Multi-run 3D Streetside Reconstruction from a Vehicle

Yi Zeng and Reinhard Klette

The *.enpeda..* Project, Department of Computer Science
The University of Auckland, New Zealand

Abstract. Accurate 3D modellers of real-world scenes are important tools for visualizing or understanding outside environments. The paper considers a camera-based 3D reconstruction system where stereo cameras are mounted on a mobile platform, recording images while moving through the scene. Due to the limited viewing angle of the cameras, resulting reconstructions often result in missing (e.g. while occluded) components of the scene. In this paper, we propose a stereo-based 3D reconstruction framework for merging multiple runs of reconstructions when driving in different directions through a real-world scene.

1 Introduction

Current large-scale camera-based reconstruction techniques can be subdivided into aerial reconstruction or ground-level reconstruction techniques. Although a large amount of user interaction is needed, the resulting model is often of high quality and visually compelling. There are various commercial products available in the market, demonstrating high quality, such as 3D RealityMaps [1], for example. However, reconstruction methods using aerial images only cannot produce models with photo-realistic details at ground level. There is an extensive literature on ground-level reconstruction; see, for example, [5, 9, 17]. In both aerial and ground-level reconstructions, cameras capture input images as they travel through the scene. Standard cameras only have limited viewing angles. Thus, a large number of blind spots of the scene exist, resulting in incomplete 3D models, and this is inevitable for a *single run reconstruction* (i.e. when moving cameras on a "nearly straight" path, without any significant variations in the path). A single run has a defined *direction*, being the vector from start and end point of the run.

In this paper, we propose a stereo-based reconstruction framework for automatically merging reconstruction results from multiple single runs in different directions. For each single run, we perform binocular stereo analysis on pairs of *left* and *right* images. We use the left image and the generated disparity map for a bundle-adjustment-based visual odometry algorithm. Then, applying the estimated changes in camera poses, a 3D point cloud of the scene is accumulated frame by frame. Finally, we triangulate the 3D point cloud using an α-shape algorithm to generate a surface model. Up to this stage we apply basically existing techniques. The novelty of this paper is mainly in the merging step, and we

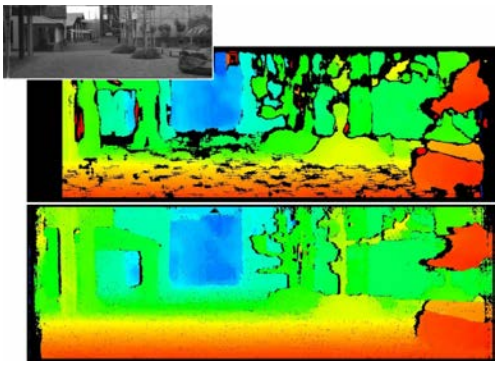

Fig. 1. From top to bottom: original image of a used stereo frame sequence, and colour-coded disparity maps using OpenCV (May 2013) block matching or iSGM.

detail the case where two surface models are merged generated from single runs in opposite directions. Input data are recorded stereo sequences from a mobile platform. In this paper we discuss greyscale sequences recorded at Tamaki campus, The University of Auckland, at a resolution of 960×320 at 25 Hz, with 10 bit per pixel. Each recorded sequence consists of about 1,800 stereo frames. For an example of an input image, see the top of Fig. 1.

The quality of the used stereo matcher has crucial impact on the accuracy of our 3D reconstruction. We decided for iterative semi-global matching (iSGM), see [8], mainly due to its performance at ECCV 2012 [7]. A comparison with the block-matching stereo procedure in OpenCV (see Fig. 1, middle) illustrates the achieved improvement by using iSGM.

The rest of the paper is structured as follows. In Section 2, we estimate the ego-motion of the vehicle using some kind of bundle adjustment. Section 3 discusses alpha-shape, as used for the surface reconstruction algorithm applied in the system. Finally, the merging step is discussed in Section 4, also showing experimental results. Section 5 concludes the paper.

2 Visual Odometry

Visual Odometry [13], the estimation of position and direction of the camera, is achieved by analysing consecutive images in the recorded sequence. The quality of our reconstructed 3D scene is directly related to the result of visual odometry. Drift in visual odometry [10] often leads to a twist in the 3D model. The basic algorithm is usually: (1) Detect feature points in the image. (2) Track the features across consecutive frames. (3) Calculate the camera's motion based on the tracked features. In this paper, since we focus on quality, an algorithm [15] based on *Bundle Adjustment* (BA) is used for visual odometry.

We tested a basic algorithm for comparison. 2-dimensional (2D) feature points are detected and tracked across the left sequence only. The *speeded-up robust feature detector* (SURF), see [2], is used to extract feature points in the first frame. We chose SURF over the Harris corner detector [6] (which is a common

choice in visual odometry) because corner points may not be evenly distributed depending on the geometry of the scene. The Lucas-Kanade [12] algorithm is used to track these detected features in the subsequent frame. Tracked feature points serve then as input, and are again tracked in the following frame, and so on. Since the same set of feature points is tracked, the total number of features decays over frames. When the total number of features drops below a threshold τ then a new set of features is detected using again the SURF detector. After calculating a relative transformation between Frames $t-1$ and t, the global pose of the cameras at time t is obtained by keeping a global accumulator, assuming that the pose of the camera at time 1 is a 4×4 identity matrix for initialization. However, in our experiments, when applying this basic algorithm, the estimation of camera pose transformations was inaccurate, and became less stable as errors accumulate along the sequence. In order to improve the accuracy, we apply a sliding-window bundle adjustment.

Bundle adjustment [16] is the problem of refining the 3D structure as well as the camera parameters. Mathematically, assume that n 3D points b_i are seen from m cameras with parameters a_j, and X_{ij} is the projection of the ith point on camera j. Bundle adjustment is the task to minimize the *reprojection error* with respect to 3D points b_i and cameras' parameters a_j. In formal representation, determine the minimum

$$\min_{a_j, b_i} \sum_{i=1}^{n} \sum_{j=1}^{m} d(Q(a_j, b_i), X_{ij})^2$$

where $Q(a_j, b_i)$ is the function projecting point i on camera j, and d is the Euclidean distance between points in the image plane. Bundle adjustment is a non-linear minimization problem which can be solved by using iterative methods such as Levenberg-Marquardt.

Ideally, the best result can be obtained by applying bundle adjustment to all the recorded frames. But, considering its complexity and the limited computing power we have, we use a sliding window bundle adjustment (similar to the method used in [15]), i.e. only optimizing the camera poses within a window of k frames, and moving this window across the whole sequence (only the left images are used for bundle adjustment).

Starting from frame F_1, a window of k images is constructed and the estimated camera poses are used as initial estimates. Then, bundle adjustment is applied for the window using the tracked features. In the next iteration, the window advances by one frame, i.e. we estimate now the camera pose for frame F_{k+1}, as described in the previous subsection. The estimated pose for F_{k+1} plus bundle-adjusted poses for F_2 to F_k, serve then as initial estimates for the camera pose for frame F_{k+2}, and so on.

3 Surface Reconstruction

In this section, we build a 3D model of the scene using results of visual odometry. The final surface representation is polygonal, but in order to build it we construct

a point cloud model first. Once we calculate the pose for cameras for all frames, building a 3D point cloud model can be as easy as projecting all 3D points derived from pixels with valid disparities into a global coordinate system. However, we did not accumulate pixels for all the frames, because the number of points grows exponentially, and a large percentage of points is actually redundant information. (The vehicle was driving at 10 km/h only, and recall that images were captured at 25 Hz.) For each frame, only pixels within a specified disparity range are used, due to the non-linear property of the Z-function. See Fig. 2 for an example.

Point-cloud data usually contain large portion of noise and outliers, and the density of points varies across the 3D space. Two additional steps are to be carried out to refine the quality of the point cloud.

Down-Sampling. A voxel grid filter is applied to simplify cloud data, thus improving the efficiency of subsequent processing. The filter creates a 3D voxel grid spanning over the cloud data. Then, for each voxel, all the points within are replaced by their centroid.

Outlier Removal. Errors in stereo matching and visual odometry lead to sparse outliers which corrupt the cloud data. Some of these errors can be eliminated by

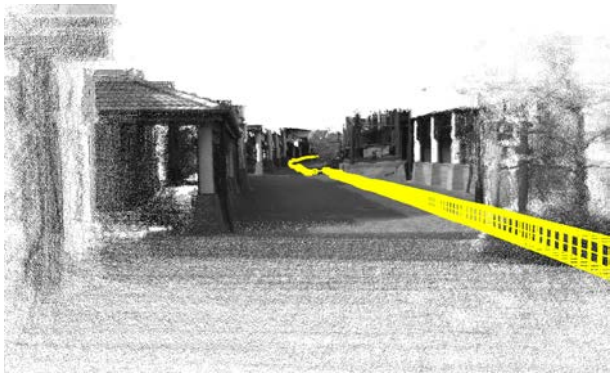

Fig. 2. A generated point cloud model. Yellow cubes indicate detected camera poses.

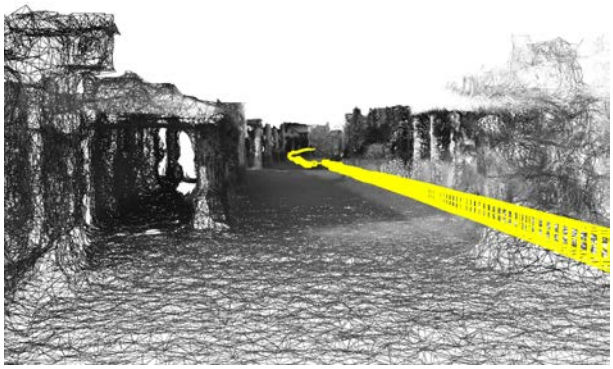

Fig. 3. A created surface model. Yellow cubes indicate camera poses.

applying a statistical filter on the point set, i.e. for each point, we compute the mean distance from it to all of its neighbours. If this mean distance of the point is outside a predefined interval, then the point can be treated as an outlier and is removed from the set. The order of these steps affects the overall performance of the process. The down-sampling process is significantly faster than outlier removal. Thus we decided to perform these two processes in the listed order.

Given a set S of points in 3D, the *α-shape* [4] was designed for answering questions such as "What is the shape formed by these points?" Edelsbrunner and Mücke mention in [4] an intuitive description of 3D α-shape: Imagine that a huge ice-cream fills space $\mathbb{R}^3$ and contains all points of S as "hard" chocolate pieces. Using a sphere-formed spoon, we carve out all possible parts of the ice-cream block without touching any of the chocolate pieces, even carving out holes inside the block. The object we end up with is the α-shape of S, and the value α is the squared radius of the carving spoon.

To formally define the α-shape, we first define an α-complex. An α-complex of a set S of points is a subcomplex of the 3D Delaunay triangulation of S, which is a tetrahedrization such that no point in S is inside the circumsphere of any of the created tetrahedra. Given a value of α, the α-complex contains all the simplexes in the Delaunay triangulation which have an empty circumscribing sphere with squared radius equal to, or smaller than α. The α-shape is the topological frontier of the α-complex.

In our reconstruction pipeline, after obtaining and refining a point-cloud model, the α-shape is calculated and defines a 3D surface model of the scene. See Fig. 3 for an example. Compared to Fig. 2, the reader might agree with our general observation that the surface model looks in general "better" than the point-cloud visualization.

4 Merging Models from Opposite Runs

Now we are ready to discuss our proposed merger of point-cloud or surface data obtained from multiple runs through a 3D scene.

The 3D model reconstructed from a single run (i.e. driving through the scene in one direction) contains a large number of "blind spots" (e.g. due to occlusions, e.g. the "other side of the wall", or the limited viewing angle of the cameras, but also due to missing depth data, if disparities were rated "invalid"). By combining the results from opposite runs, we aim at producing a more accurate and more complete model of the scene.

The task of aligning consistently models from different views is know as *registration*. Fully automatic pairwise registration methods exist for laser-scanner data, and the main steps are listed below

1. Identify a set of interest points (e.g. SIFT [11]) that best represent both 3D point sets.
2. Compute a feature descriptor at each interest point, using methods such as *fast point feature histograms* (FPFH); see [14].

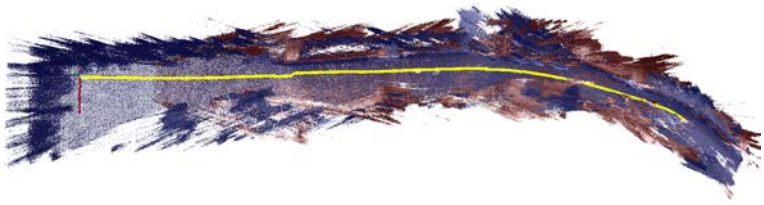

Fig. 4. Bird's-eye view of an initial alignment of two opposite runs. Results of each run are shown in different colours.

3. Estimate the correspondence between two sets of feature points based on their similarities. The simplest method is brute-force matching.
4. Assuming that the data is noisy, invalid correspondences are rejected to improve the registration.
5. Compute the pose transformation from the remaining correspondences.
6. Use the resulting estimation as an initial alignment; then apply an *iterative closest points technique* (ICP) to further align two point sets; see [3].

However, compared to laser-scanner data, stereo data is more inaccurate and contains a significant amount of noise, especially around the edge areas of scene objects. Therefore, the method stated above is not applicable for our system in this form. Considering the complexity of the scene (i.e. objects may look completely different from opposite directions) and the inaccuracy of stereo data, we propose the following semi-automatic method to align the two stereo point clouds.

Initial Alignment. We let the user manually select a set of corresponding points from both models. Then, a rough estimation of alignment is calculated by applying the least-square method. See Fig. 4 for an example.

Adjustment. Due to (not fully avoidable) errors in the visual odometry process and the considerable dimension (length) of the recorded scene, both point-cloud

Fig. 5. Bird's-eye views of individual and merged surface models.

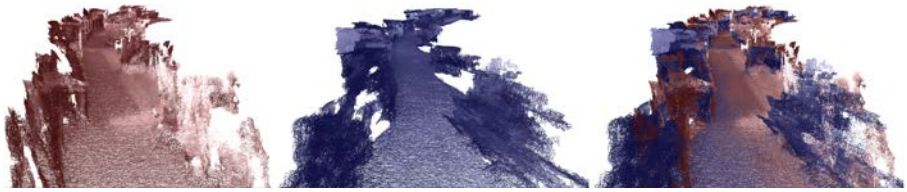

Fig. 6. Street views illustrating the benefit of merging 3D data.

models cannot be perfectly aligned as a whole. (Both models are twisted to a certain degree in 3D space.) Therefore, we break the point cloud models into a few segments along the Z direction (the main driving direction). Then we loop through each segment, apply feature matching across the two point-cloud models using 3D feature detectors, such as SIFT. A more precise alignment for this segment is calculated by matching the two feature sets. If the new alignment does not differ from the initial alignment more than a threshold τ, the new alignment is applied to the cloud segment.

Post Processing. Since we merged two (very extensive) point clouds, the point density is not uniform any more. We need to down-sample the merged point cloud again (as described in the previous section), for the convenience of subsequent processing. After the merged point cloud is simplified, a surface model can be created using the α-shape algorithm. See Fig. 5 for surface models of two separate runs, and for the merged point cloud.

The street views in Fig. 6 show clearly the benefit of merging: many of the missing parts in one run are filled-in by reconstruction results of the second run. The facades of buildings and other details of the scene are getting more complete, with an accuracy as defined by stereo matching and visual odometry. We will not further illustrate the obvious positive effects, but like to point on two detected issues when merging. Figure 7 reveals that occlusions walls from opposite directions intersect each other. Due to the inaccurate disparities around the edge area, a wall structure can be formed along the viewing direction on the edge. When merging models from opposite runs, the occlusion walls from the two models intersect each other.

Fig. 7. Occlusion walls from opposite directions intersect each other.

5 Conclusions and Future Work

In this paper we described a stereo-based 3D reconstruction pipeline for modelling street scenes. We proposed a semi-automatic method for aligning models reconstructed from opposite directions, to fill-in missing components. Our proposed system is certainly useful for improving the completeness of ground-level 3D reconstruction. It might also be useful for combining results of aerial and ground-level large-scale 3D reconstruction. For future improvements we see needs to increase the accuracy of the visual odometry process, and to enhance the quality of the point cloud model. Evaluation on the quality and performance of the reconstruction system also needs to be done.

Acknowledgment: The authors thank Simon Hermann for the provision of iSGM for stereo matching.

References

1. 3D Reality Maps: www.realitymaps.de/en/. Last visited in April (2013)
2. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded up robust features. In Proc. ECCV (2006) 404–417
3. Besl, P. J., McKay, N. D.: A method for registration of 3-D shapes. IEEE Trans. Pattern Analysis Machine Intelligence **14** (1992) 239–256
4. Edelsbrunner, H., Mücke, E. P.: Three-dimensional alpha shapes. ACM Trans. Graphics **13** (1994) 43–72
5. Geiger, A., Ziegler, J., Stiller, C.: StereoScan: Dense 3d reconstruction in real-time. In Proc. IEEE IV (2011) 963–968
6. Harris, C., Stephens, M. J.: A combined corner and edge detector. In Proc. Alvey Vision Conf. (1988) 147–151
7. Heidelberg *Robust Vision Challenge* at ECCV 2012. hci.iwr.uni-heidelberg.de/Static/challenge2012/ (2012)
8. Hermann, S., Klette, R.: Iterative semi-global matching for robust driver assistance systems. In Proc. ACCV (2012)
9. Huang, F., Klette, R.: City-scale modeling towards street navigation applications. J. Information Convergence Communication Engineering **10** (2012)
10. Jiang, R., Klette, R., Wang, S.: Statistical modeling of long-range drift in visual odometry. In Proc. ACCV workshop, LNCS 6469 (2011) 214–224
11. Lowe, D. G.: Distinctive image features from scale-invariant keypoints. Int. J. Computer Vision **60** (2004) 91–110
12. Lucas, B. D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In Proc. IJCAI **2** (1981) 674–679
13. Nister, D., Naroditsky, O., Bergen, J.: Visual odometry. In Proc. CVPR **1** (2004) 652–659
14. Rusu, R. B., Blodow, N., Beetz, M.: Fast point feature histograms (FPFH) for 3D registration. In Proc. IEEE ICRA (2009) 3212–3217
15. Sünderhauf, N., Konolige, K., Lacroix, S., Protzel, P.: Visual odometry using sparse bundle adjustment on an autonomous outdoor vehicle. In Proc. Autonome Mobile Systems (2005) 157–163
16. Triggs, B., McLauchlan, P., Hartley, R., Fitzgibbon, A.: Bundle adjustment - A modern synthesis. In Proc. Vision Algorithms Theory Practice (2000) 298–375
17. Xiao, J., Fang, T., Zhao, P., Lhuilier, M., Quan, L.: Image-based street-side city modeling. In Proc. SIGGRAPH (2009) 114:1–114:12