1 Introduction

There are a number of challenges associated with 3D range scan digitizations (or 3D surface reconstructions in general) of real-world objects. At first the captured data should be accurate, and second the various data sets need to be unified into one consistent surface model. Usually only a partial section of an object surface is acquired with each scan, and a number of scans need to be aligned and subsequently merged together. Especially when each scan is taken from a different uncalibrated viewpoint, aligning the scans can be time consuming (Levoy 1999), (Pulli et al. 2002).

There exists a number of algorithms (Brown and Rusinkiewicz 2004), (Chang et al. 2004), (Rusinkiewicz and Levoy 2001a), (Pulli 1999), including the widely used Iterative Closest Point algorithm (ICP), that can refine a given rough alignment of scan pairs, two at a time. Initial matching and alignment of multiple (possibly many) 3D scans is still largely an open problem. Since range scans normally already each have the same scaling, alignment to some fixed reference points involves a linear transformation consisting of a 3D translation and a 3D rotation. Exact alignment is in practice not possible due to the noise in real-world data. Input data uncertainty is also present when surface patches have been generated via less reliable computer vision 3D surface recovery techniques such as photometric stereo or structure from motion (see, e.g., (Klette et al. 1998) for 3D surface recovery techniques).
Early work used 2D features, such as contour based grouping using relaxation methods (Parent and Zucker 1989). For the visualization of implicit surfaces we cite (Balsys and Suffern 2001). More recent work in (Gatzke et al. 2005) uses (what they refer to) as a “curvature map method” to characterize a local signature for every point in a scan. This can be seen as a continuation of surface characterizations by Gauss maps in differential geometry, or by extended Gaussian images (see, e.g., (Sun and Sherrah 1997)) in computer vision.

In this chapter we present a matching and initial alignment approach based on identifying a small number of registration markers. The registration markers that we generate are based on local surface curvature features. An advantage to using surface curvature is that it is rotation and translation invariant. However, one problem that needs to be overcome is that curvature, being a second derivative property (Davies and Samuels 1996), is very sensitive to local noise.

The sequence of steps in the processing pipeline that we use for surface marker generation is to 1) calculate (noisy) curvatures from the point data, 2) filter and segment the curvature data, and then 3) identify and mark local features using the curvature data. We illustrate the application of this sequence of steps to multiple scans by example. For each step in this processing pipeline, we select and define appropriate methods and algorithms.

As a particular difficulty, in practice, scan overlap regions can be small relative to the scan size, and thus our registration markers need to be smaller than the scan overlap. We keep, from the outset, the demands of extensive real-world data sets in mind. Finally, we demonstrate our approach on data from The Digital Michelangelo Project (Levoy 1999), (Levoy et al. 2000).

### 7.1.1 Curvature and Curvature Estimation

In this chapter, the uncompensated orthogonal cut method (see Chapter 3) is used to calculate an estimated mean curvature. For curvature estimation, for each scan point, we firstly identify the four nearest neighbor points associated with two orthogonal planar cuts. Next, for each of the two cuts, we calculate an estimated signed planar line curvature. Finally, we take the mean of these two curvatures as an estimate of the mean surface curvature (see Section 3.2.2).

In practice this results in a (noisy) mean curvature value associated with each
7.1. Introduction

3D scan point. Other curvature measures can be used, however, the mean curvature gives results that correspond to actual image appearance of the object and thus is the most relevant feature for our purposes.

7.1.2 Curvature Maps, Filtering and Segmentation

In this section, we convert the mean curvature data at surface scan points into a (2D) curvature map, which is an array of the same dimension (ignoring squashing) as the given 2D scan array, where values are mean curvatures at scan points (see Chapter 5). There are a number of advantages when using such curvature maps, including the possibility of visualization and processing using 2D image processing techniques. Pixels in 2D images are, by standard convention, stored in 2D arrays and we apply adjacency definitions based on the related orthogonal grid. If the 3D point data has been acquired in a 3D orthogonal grid, then the curvature mapping is straightforward, being defined by orthogonal cuts parallel to coordinate planes (Hermann and Klette 2007).

Data acquired in the Michelangelo project uses a hexagonal adjacency (i.e., 6-adjacency in the image plane) in a virtual projection plane, defined by the order and geometry of scan acquisition. In such a case we can use a special squashed dot mapping as defined in Section 5.2.2.

Mappings for both cases, either orthogonal or hexagonal, are shown in Figure 7.1 with the reverse mapping as shown in Figure 7.2. The second mapping has the expense of quadrupling the number of pixels. (We do not discuss further scan acquisition geometries in this thesis.)

We then apply a 2D Gaussian filter\(^1\) to the curvature map to reduce noise. As an

---

\(^1\)This standard simple filter was selected for computation speed performance reasons.
illustration of the effectiveness of this technique, consider noisy data from a sampled planar surface. The noisy data points will (incorrectly) exhibit a symmetrical distribution of positive and negative curvatures centered around zero, with the average value being (correctly) zero. The Gaussian filter performs exactly the desired spatial averaging. Note that we are addressing the noisy data problem in 2D rather than in the original 3D domain. The terms in an $n_1 \times n_2$ Gaussian convolution kernel centered at $(0, 0)$ are determined using the standard formula

$$h(n_1, n_2) = \frac{h_g(n_1, n_2)}{\sum_{n_1} \sum_{n_2} h_g} \text{ with } h_g(n_1, n_2) = e^{-\left(n_1^2 + n_2^2\right)/2\sigma^2}$$

where, in this application, $\sigma$ is effectively a smoothing area factor.

The filtering process correctly reduces the extremes in the distribution of mean curvature values. The resulting mean curvatures are more representative of the actual surface curvature, which is concentrated in a much smaller range around zero. Now, in order to reveal previously obscured detail, we perform a linear expansion centered around this zero-curvature region of interest.

Multi-threshold segmentation of the expanded data is then performed to partition the data into a number of curvature bins. This data reduction enables the approach presented in the next section. An example of how we perform a combined expansion-segmentation operation is given in the experiment section later in this chapter.

Also, in preparation for local feature identification, we reverse the 3D to a 2D curvature mapping process, by selecting pixels, depending on the original adjacency as illustrated in Figure 7.2 and update the mean curvature associated with each of the 3D scan points with its filtered and expansion-segmented value. The motivation for this can be explained by saying that, even though we have temporarily moved into

Figure 7.2: Reverse mappings of (a) orthogonal, or (b) squashed hexagonal grids into the original grid.
7.2 Local Feature Identification

We use histogram-templates to search through the filtered and segmented curvature data for local surface features. A histogram-template consists of (i) a sliding window that is segmented into subsets and (ii) a set of histograms, one for each subset. Each histogram accumulates curvature counts into a number of curvature bins. The number of histogram curvature bins is assigned to be exactly the same as the number of segmentation curvature bins in the previous section.

Note that, even though we are now working within the 3D dataset, we do the histogram-template search in a 2D fashion using the 2D orthogonal grid.

The histogram-templates are carefully designed (at the given scale of scan data) to characterize local surface features; a multi-scale approach would be relevant if uncertainty increases for the given scale of scan data. An example histogram template is shown in Figure 7.3.
7. Surface Registration Markers

In the example, the histogram-template is designed to characterize pit-like surface defects which have radial symmetry. The template indicates that four histograms, indexed zero through three, are tabulated for each selected 3D data point. Histogram two is the center region which should contain negative curvature values, histogram one is the pit perimeter which should contain positive curvature values, histogram zero is a guard ring which should contain low curvature magnitude values, and histogram three is a “don’t care” region. Individual histogram bins are assigned such that first bin accumulates the count of largest negative magnitude curvatures and the last bin accumulates the count of largest positive magnitude curvatures.

The histogram-template is used to search through the curvature values looking for the desired feature match.

7.3 Performance Evaluation

The value of algorithm performance evaluation in the field of computer vision has been discussed in (Klette, Stiehl, Viergever and Vincken 2006). A possible starting point for performance evaluation is to use a data set having known indisputable characteristics. These indisputable characteristics can be used to establish what is referred to as a ground truth.

7.3.1 Synthetic Data

One way to insure the existence of ground truth in a data set is to model and generate synthetic data having known properties. In our case we have chosen to synthesize data using a hexagonal adjacency orthogonal projection grid. Because curvature is scaling dependent, we needed to establish numerical distance values for our grid. Figure 7.4 shows the values that we used. We will refer to the illustrated vertical dimension as the resolution of the scan.

The synthetic objects that we “scanned” consisted of a planar surfaces having spherical indentations and spherical bumps. The indentation model is shown in Figure 7.5. Note that the sphere cuts into the plane by a distance of one half the

---

2Excluding those close to the array’s border.
3These values closely match those used in the real-world data that we explore later in this paper.
radius. Spherical bumps are modeled similarly.

7.3.2 Performance Measures

We evaluated our curvature estimator in the presence of varying levels of Gaussian noise applied to scan data depth values. We defined noise level by a scale factor relative to the scan resolution. A noise scale factor of one means that the noise sigma value is equal to the resolution. A factor of two means that the noise sigma value is equal to the resolution divided by two, and so on. Of course, the actual curvature of a planar surface is zero and the curvature at any point on a sphere is a constant equal to the reciprocal of its radius.

Figure 7.6 illustrates the performance of the 2-cut curvature estimator in the presence of varying levels of noise. Positive curvature is shading coded as white. Zero curvature is shading coded as medium gray and maximum negative curvature is encoded as black. Thus, bumps are shown in the top half of the figure and indentations in the bottom half of the figure.

Figure 7.7 illustrates the performance of the filtering and segmentation process,
7.3.3 Feature Identification Evaluation

We applied the complete feature identification process to a scan of a more comprehensive synthetic object. The object has sixteen indentations associated with spheres having radii which vary from 0.1 to 2.0. Figure 7.8 illustrates the results pictorially. Identified features are each marked with five red pixels. Note that because the feature identification process searches through (nearly) every image pixel, the same feature is usually hit and marked more than once.

The results from Figure 7.8 are summarized in Table 7.1. We can make a number of observations. Increasing noise levels tend to cause the process to associate a smaller size to the feature. The fixed size matching template does in fact identify a
7.4 Experiments and the David Dataset

We have also tested our approach using the extensive David dataset from the University of Stanford Digital Michelangelo Project (Levoy 1999), (Levoy et al. 2000).
The David data set consists of 1.93 giga-bytes of data which has been made available in nine compressed files. The uncompressed dataset represents approximately 1.1 billion 3D space points. There area total of 6,540 raw scan files collected into 515 groupings. Each scan was acquired over a fixed width of approximately 140mm and a height of generally no larger than 600mm. The David statue is over five meters tall.

The scanner is shown (earlier in this thesis) in Figure 5.1 acquiring data using structured lighting, and a rendered image of the points from two adjacent overlapping scans is shown in Figure 7.9. It is not uncommon for a scan to contain upwards of 800,000 points.

The scanning system’s physical geometry is illustrated in Figure 7.10. The cyan and yellow colored boxes indicate volume regions associated with possible individual scans.

The University of Stanford group responsible for the project undertook a rather time consuming initial manual alignment of the individual scans. They reported both on this process and the need for an automated global matching method (Levoy et al. 2000). The surface identification markers described in this paper are candi-
7.4. Experiments and the David Dataset

Figure 7.10: The scanner’s imaging volume (Levoy 1999).

![Diagram showing the scanner's imaging volume](image)

Figure 7.11: Top left corner of the scanning sequence.

![Diagram showing the top left corner of the scanning sequence](image)

dates for a feature based global matching technique.

### 7.4.1 Data Structure

Each scan is acquired in a regular sweeping pattern of scan lines, left-to-right, and top-to-bottom. Each scan line consists of an alternating pattern of 486 points as shown in Figure 7.11. The scanning pattern results in a hexagonal grid. Of course,
the scanner does not find the reflective surface of an object at all points, and this is illustrated, for example, in Figure 7.11 where the white dots represent the surface of an object and the darker dots mean that nothing was found in that direction.

The regular scanning pattern suggests 6-adjacency (based on the hexagonal grid) shown in Figure 7.12. A “-1” entry in the array means that no surface point was found. A non-negative integer entry is an index value into another array of surface point data including its 3D location and a not yet calculated mean curvature value.

7.4.2 Curvature Estimators, Filtering and Segmentation

Since the 3D data points have 6-adjacency within each scan, the nearest neighbor orthogonal planar cut points are selected as shown in Figure 7.13 (dark squares),

---

5Note that, particularly in the case of data having 6-adjacency, a three-cut mean calculation is also possible. Experiments using this more expensive method gave similar results to those presented in this chapter.
7.4. Experiments and the David Dataset

Figure 7.14: Noisy mean curvature map and its histogram.

and the squashed dot mapping is used as previously discussed in this chapter.

An eight-bit precision mean curvature map of a section of a scan of David’s face is shown in Figure 7.14 along with a histogram of that entire scan. Positive surface curvature is defined as that which bends away from the scanning source or, equivalently in this case, as that curvature associated with viewing a convex hull from the outside. Maximum positive curvature is shading coded as white. Zero curvature is shading coded as medium gray and maximum negative curvature is encoded as black. Surface detail in this image is obscured by noise.

For Gaussian filtering of the David data, a sigma of four is large enough to smooth noise but not so large as to remove surface detail. A sample Gaussian convolution kernel matrix having dimensions eight and sigma four is:

\[
\begin{bmatrix}
0.0099 & 0.0119 & 0.0135 & 0.0144 & 0.0144 & 0.0135 & 0.0119 & 0.0099 \\
0.0119 & 0.0144 & 0.0163 & 0.0174 & 0.0174 & 0.0163 & 0.0144 & 0.0119 \\
0.0135 & 0.0163 & 0.0185 & 0.0197 & 0.0197 & 0.0185 & 0.0163 & 0.0135 \\
0.0144 & 0.0174 & 0.0197 & 0.0209 & 0.0209 & 0.0197 & 0.0174 & 0.0144 \\
0.0144 & 0.0174 & 0.0197 & 0.0209 & 0.0209 & 0.0197 & 0.0174 & 0.0144 \\
0.0135 & 0.0163 & 0.0185 & 0.0197 & 0.0197 & 0.0185 & 0.0163 & 0.0135 \\
0.0119 & 0.0144 & 0.0163 & 0.0174 & 0.0174 & 0.0163 & 0.0144 & 0.0119 \\
0.0099 & 0.0119 & 0.0135 & 0.0144 & 0.0144 & 0.0135 & 0.0119 & 0.0099 
\end{bmatrix}
\]

A Gaussian filtered version of the previous curvature map is shown in Fig-
Figure 7.15: Gaussian filtered curvature map and its histogram.

Note that there is now not much visible detail and that the levels are now mostly concentrated around the middle medium gray as shown in the associated histogram.

As mentioned earlier in this chapter, we need to expand the middle histogram region to recover detail. Because of the eight-bit precision used, the middle region contains only a small discrete range of integer values. We perform a linear expansion around the center (medium gray) and clip the results (to full-black and full-white). The result, in this example, is an image containing pixels having only eleven different integer intensity values.

It is significant to note that, in general, this limited precision expansion will always also simultaneously multi-level segment the data into a limited number of segmentation bins.

An expansion segmented version of the previous mean curvature map is shown in Figure 7.16. Curvature detail is now readily apparent. The histogram shows the range of eleven segmentation values.

We now reverse the squashed dot mapping process, by selecting pixels as previously illustrated in Figure 7.14 and update the mean curvature associated with each 3D point with its filtered and expansion-segmented value.
7.4. Experiments and the David Dataset

7.4.3 Feature Identification

There are numerous small (approximately 2-4mm diameter) surface ‘pits’, due to weathering and abuse, found scattered on the surface of the David statue\footnote{This can be verified by rendering images of triangle mesh versions of the scan data with specular reflections and appropriately placed lighting.} that are ideal candidates for surface markers. The pit-like features certainly meet the requirement of being smaller than scan overlap regions. One of these pits is clearly visible in Figure 7.16. A pit can be characterized by its center of negative curvature and a rim of positive curvature.

Experimental comparison showed that it was possible to reduce the histogram template search time by employing data thinning, as illustrated in Figure 7.17, where...
only every other row is examined\textsuperscript{7}. This thinning also results in pixels defined in
an orthogonal grid, which, rather conveniently, means that we can use a rectangular
search template.

We used exactly the histogram template given earlier as an example in Figure \textsuperscript{7.3}
to search through the curvature data for pit surface features. In this particular ex-
periment, the non-square dimensions of the template compensate for the unequal
aspect ratio of the rectangular data pixel subset. In fact, this template maps back to
a (nearly) square region in the 3D dataset.

The identification test for pit features that we used with the David dataset is a
minimum of twelve black (fully negative curvature) values in the center region, a
minimum of four white, or nearly white (very positive curvature) values in the rim,
a maximum of four black or nearly black (very negative curvature) values in the
guard ring, and a maximum of two white (fully positive curvature) values in the
guard ring. This identification test was able to identify pit surface features as re-
ported in the next section.

\subsection*{7.4.4 Some Feature Mapping Results}

We illustrate the application of the example histogram template to four different,
but overlapping, scans of David. The results are shown in Figure \textsuperscript{7.18}. The first and
third scans were acquired from the same view-point. The second and forth scans
were acquired from different view-points. Note that there is at least one feature that
links each scan to at least two other scans. For example, there is a feature just below
the right eye that links scan one to each of scans two and three.

Meticulous (and time consuming!) exploration of full resolution 3D triangle
mesh renderings (by the author of this thesis) has confirmed that common features
in different scans have been successfully identified and marked using our auto-
mated processing pipeline. Because these registration markers are associated with
the 3D dataset, additional 3D marker characteristics, such as the size of the pits, can
be calculated.

\textsuperscript{7}The sampling frequency appears to be at least double that of the Nyquist frequency associated with
the smallest underlying surface features, and thus, there is no aliasing associated with data thinning by
a factor of two.
We have demonstrated the effectiveness of data smoothing after a shading encoded mean curvature calculation on noisy data. In general, this approach reduces complexity in that it moves the smoothing process from the 3D domain into the 2D domain. We have also demonstrated the effectiveness of using 2D (rotation invariant) radially symmetric matching templates to identify 3D surface features.

Further work is anticipated to include additional marker characterizations and
using curvature based registration markers to automatically generate an initial alignment of diverse overlapping scans of the same object. Using different curvature measures (besides the mean curvature) is also an opportunity for further work.
Chapter 8

Projecting Surface Curvature

High resolution 3D range scan digitizations often produce data sets that, when directly converted to 3D models, are too large to be rendered quickly for display. The usual solution to this problem is to sacrifice detail and reduce the model size by simplification (Brodsky and Pedersen 2003), (Isenburg et al. 2003), (Shaffer and Garland 2001). In this chapter, we propose a method whereby high resolution detail can be merged with a simplified mesh model for visualization purposes.

8.1 Introduction

Prior work by other authors includes a number of techniques for adding surface detail to mesh models. Texture mapping is a widely used technique which requires an explicit mapping from the texture to the model; see, for example, (Sheffer and de Sturler 2002) and (Lévy 2001). Prior work in conveying the appearance of shape includes suggestive contours (DeCarlo et al. 2003) along with other types of non-photorealistic rendering (DeCarlo and Rusinkiewicz 2007), (Xu and Chen 2004).

In the approach presented in this chapter, high resolution detail is extracted from the raw scan data by producing curvature maps which are calculated from the original scan data associated with each scan view point. These curvature maps are then projected onto a simplified model in a way that is analogous to using slide projectors to project multiple photographs onto a 3D object.

As an illustration of this approach, consider the rendered images of a simplified mesh model of Michelangelo’s David (Levoy et al. 2000) shown in Figures 8.1, 8.2, 8.3. The rendering shown in Figures 8.1 uses a flat shading technique in which the underlying triangle mesh structure of the model is readily apparent.

The rendering shown in Figures 8.2 uses a smooth shading technique in which the underlying triangle mesh structure of the model is same, but the shading values
are smoothly interpolated over each triangle face.

In the rendering shown in Figures 8.3, a projected strip of surface curvature detail has been added. The underlying structure of the model is again the same.

There were a number of problems associated with this approach that needed to be solved. Generally raw scan data contains noise, and this noise destroys the possibility of direct curvature calculation. Also, the standard 3D graphics lighting projection model is not appropriate because it does not handle the unavoidable projection overlap seamlessly.

To convey some idea of possible noise magnitude, consider Figures 8.4 and 8.5. Figure 8.4 shows a subset of the raw scan points centered at the outside corner of David’s right eye. Some surface detail is apparent in this rather dense view. The closer edge-on view of a slice through a portion of the data shown in Figure 8.5 highlights the data noise level.
Figure 8.2: Michelangelo’s David: triangle mesh (smooth).

Figure 8.3: Michelangelo’s David: with a strip of projected curvature.
8. Projecting Surface Curvature

8.2 The Processing Pipeline

The source material in this application is multiple range scans, each consisting of high resolution raw data points. Point adjacency is inferred from the scan rasterization pattern. The point adjacency information is used in turn to produce a high
8.2. The Processing Pipeline

Figure 8.6: Shading encoded noisy curvature map.

resolution 3D triangle mesh model of each scan. Note that each range scan has an associated acquisition viewpoint position and direction in 3D space; this is used later.

We also assume the existence of a single complete lower resolution simplified 3D triangle mesh model. There are a number of techniques for merging multiple scans and creating such simplifications; see, for example, (Ezra et al. 2006), (Gelfand et al. 2005) and (Li and Guskov 2005).

We handle the noisy data problem by firstly doing a noisy mean curvature calculation on the raw scan data. This application uses the uncompensated orthogonal cut method to calculate a mean curvature as previously described in Section 3.2.2. A curvature map is produced by mapping a shading encoded version of these (noisy) curvatures into the 2D domain. An example of a noisy curvature map is shown in Figure 8.6.

Standard 2D image filtering and segmentation is then applied. Example results are shown in Figure 8.7. The results are mapped back into the associated high resolution scan triangle mesh in the 3D domain, as vertex colors. An example is illustrated in Figure 8.8. Notice that there are some holes in this rendered image due to
Figure 8.7: After applying 2D image filtering and segmentation.

Figure 8.8: Results back into vertex colors in the 3D domain.
occlusions in the original scan.

A perspective correct final projection map is created as a standard vertex colored single frame rendering of the high resolution mesh. This rendering is done with a camera position and orientation matching that from which the scan was originally acquired. Additionally, the camera field of view is chosen to insure that the entire high resolution mesh is in-frame. An example of a complete filtered and segmented curvature map as well as its projection map, is shown at the end of this chapter, in Figure 8.13.

Ultimately, the projection map is used as a projection slide to modulate a spotlight which illuminates a lower resolution model. Note that, to ensure proper registration, the spotlight needs to have the same location, orientation, and field of view as the camera that was used in the creation of the projection slide.

Many spotlights from many directions can be used to color-in additional surface detail on the low resolution model. Of course, any scene which contains the low resolution model and the spotlights can be animated, or single frames can be rendered from any view-point.

### 8.3 Seamless Projection

A goal from the outset was to, as much as possible, use existing 3D graphics techniques. However, the standard lighting projection model is additive for multiple sources (Foley et al. 2002). The color reflected from each point on a surface includes a single ambient contribution as well as a sum of specular and diffuse contributions over all light sources:

\[
I_{\lambda} = \text{Ambient} + \sum \text{Specular} + \sum \text{Diffuse}
\]  

(8.1)

We extend the standard lighting model by introducing a multiple projection source term as follows:

\[
I_{\lambda} = \text{Ambient} + \sum \text{Specular} + \left( \frac{\sum_{i=1}^{n} \text{Projection}}{n} \right) \sum \text{Diffuse}
\]  

(8.2)

The diffuse contribution from each of the projection sources is averaged into a total value that is used to modulate standard diffuse lighting value. Experiments support the claim that this new model produces seamless projection.
8. Experiments

We tested our approach on the David dataset from the Digital Michelangelo Project (Levoy et al. 2000). An image showing projected surface curvature has already been shown at the beginning of this chapter as Figure 8.3. Rendered images that specifically illustrate the success of our averaging lighting model are shown in Figures 8.9 and 8.10.

Because the curvature maps were created outside any lighting model, and the fact that curvature itself is rotation and translation invariant, the projected surface curvature detail retains a unique clarity in animated visualization. See the video file (psc.wmv) on the CD-ROM which accompanies this thesis. Two frames extracted from the video are shown in Figures 8.11 and 8.12.
Figure 8.10: The new lighting model, seamless scan strip overlap.
Figure 8.11: A patch of projected curvature on the front of David’s left knee.
Figure 8.12: Projected curvature detail on the back of David’s left knee.
Figure 8.13: A complete curvature map and a perspective correct final projection map.
Chapter 9

Identifying and Visualizing Surface Detail

In this chapter, we present the results of experiments in which we have identified, characterized, and produced visualizations of selected fine surface detail on Michelangelo’s David statue. Starting with available raw scan data (Levoy et al. 2000), we have applied a number of techniques, both developed and refined by us, including the calculation of curvature maps, 2.5D spatial noise filtering, texture projection merging, and image processing assisted physical measurement.

9.1 Introduction

Computer based visualization is an important and widely used tool in scientific research (Alexander and Wang 2006), (Johnson July-Aug. 2004). The large body of scientific visualization work by others includes that which specifically relates to 3D surfaces; see, for example, (Zhang et al. 2007), (Bonneau and Gerussi 1998) and (Mandal et al. 1997).

The work presented in this chapter began with existing raw scan data, as well as an existing complete simplified 3D triangle mesh model of Michelangelo’s David statue. The scans came with alignment information which enabled the use a common coordinate system shared by both the scans and the mesh model. Data processing began with the raw scan data which, as supplied by Stanford, is held in 6540 sweep files organized in a hierarchical directory structure consisting of 9 groups, 30 dates, and 515 scan directories in which the individual sweep files can be found. The sweeps contain over 1.1 billion individual 3D points.

Having never viewed the real statue, we specifically searched the scan data for fine surface features reported by the group of scientists responsible for recent

1Refer to Section A.1 for more information on the David data-set.
restoration work (Bracci et al. 2004). The group reported the existence of detail including chisel marks left on the statue by Michelangelo, graffiti, and thin stress fractures, however, the report did not include any images of these details. The location of a number of fracture lines was provided, but given only loosely for the chisel marks (“under protective shelves of hair on head”) and the graffiti (“confined to lower legs”).

Note that, in fact, with our surface visualization techniques, no photographic images of the statue were used at any time.

In this chapter, curvature maps, as previously described in Chapter 5, are used extensively. The curvature maps have been generated using a number of different surface curvature estimators including the new three-cut mean estimator described in Section 3.2.2. Curvature maps have been processed using an assortment of different filtering techniques, including the new 2.5D spatial noise filter described in Chapter 6. Curvature based surface texture was merged with the simplified mesh model using the projection technique described in Chapter 8. Multiple projections were seamlessly merged using the technique also described in that same chapter.

Final output after processing includes a collection of curvature map images, single frame rendered images of the mesh model including surface curvature texture, and, for 3D visualization, a video of an animated scene which contains the mesh model.

9.2 Processing Tools and Pipeline

We have produced software tools including a 3D point visualization tool (K-Scan) and a number of batch scriptable utilities. K-Scan was used for interactive exploration, analysis, import/export, and 3D stereo viewing. Scriptable access to modules within K-Scan is possible. Software utilities include file conversion, 3D bounding box calculation, 3D point data searching, and directory tree walking.

The first goal was to create a curvature map image library which could be browsed for initial feature search and identification. The 6540 sweep files as provided by Stanford included all of the data that they collected, some of which was problematic for a number of reasons including false starts, bad calibration, and mechanical problems. We produced a bad sweeps list, in part manually, but primarily using
9.2. Processing Tools and Pipeline

scriptable automatic criteria such as a good sweep always contains a certain minimum number of scan points.

After identifying and removing bad sweeps, we created a collection of 6355 2D curvature map reference images, one for each good sweep file, using K-Scan in batch mode. An image browsing tool was used for initial visual identification of interesting sweeps. Some extracts from the curvature map image library are shown in Figure 9.1. In the curvature map images, positive curvature is gray-scale shaded towards white, negative curvature towards black, and zero curvature as medium gray. (A number of the curvature map reference images can be found on the CD-ROM which accompanies this thesis.)

The next task was, given any feature, find all sweeps containing that feature. This was accomplished as follows. First, the 3D coordinates of the interesting features were determined by viewing the sweeps associated with each feature in K-Scan. Then our sweep search utility was used to produce a list of all sweeps containing each 3D feature location of interest. The sweep search tool makes use of bounding boxes associated with each sweep and each directory in the sweep file hierarchy. Note that the bounding boxes were pre-computed and organized off-line by another of our utilities.

After identifying all sweeps containing a feature of interest, we applied our collection of curvature estimators and image processing options, selecting those which highlight the given feature of interest. Results for chisel marks, grafitti, and fracture lines are shown in Figure 9.2. Note that each of the features exhibit different characteristics. The chisel marks feature fills a small area and can be likened to parallel waves in water. The grafitti initials resemble long half-cylinder grooves in the surface. The fracture lines were the most challenging to visualize, because in fact the fracture width is as small as the scan point resolution, but still visible.

Because the 3D location of each feature was now known, it became possible to indicate feature locations on the triangle mesh model. In Figure 9.3, small red spheres are used to mark locations on the mesh model.

In addition to producing (2D) curvature maps for visualizing features of interest, we also employed our techniques for projecting surface texture onto the (3D) mesh model. Figure 9.4 shows single frame rendered images of the same features of interest. The video file (isvd.mov), on the CD-ROM which accompanies this thesis, includes animations of a scene which contains the mesh model and multiple texture...
Figure 9.1: Extracts from the curvature map image library.
9.3 Characterizing Chisel Marks

In this section we attempt to answer the question: *What was the size of the toothed chisel that Michelangelo used on the David statue over 500 years ago?*

A chisel, similar to those thought to have been in use during that time, is shown in Figure 9.5. Note that the approach we have used, which can be described as im-
Identifying and Visualizing Surface Detail

Figure 9.4: Michelangelo’s David with projected texture close-up: chisel marks, graffiti initials, thin fracture lines.

Figure 9.5: A typical three-toothed chisel used for carving stone.

The first steps in the approach presented in this section made use of our K-Scan software in an interactive mode. K-Scan screen shots of zoomed-in views of the chisel marks are shown in Figure 9.6. In the case of both images, surface points are gray-scale shaded for curvature visualization. A distance measurement reference plane, which was placed interactively (in 3D), can be seen in the second image.

Next, the distance from each point to the reference plane was calculated within K-Scan. The surface points are now gray-scale shaded to indicate distance, as shown in Figure 9.7. Points in front of the reference plane are shading coded increasing white the further they are away from the reference plane, and points behind the ref-
9.3. Characterizing Chisel Marks

A 2D image representation of the gray-scale shaded distances was exported from K-Scan. We now think of this 2D image, shown in Figure 9.8, as a (calibrated) depth map. Note that this depth map is in many ways very similar to a curvature map.

The depth map image was rotated, using 2D image processing, such that the chisel mark grooves aligned with the vertical direction. Then a subset of the image was extracted from the depth map and saved as the image shown in Figure 9.9.
image was also exported as a two-dimensional array of depth values.

The two-dimensional array was then averaged in the vertical direction, resulting in a one-dimensional array of smoothed depth values versus distance along the surface perpendicular to the chisel mark grooves.

Finally, a calibrated plot of the vertical average of pixel values, shown in Figure 9.10, reveals chisel information:
1) The spacing between chisel teeth is 3.5mm.
2) The chisel cut depth depth is only 0.20 - 0.24mm.
3) Evidence supports the use of a three toothed chisel.
9.3. Characterizing Chisel Marks

Figure 9.9: Chisel marks: extracted depth map subset.

Figure 9.10: Plot.
9.4 Summary and Conclusions

Analysis in this chapter included using a number of different curvature estimators along with an assortment of filtering techniques. 3D distance measurements associated with surface features were taken. Both curvature maps and depth maps were output for visualization of significant features. External processing of distance map files was used to produce visualizations and plots of depth measurements.

Chisel marks, graffiti initials, and thin fracture lines on Michelangelo’s David statue have been located and visualizations suitable for further analysis have been produced. We characterized a sample of the chisel marks left by Michelangelo.

We employed an existing simplified 3D triangle mesh model of the complete David statue and projected multiple curvature texture images onto this simplified model for full 3D visualization (See the video file isvd.mov, on the CD-ROM which accompanies this thesis).

The combination of differential geometry (for curvature) and computer graphics (for image processing and display) has been shown to be a powerful tool for the visualization and analysis of surfaces associated with scan point data.
Appendices
Appendix A

The Digital Michelangelo Project

In the 1998 - 1999 academic year, a 30 person team of faculty, staff, and students from the Computer Science Departments at the University of Stanford and the University of Washington traveled to Italy to digitize a number of Michelangelo’s sculptures (Levoy et al. 2000). The sculptures included most notably the David\textsuperscript{1}, along with Slave called Atlas, Awakening Slave, Bearded Slave, Youthful Slave, Dusk, Dawn, Day, Night, and St. Matthew. This was in many ways a technically ambitious undertaking. They needed to pack and ship their four tons of equipment, including a purpose built laser triangulation scanner, control electronics, and associated computers. The system used custom designed control software as well as a collection of custom designed data processing tools.

\textsuperscript{1}For historical details surrounding Michelangelo and his David statue, see (Scigliano 2005), (Gill 2002) and (Vasari 1972).

Figure A.1: Digitizing Michelangelo’s David (Levoy et al. 2000).
In addition to being the most well-known, the David statue, at over five meters tall, was the largest of the statues digitized. (Note the size of the statue relative to the two computer operators in Figure A.1.) For the David portion of the project, the total size of scanning team was twenty-two people plus museum staffing of, on average, three more people. The time spent in the Galleria dell’Accademia (Florence), scanning David, amounted to 360 hours over thirty days, totaling 1,080 man-hours.

The resultant data has been made available by Stanford, on application, to published researchers who are currently affiliated with a university, company, or other major institution, for non-teaching, non-commercial, non-redistributable use only.

A.1 The David Data-Set

A project overview and a data-set summary can be found on-line at a web site hosted by Stanford. However, for the purposes of this thesis, more detailed data-set information was deduced from the data itself.

The large data-set was expected by the Stanford team from the outset. With the desired scan resolution of 0.25mm, and the fact that the David statue is 5.17 meters tall, the linear resolution scale factor for this statue was 20,000 : 1. The David scan data, as provided by Stanford, comes in nine compressed downloadable archive files averaging over 200MB each in size. Extracting the archive contents yields a file hierarchy organized as, in the terminology used by Stanford, *days, scans,* and *sweeps.* The days are the actual calendar days on which scanning took place. Each scan (group) is associated with a particular data acquisition physical position. Finally, the acquired data can be found in individual sweep files within each scan directory.

To keep file size to a minimum, the acquired data points are stored within sweep files as 16-bit integer range values in a 2D array which matches the acquisition rasterization pattern. The sweep files include headers containing sufficient information to map these integer values back into 3D floating point space coordinates. The 2D array is run-length encoded, removing empty locations to further reduce file size.

Scan alignment files, containing a 4x4 translation and rotation matrix, are also included in each scan directory. These files are the result of a sequence of refined efforts by Stanford to align and merge all of the scans into unified real-world coor-

---

2http://graphics.stanford.edu/projects/mich/
The downloadable material also includes several complete triangle mesh models of David in various file formats, with varying quality and size. In the applications part of this thesis we made use of the highest quality simplified model. Interestingly and informatively, journal log entries, recorded on-site by scanner operators, can also be found in the download material.

Stanford produced a custom software tool, Scanalyze (Pulli et al. 2002), which they used to visualize 3D scan data models and generate scan alignment files. The C++ source code for Scanalyze is available for download from Stanford3. The author of this thesis extracted code from Scanalyze to convert raw sweep files into files containing 3D floating point values in aligned real-world coordinates.

A number of script-based analysis utilities where produced for this thesis. Initial exploration of the complete David data-set hierarchy revealed that it is organized as:

- 30 Days
- 515 Scans
- 6540 Sweeps

One of our utilities was used to identify bad sweeps. After removing bad sweeps, using such criteria as below minimum number of data points and out-of-range data, we were left with:

- 30 Days
- 514 Scans
- 6355 Sweeps

Another of our utilities was used to walk through the entire hierarchy, looking inside each sweep file. The total number of points in the data-set was determined to be 1,159,941,062. We also calculated an axis-aligned bounding box for the 3D data:

- MIN: [-963, -4427, -891]
- MAX: [216, 972, 1840]

3http://graphics.stanford.edu/software/scanalyze/
Because the data points were stored with real world (metric) coordinates, all subsequent mesh models could be used for realistic measurements. Three orthogonal projection reference renderings of the model were produced, using the right-handed xyz-axes convention, with positive y-axis up. The right side view is shown in Figure A.2. The coordinate axes are marked with colored arrows centered at the datum point \([0.0, 0.0, 0.0]\). The reference grid large squares are one meter per side and the small squares are a tenth of a meter per side. It can be clearly seen that the statue is,
A back view is shown in Figure A.3. In this view, the importance of the tree stump in supporting the statue at its center of gravity can be appreciated.
A view from the top is shown in Figure A.4. The constraints on David’s pose imposed by the reported relatively shallow depth of the original block of stone (Vasari 1972) can be appreciated in this view.

Finally, a perspective rendering of the full model, with lighting, by the author of this thesis, is shown in Figure A.5.
Figure A.5: Michelangelo’s David: rendered from the simplified model.
Appendix B

Publications

To date, the following refereed publications and conference presentations resulted have included material from this thesis:


Appendix C

The CD-ROM which accompanies this thesis contains the following material produced by the author:

- a high-resolution PDF file of the thesis,

- the video files psc.wmv and isvd.mov which complement Chapters 8 and 9 respectively,

- and a number of TIFF image files from the curvature map library, contained in a folder named David.
Bibliography


Levoy, M., Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D., Pereira, L., Ginzton, M.,
Michelangelo project: 3D scanning of large statues, Proc. SIGGRAPH, pp. 131–
144.


Lévy, B.: 2001, Constrained texture mapping for polygonal meshes, SIGGRAPH ’01: 
Proceedings of the 28th annual conference on Computer graphics and interactive tech-

Li, X. and Guskov, I.: 2005, Multi-scale features for approximate alignment of point-
based surfaces, SGP ’05: Proceedings of the third Eurographics symposium on Ge-
ometry processing, Eurographics Association, Aire-la-Ville, p. 217.

Mandal, C., Qin, H. and Vemuri, B.: 1997, Dynamic smooth subdivision surfaces for 

Meek, D. S. and Walton, D. J.: 2000, On surface normal and gaussian curvature ap-
Des. 17(6), 521–543.

Meyer, M., Desbrun, M., Schröder, P., and Barr, A. H.: 2003, Discrete differential-
egometry operators for triangulated 2-manifolds, Springer-Verlag, pp. 35–57.

data, SCG ’03: Proceedings of the nineteenth annual symposium on Computational 
geometry, pp. 322–328.

Mokhtarian, F. and Bober, M.: 2003, Curvature Scale Space Representation: Theory, 
Applications, and MPEG-7 Standardization, Kluwer, Dordrecht.

Montani, C. and Scopigno, R.: 1990, Rendering volumetric data using sticks repre-


of America, Washington.

Parent, P. and Zucker, S. W.: 1989, Trace inference, curvature consistency, and 
curve detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 
11(8), 823–839.


voting in 8d, The Proceedings of the Seventh IEEE International Conference on Com-

Tang, X. and Agam, G.: 2004, Curve-based local surface geometry estimation in
polyhedral surfaces, Vision Geometry XII, pp. 53–61.

Tomasi, C. and Manduchi, R.: 1998, Bilateral filtering for gray and color images,

by tensor voting, IEEE Transactions on Pattern Analysis and Machine Intelligence
27(3), 434–449.

Press, Boca Raton.

object matching, Pattern Recogn. Lett. 28(1), 49–57.


Veblen, O.: 1905, Theory of plane curves in non-metrical analysis situs, Trans. Amer-
ican Mathematical Society 6, 83–98.

in gorizia (italy) from laser scanning and photogrammetric 3d surveys, Web3D
’07: Proceedings of the twelfth international conference on 3D web technology, New
York, pp. 165–168.

Šrámek, M.: 1994, Fast surface rendering from raster data by voxel traversal using
chessboard distance, VIS ’94: Proceedings of the conference on Visualization ’94,

Xu, H. and Chen, B.: 2004, Stylized rendering of 3d scanned real world envi-
enments, NPAR ’04: Proceedings of the 3rd international symposium on Non-
photorealistic animation and rendering, ACM, New York, pp. 25–34.

Zhang, E., Hays, J. and Turk, G.: 2007, Interactive tensor field design and visual-
ization on surfaces, Visualization and Computer Graphics, IEEE Transactions on
13(1), 94–107.