ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition

Qi Pan
http://mi.eng.cam.ac.uk/~qp202
Gerhard Reitmayr
http://mi.eng.cam.ac.uk/~gr281
Tom Drummond
http://mi.eng.cam.ac.uk/~twd20

Machine Intelligence Laboratories
Department of Engineering
Cambridge University
Cambridge
United Kingdom

Abstract

Off-line model reconstruction relies on an image collection phase and a slow reconstruction phase, requiring a long time to verify a model obtained from an image sequence is acceptable. We propose a new model acquisition system, called ProFORMA, which generates a 3D model on-line as the input sequence is being collected. As the user rotates the object in front of a stationary camera, a partial model is reconstructed and displayed to the user to assist view planning. The model is also used by the system to robustly track the pose of the object. Models are rapidly produced through a Delaunay tetrahedralisation of points obtained from on-line structure from motion estimation, followed by a probabilistic tetrahedron carving step to obtain a textured surface mesh of the object.

1 Introduction

3D models have an extensive range of uses in computer vision, but generating these models can be a difficult and time consuming task. Whilst many automated and semi-automated 3D reconstruction techniques exist, current methods have a variety of drawbacks meaning that ultimately many high quality 3D models are still generated using labour intensive methods or completely by hand. Off-line reconstruction methods [17, 22] have the potential to create

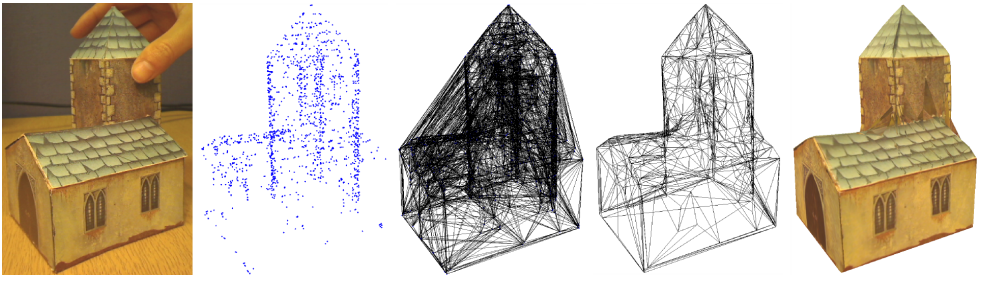

Figure 1: Results from each stage of the reconstruction pipeline. Left to right: (a) Object. (b) Point cloud. (c) Tetrahedralisation. (d) Surface mesh. (e) Textured model.

BMVC 2009 doi:10.5244/C.23.112

highly detailed and accurate 3D reconstructions, but this high level of detail usually implies that they are computationally expensive and cannot operate at frame rate. This means that the reconstruction pipeline must be split into two phases: an image sequence or video collection phase (fast), and an off-line processing phase (slow), after which the model is obtained. If at this stage the model is found to be flawed, additional images or an entire video sequence must be collected and the lengthy reconstruction phase repeated. It may take several iterations and many hours before obtaining an acceptable model. We implement a system which reconstructs an object on-line, allowing the user to ascertain the quality of the input sequence during collection and to obtain new views based on the current state of the model.

Object reconstruction systems need to segment the object to be modelled from the background. The simplest method is to use a black background, although this limits the modelling environment. Another option is colour segmentation, although this requires careful parameter tuning. Colour segmentation combined with a volumetric graph cut [3] can yield a segmentation, but is very computationally expensive. A prior model [15] or user input [18] can also be used to aid segmentation, but these methods are usually too time consuming or labour intensive for on-line systems. Usually, segmentation is difficult because the object remains stationary relative to the background while the camera is moved around to take images. Therefore to make segmentation easier, we opt to keep the camera stationary and to move the object around instead, allowing segmentation of the object from the background through use of geometric constraints based on rigid body motion.

A few real-time and on-line reconstruction systems exist. Rusinkiewicz *et al*. [25] implement a system which uses a projector to project structured light onto a white untextured object held using a black glove against a black background. The use of a black background, black glove and projector are undesirable, as well as the system's inability to model textured objects. Pollefeys *et al*. [23] present a plane-sweeping system capable of producing 3D models of urban environments in real-time, although an array of 4 cameras is required as well as GPS and INS measurements for pose estimation. Several monocular SLAM [6, 7] based systems [4, 10] exist which fit planes to sparse point clouds maps in real-time, although only sparsely and thus do not create fully dense models. Paalanen *et al*. [20] describe an on-line SLAM-based system for producing a coloured semi-dense point cloud of a scene, but (purposefully) avoid the issue of simplifying the point cloud into a geometric model. Brown *et al*. [2] implement a method of producing edge models of simple edge-based objects through a series of mouse interactions from a live video feed. VideoTrace [28] is capable of producing 3D models from videos through user interactions with a mouse. However, the video must be preprocessed using structure and motion analysis [22] and interactions must be performed on still frames, constraining the system to off-line operation.

To our knowledge, apart from ProFORMA, no completely automatic system exists which reconstructs a textured geometric model from an object on-line using only a camera and commodity hardware.

1.1 Contribution

Our system, ProFORMA, enables textured 3D models to be acquired on-line in just over a minute, with models being reconstructed fast enough to provide feedback for view planning. The user is free to interact with the object which is robustly tracked using a method that makes immediate use of the partial model. The reconstruction environment is not constrained to a studio and enables models of textured objects to be acquired using a single camera and commodity hardware. Rapid reconstruction is enabled by using a novel recur-

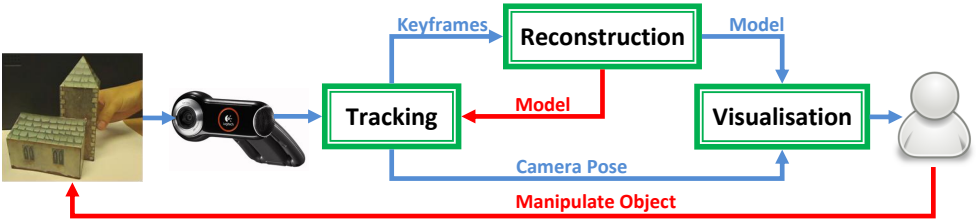

Figure 2: Flowchart of system operation.

sive probabilistic tetrahedron carving algorithm, which uses visibility of observed features to rapidly obtain a surface model of an object. Parts of an object not yet seen by the system are indicated to the user who can then manipulate the object in order to provide new views.

2 System Overview

We implement an on-line reconstruction system with a fixed-position video camera. The user can move a textured object in front of the camera using their hand. This enables the object to be orientated so that all parts of the object (including the base) can be viewed and modelled. No assumptions about the object are made and no prior information is known about the object, although the object must be sufficiently textured. A partial model is built as the user moves the object around, providing immediate feedback to the user about the state of the reconstruction.

When tracking the object to be modelled, it is very advantageous to use a video sequence rather than an image sequence as this takes advantage of small inter-frame motions (at high frame rates). In terms of reconstruction, however, temporally close and thus spatially close video frames provide very little 3D information, whilst adding to the reconstruction cost (dominated by a global bundle adjustment which grows $O(n^3)$ with the number of frames used [8]). Therefore we opt for a two-threaded keyframe based approach, with separate tracking and reconstruction threads, as used by Klein and Murray [13].

Figure 2 shows the design of the system. The *tracking* thread takes a video input of the scene and calculates the pose of the camera relative to the object at frame rate. It also tracks the location of 2D image features which adhere to rigid body motion constraints (epipolar constraints). When a large enough rotation is detected, a keyframe is taken and passed to the *reconstruction* thread, which generates a partial 3D model of the object. This model is fed back into the tracker to provide additional structural information that can be used during tracking. The model is also utilised by the *visualisation* system to show the user the state of the reconstruction so far, with the model pose updated to the live pose of the object obtained from the tracking thread. The user provides feedback to the system by looking at the visualisation and manipulating the object accordingly to provide new views.

3 Tracking

The tracking thread tracks the 6 DOF pose of the calibrated camera and establishes new feature tracks for building unmodelled parts of the object. It requires a partial model (mesh of triangles) from the reconstruction thread, as well as a list of landmarks. Pose tracking is performed using a similar method to Rosten and Drummond [24] but by using a fused pair of point trackers instead of a fused point and line tracker. New feature tracks on currently unmodelled parts of the object are obtained from a 2D tracker using epipolar constraints.

3.1 Robust Point Tracker

The robust point tracker follows transient features with 3D location from frame to frame. It operates on frames H_n half-sampled from video frames I_n. For non-maximally suppressed FAST features $F_{NM}(H_{n-1})$ extracted from the last tracked frame, 3D locations are calculated by intersecting rays cast from the last camera position with the mesh model. Feature descriptors are 16 pixel rings of intensities. These are matched against nonmax features $F_{NM}(H_n)$ extracted from the current frame, by SSD matching. The resulting 2D-3D correspondences are used to estimate an updated camera pose using the PROSAC robust estimation technique [5] with the three point pose algorithm [9].

3.2 Drift-free Point Tracker

The first point tracker is robust to large motions, but drifts with time as errors accumulate frame to frame. Therefore, the pose obtained from the first tracker is used as the starting point for the second point tracker, which is not robust to large motions but does not suffer from much drift. This tracker works by reprojecting visible landmarks from the model into the current image, I_n. Visibilty is determined by intersecting rays from camera to landmark with triangles in the model. Each landmark L_k is matched to FAST features $F(I_n)$ extracted from I_n which are within a 10 pixel radius of its reprojected position by lowest SSD. Descriptors are 5x5 mean and variance normalised image intensity patches, rotation normalised by aligning the patch in the direction of strongest blurred gradient around the feature. The descriptor for L_k is chosen to be the landmark's appearance in the keyframe with closest camera rotation to the current camera pose, $K_C(L_k)$. Married pair matching is performed such that the feature $F(I_n)_i$ matched to L_k is matched in the reverse direction to features in $K_C(L_k)$ within a 10 pixel radius of the landmark location in $K_C(L_k)$. A feature is successfully matched to a landmark only if the descriptors mutually have the lowest SSD when matching in both directions, and are below a threshold value. Due to the model being incomplete, visibility information may be incorrect (*e.g.* initially in concavities), so landmarks in the closest 3 keyframes are also reprojected into the current image.

Once the matches are obtained, a robust M-estimator with Tukey influence function [27] is used to minimise reprojection error of landmarks by optimising camera pose. Camera pose is parameterised as a member of the Lie group, $SE(3)$ [29], the group of rigid body transformations in $\mathbb{R}^3$. The derivative of reprojection error with respect to camera pose, the Jacobian, is calculated by considering the tangent space of $SE(3)$, the 6 dimensional Lie algebra $\mathfrak{se}(3)$. Using the pose estimate from the first tracking stage as a starting point and iterating allows an accurate and robust pose estimate to be obtained. Combining the two stages of tracking together in the manner described enables the system to track the object over large motions without succumbing to the effects of drift.

3.3 2D Tracker

As well as tracking the pose of the camera and position of 3D landmarks, the tracking thread is also required to detect and track new features on the object for reconstruction. This is achieved by tracking features which adhere to the epipolar geometry between the current camera pose and the pose of the last keyframe. The essential matrix, E, is calculated from the relative poses of the cameras [11].

Non-maximally suppressed FAST features from the last keyframe, $F_{NM}(K_j)$, which do not correspond to an existing landmark, are matched to FAST features in the current frame, $F(I_n)$. Matching is constrained to within a 10 pixel radius of the last seen location and is performed in the same manner as in Section 3.2, with descriptors again being 5x5 rotation, mean and variance normalised patches. If a feature $F(I_n)_i$ is a match to $F_{NM}(K_j)_k$, and $F(I_n)_i$ lies within a threshold distance from the epipolar line cast by $F_{NM}(K_j)_k$, the match is accepted as an epipolar match and the last seen location of $F_{NM}(K_j)_k$ is updated to the location of $F(I_n)_i$. The descriptor of $F_{NM}(K_j)_k$ is not updated each frame order to eliminate drift. This means that changes in appearance of a feature due to pose changes are unaccounted for, but as keyframes are taken regularly, this does not have much of an effect on tracking. If a feature $F_{NM}(K_j)_k$ has not been an epipolar match in any of the last 5 frames, it is regarded as not adhering to the rigid body motion of the object (or has been occluded) and is no longer tracked by the 2D tracker.

3.4 Keyframe Addition

When the pose has changed by a small rotation ($\sim 10^\circ$) since the last keyframe, the current video frame is taken as a new keyframe. Existing landmarks visible in the keyframe are augmented with a new location and appearance (descriptor). Feature tracks obtained from the 2D feature tracker are added as new landmarks, with an estimated 3D position provided by triangulating the rays from cameras to feature coordinates. The tracking thread then creates a new reconstruction thread to reconstruct the model in the background.

3.5 Tracker initialisation

When the system first starts, no structural information about the object to be modelled is known. A fixation constraint is imposed; it is assumed that at least part of the object is within the central area of the image when reconstruction starts. A keyframe is taken when the system starts. The 2D tracker (Section 3.3) is used as the sole tracker as no partial 3D model or landmarks are available at this stage. PROSAC is used in conjunction with essential matrix estimation from 5 points [19] to find the essential matrix between the current video frame and the previous keyframe. Hypotheses for the five point pose algorithm are drawn from matches near the centre of the image due to the fixation constraint. Once the essential matrix is obtained, the 2D tracker works in the same way as in Section 3.3. The essential matrix is decomposed [12] to find the camera pose (rotation and translation up to scale). Translation scales of transformations between initial keyframes are normalised by setting the translation scale between the first two keyframes to 1 and ensuring that landmarks visible in multiple keyframes have (approximately) the same 3D location.

4 Reconstruction

The reconstruction thread creates a rendered 3D model from a list of landmarks, keyframes and keyframe camera poses received from the tracking thread. Results from each stage of reconstruction can be seen in Figure 1. First, a point cloud is created by refining camera and landmark positions using bundle adjustment. This is is then converted into a mesh through a Delaunay tetrahedralisation. Tetrahedra are carved away based on visibility to obtain a surface mesh, which is then textured mapped to create the 3D model.

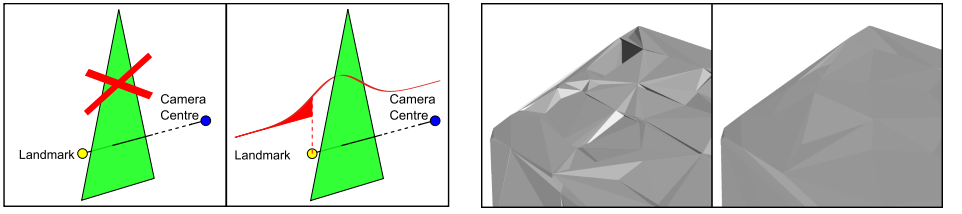

Figure 3: Left to right: (a) Simple carving. (b) Probabilistic carving. (c) Church roof after simple carving. (d) Church roof after probabilistic carving.

4.1 Bundle Adjustment

Bundle adjustment is the minimisation of reprojection error by simultaneously optimising over all 3D point locations and camera poses. Global bundle adjustment is usually too expensive to perform for on-line reconstruction, but using a two threaded approach means that tracking can continue whilst bundle adjustment runs in the background. As we are modelling a single object, the number of keyframes remains relatively small. We use an efficient implementation of bundle adjustment which performs a sparse factorisation of the bundle Hessian [26] and is able to bundle adjust 7776 observations of 2127 landmarks over 30 keyframes in 1.07 seconds (single core of a 2.4GHz Intel dual core CPU). This means that results from this stage, a 3D point cloud and list of keyframe camera positions, become available in a timely manner for both the next stage of reconstruction and hence for tracking. A point cloud produced by the system can be seen in Figure 1(b). Landmarks which have a reprojection error of more than 1 pixel are rejected at this stage.

4.2 Surface Recovery

A point cloud in itself is not a good enough representation of an object, so processing is required to retrieve the model surface. There are many methods of achieving this, of which two effective ones are volumetric graph cuts [14] and space carving [16]. These methods are, however, too costly for on-line reconstruction, taking tens of minutes or even hours to complete. Space carving also requires detection of the object's silhouette, which in this case is corrupted the the user's hand. Therefore, a different approach is taken which allows the mesh of the object to be recovered using an efficient algorithm that is also robust to partial occlusion from the user's hand. A variation to basic space carving is used; the space is first partitioned into non-regular voxels, which are then carved based on visibility of landmarks. Labatut *et al*. [17] also use a Delauanay tetrahedralisation and landmark visibility information, but as part an off-line reconstruction process involving a computationally expensive graph cut.

4.2.1 Delaunay Tetrahedralisation

Space partitioning is performed through a Delaunay tetrahedralisation using CGAL [1]. A Delaunay tetrahedralisation is the extension of a 2D Delaunay triangulation to 3D. The convex hull of the point cloud is partitioned into tetrahedra with the condition that the circumsphere of each tetrahedron contains no vertices of any other tetrahedra. Figure 1(c) shows the Delaunay tetrahedralisation of the Church point cloud. The model at this stage is a partitioned convex hull, so tetrahedra must be carved away to obtain the true surface.

4.2.2 Carving

One fast and efficient method of removing tetrahedra is by looking at visibility of landmarks. Each triangular face of a tetrahedron is tested in turn against all rays from landmarks to cameras in which they were visible as shown in Figure 3(a). Tetrahedra with one or more faces which intersect with rays are removed. Let T_i represent a triangle in the model, j the keyframe number, k the landmark number and $R_{j,k}$ the ray from the camera centre at keyframe j to landmark k. Let υ represent the set of all rays with indice pairs (j,k) for which landmark k is visible in keyframe j. For each triangle in the model, the probability that it exists given the set of visible rays can be expressed as:

$$P_{exist}(T_i|\upsilon) = \prod_{\upsilon} P_{exist}(T_i|R_{j,k}) = \prod_{\upsilon} \left(1 - Intersect(T_i, R_{j,k})\right) \tag{1}$$

$$Intersect(T_i, R_{j,k}) = \begin{cases} 1 & \text{if } R_{j,k} \text{ intersects } T_i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

In this formulation, $P_{exist}(T_i|\upsilon)$ takes the value of 0 if any rays intersect the triangle and 1 if no rays intersect the triangle. This yields a very noisy model surface as points slightly below a true surface due to noise cause the true surface to be carved away (Figure 3(c)). Therefore, we design a probabilistic carving algorithm which takes surface noise into account, yielding a far smoother model surface (Figure 3(b) and (d)). Landmarks are taken as observations of a surface triangle corrupted by Gaussian noise along the ray $R_{j,k}$, centered at the surface of triangle T_i with variance σ^2. Let $x = 0$ be defined at the intersection of $R_{j,k}$ and T_i, and let x be the signed distance along $R_{j,k}$, positive towards the camera. Let l_k be the signed distance from $x = 0$ to landmark L_k. The null hypothesis is that T_i is a real surface in the model and thus observations exhibit Gaussian noise around this surface. The hypothesis is tested by considering the probability of generating an observation at least as extreme as l_k:

$$P(L_k|R_{j,k}, T_i) = \int_{-\infty}^{l_k} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-x^2}{2\sigma^2}\right) dx \tag{3}$$

This leads to a probabilistic reformulation of simple carving:

$$P_{exist}(T_i|\upsilon) = \prod_{\upsilon} P_{exist}(T_i|R_{j,k}) \tag{4}$$

$$P_{exist}(T_i|R_{j,k}) = \begin{cases} P(L_k|R_{j,k}, T_i) & \text{if } R_{j,k} \text{ intersects } T_i \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

If $P_{exist}(T_i|\upsilon) > 0.1$, the null hypothesis that T_i exists is accepted, otherwise it is rejected and the tetrahedron containing T_i is marked for removal.

It is unnecessary to use all visible rays in the probability expression for each triangle, as rays which do not intersect the triangle do not impact the score whilst adding to the computational cost. To obtain a subset of rays which might intersect the triangle, range trees are formed for each keyframe's visible landmarks (in 2D screen coordinates). Each triangle of each tetrahedron is projected onto a keyframe. The keyframe's range tree is then used to find all features (and corresponding rays) which lie within the bounding box of the projected triangle, and only these rays are used in the probability expression (as the others are guaranteed not to intersect the triangle). It can be further noted that each tetrahedron shares faces with other tetrahedra, and that adjacent faces only need to be tested once due to having the same probability score.

If a tetrahedron with four neighbours is carved away whilst none of its neighbours are, it makes no difference to the surface model as the hole left by the tetrahedron is occluded by its neighbours. Similarly, tetrahedra on the surface of an object can be seen as a barrier, shielding tetrahedra beneath it from influencing the surface model. Starting with the partitioned convex hull, we can postulate that its surface tetrahedra form the carved surface of the model. If this is true, trying to carve tetrahedra on the surface will not remove any tetrahedra, but if this is false, tetrahedra will be carved away and the surface altered. If the process is repeated for the new surface until convergence (i.e. no more tetrahedra are removed), the surface is now visually identical to the carved surface obtained by testing all rays against all tetrahedra. This is due to the effects of now removing any non-surface tetrahedron being occluded by the surface tetrahedra. In its current form, the process is slow due to needing to compute which tetrahedra lie on the surface after each iteration. It can be noted, however, that removing a single surface tetrahedron simply adds its neighbours to the set of surface tetrahedra whilst removing itself. Therefore, a very efficient recursive carving algorithm can be designed. A carving function takes a (surface) tetrahedron as input, which is tested for removal. If is it is removed, then the function calls itself on the neighbours of the tetrahedron. If it is not removed, then it is known to lie on the true carved surface and the function terminates. If the function is called on all surface tetrahedra of the convex hull, then the true carved surface can be obtained. Testing is done efficiently by ensuring that no tetrahedron is ever tested twice and that only one score is calculated per pair of adjacent faces. The recursive algorithm yields significant speed-ups over the naïve approach of testing all tetrahedra due to only needing to test a small proportion of tetrahedra in the partitioned convex hull. Table 1 in Section 6 presents an evaluation of the performance improvement thus obtained.

5 Visual Feedback and User Interaction

After carving the tetrahedra based on visibility, texture is rendered onto all triangles on the surface mesh. Texture for each triangle is chosen from the keyframe with the most vertices of the triangle visible. If this is tied between several keyframes, the keyframe with the most normal view of the surface is chosen. The rendered model is displayed to the user and orientated to the live pose obtained from the tracker, although the view can also be manipulated using the mouse. Surfaces not yet visible from any keyframe (surface normal points away from all cameras) are coloured in black and yellow stripes to indicate this fact to the user (Figure 4).

6 Results and Discussion

The reconstruction system was implemented on a machine with an Intel 2.4GHz dual core CPU and Logitech Quickcam Pro 9000 (640×480 @ 15fps). Models are displayed using Geomview [21]. Videos showing real-time footage of the system during reconstruction can be found at http://mi.eng.cam.ac.uk/~qp202.

Figure 4 show models in various poses compared to a real image of the object. Table 1 shows that models are produced very quickly using ProFORMA, taking under 3s to generate a complete model after the last keyframe is added. Using ProFORMA to acquire complete textured models of the box and church objects took 61s and 75s respectively, including video collection time. Measurements on the Box model showed that when planes were fitted (by

Figure 4: Objects and reconstructed models. Clockwise from top left: (a) Box. (b) Lilt Can. (c) Church. (d) Jewellery Box.

least squares) to the surface model, angles between pairs of faces deviated by less than 1° compared to measurements from the object. When calculating the distances between opposing fitted planes from the midpoint of faces, it was found that the ratio of distances was 1:1.03:1.72. Measurements obtained from the real object were 52x53x90mm, corresponding to a ratio of 1:1.02:1.73, i.e. a deviation of less than 1mm at object scale. Table 1 shows that the recursive carving algorithm yielded a significant speed-up compared to the naïve algorithm of testing all tetrahedra, taking $\frac{1}{7}$th of the time for the convex Box and $\frac{2}{7}$th for the Church, which contained significant concavities. These results show that even though models are generated rapidly by ProFORMA, they are still of a high quality and bear great resemblance to the real objects.

Table 1: Statistics for Box and Church models using different carving algorithms.

	Box (26 keyframes, 61s)		Church (34 keyframes, 75s)	
Carving algorithm	Test All	Recursive	Test All	Recursive
Total Tetrahedra	11298	11298	8728	8728
Tetrahedra in model	10796	10796	7834	7834
Tetrahedra tested	100%	7.8%	100%	18.9%
Carving Time/s	5.01	0.71	4.33	1.25
Total Reconstruction Time/s	6.93	2.63	5.95	2.87

7 Conclusion and Future Work

The tight feedback loop of model reconstruction and model-based tracking provides a robust and accurate estimate of camera pose for the system. Coupled with the rapid recursive probabilistic carving algorithm, ProFORMA can be used to acquire fully textured geometric models in just over a minute. The partial models provide excellent guidance for the user in planning further views, as well as making tracking robust to large motions and occlusion.

There are many avenues of further investigation for ProFORMA. Currently, a few incorrect tetrahedra may remain if there is little texture in a concavity (*e.g.* RHS of Church in Figure 4), but integrating edges into tracking and texture into carving may enable these concavities to be better modelled. It would be interesting to process image sequences obtained from ProFORMA using an off-line reconstruction algorithm to obtain higher quality models, and to validate that good models from ProFORMA correlate to good models from off-line reconstruction. Another area of interest would be user guided modelling, where the system guides the user to novel views or revisits views of possibly erroneous parts of the model.

References

[1] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[2] M. Brown, T. Drummond, and R. Cipolla. 3D model acquisition by tracking 2D wireframes. In *Proc. British Machine Vision Conf.*, 2000.

[3] N.D.F. Campbell, G. Vogiatzis, C. Hernandez, and R. Cipolla. Automatic 3D object segmentation in multiple views using volumetric graph-cuts. *Image and Vision Computing*, 2008.

[4] D. Chekhlov, A.P. Gee, A. Calway, and W. Mayol-Cuevas. Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual slam. In *Proc. Int'l Symposium on Mixed and Augmented Reality*, 2007.

[5] O. Chum and J. Matas. Matching with PROSAC-progressive sample consensus. In *Proc. Computer Vision and Pattern Recognition*, volume 1, 2005.

[6] AJ Davison, ID Reid, ND Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[7] Ethan Eade and Tom Drummond. Scalable monocular slam. In *Proc. Computer Vision and Pattern Recognition*, 2006.

[8] C. Engels, H. Stewenius, and D. Nister. Bundle adjustment rules. *Photogrammetric Computer Vision*, 2006.

[9] M. Fischler. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[10] A.P. Gee, D. Chekhlov, A. Calway, and W. Mayol-Cuevas. Discovering higher level structure in visual slam. *IEEE Trans. Robotics*, 24(5):980–990, Oct. 2008.

[11] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, 2004.

[12] B.K.P. Horn. Recovering baseline and orientation from essential matrix. *J. Optical Society of America*, 1990.

[13] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Int'l Symposium on Mixed and Augmented Reality*, 2007.

[14] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. *Lecture Notes in Computer Science*, pages 82–96, 2002.

[15] M. Pawan Kumar, Philip H. S. Torr, and A. Zisserman. Obj cut. In *Proc. Computer Vision and Pattern Recognition*, 2005.

[16] K.N. Kutulakos and S.M. Seitz. A theory of shape by space carving. *Int. J. Comput. Vision*, 38(3):199–218, 2000.

[17] P. Labatut, J.P. Pons, and R. Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *Proc. Int'l Conf. on Computer Vision*, 2007.

[18] V. Lepetit and M.O. Berger. A semi-automatic method for resolving occlusion in augmentedreality. In *Proc. Computer Vision and Pattern Recognition*, 2000.

[19] D. Nister. An efficient solution to the five-point relative pose problem. *Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.

[20] P. Paalanen, V. Kyrki, and J. Kamarainen. Towards monocular on-line 3d reconstruction. In *ECCV Workshop on Vision in Action*, 2008.

[21] M. Phillips, S. Levy, and T. Munzner. Geomview: An interactive geometry viewer. *Notices of the American Mathematical Society*, 40(8):985–988, 1993.

[22] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *Int. J. Comput. Vision*, 59(3): 207–232, 2004.

[23] M. Pollefeys, D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. Detailed real-time urban 3d reconstruction from video. *Int. J. Comput. Vision*, 78(2-3):143–167, 2008.

[24] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *Proc. Int'l Conf. on Computer Vision*, 2005.

[25] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3d model acquisition. *ACM Transactions on Graphics (TOG)*, 21(3):438–446, 2002.

[26] B. Triggs, P.F. McLauchlan, R.I. Hartley, and A.W. Fitzgibbon. Bundle adjustment-a modern synthesis. *Lecture Notes in Computer Science*, pages 298–372, 1999.

[27] J.W. Tukey. Exploratory data analysis. *Behavioral Science: Quant. Methods*, 1977.

[28] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P.H.S. Torr. VideoTrace: rapid interactive scene modelling from video. In *Proc. SIGGRAPH*, 2007.

[29] VS Varadarajan and DP Želobenko. Lie groups, Lie algebras, and their representations. *Bull. Amer. Math. Soc. 81 (1975), 865-872. PII: S*, 2(9904):13867–5, 1975.