Rapid Scene Reconstruction on Mobile Phones from Panoramic Images

Qi Pan [*] Cambridge University
Clemens Arth [†] TU Graz
Gerhard Reitmayr [‡] TU Graz
Edward Rosten [§] Cambridge University
Tom Drummond [¶] Cambridge University

Abstract

Rapid 3D reconstruction of environments has become an active research topic due to the importance of 3D models in a huge number of applications, be it in Augmented Reality (AR), architecture or other commercial areas. In this paper we present a novel system that allows for the generation of a coarse 3D model of the environment within several seconds on mobile smartphones. By using a very fast and flexible algorithm a set of panoramic images is captured to form the basis of wide field-of-view images required for reliable and robust reconstruction. A cheap on-line space carving approach based on Delaunay triangulation is employed to obtain dense, polygonal, textured representations. The use of an intuitive method to capture these images, as well as the efficiency of the reconstruction approach allows for an application on recent mobile phone hardware, giving visually pleasing results almost instantly.

Index Terms: I.2.10 [Artificial Intelligence]: Vision and Scene Understanding—3D/stereo scene analysis I.4.8 [Image Processing And Computer Vision]: Scene Analysis—Tracking; I.5.4 [Pattern Recognition]: Applications—Computer Vision C.5.3 [Computer System Implementation]: Microcomputers—Portable devices (e.g., laptops, personal digital assistants)

1 Introduction

The generation of 3D models of scenes and environments has attracted a lot of attention in recent years. The tremendous importance of such models for all sorts of applications is extremely evident. These models can be used to illustrate architectural changes in urban environments. New business paths for commerce, advertising and social media can be introduced, needless to consider the value of these models for navigation tasks and AR.

In the Computer Vision (CV) community, numerous approaches have recently been presented concerning computational demands and the scale at which the reconstruction problem is solved. Common to all approaches is the aim of shifting the principle generation method from manual modelling using, for example, CAD software, to a fully automated approach for reconstruction. Many methods use a very large volume of images whilst exploiting the power of cloud computing. Some methods use, in addition to images, information from non-visual sensors like laser-range finders, GPS receivers and compasses to solve the task. Only a very small group of approaches allows for reconstruction in an on-line and instant manner, mostly using very powerful and costly hardware. However, despite the total number of all these approaches, methods of reconstruction working on current mobile hardware, especially mobile phones, have been little explored.

In traditional usage scenarios, the focus lies on the accuracy of the model, not on the time taken to reconstruct it. Improvements in mobile phone technology have allowed a device capable of delivering AR to be available in the pocket of the average consumer. In turn, to be able to deliver AR on this ubiquitous platform, requires 3D models of the augmented scenes. As detailed 3D models for the whole world do not exist (and probably cannot exist due to environments and lighting conditions changing), the need to rapidly create coarse models of scenes in-situ, to track from and to annotate, is apparent. In this respect, we consider the possibility of solving this problem as a fundamental foundation for future AR applications on mobile phones. This group of target platforms enforces the development of a solution working under the conditions of reduced computational and memory resources. Furthermore, the applicability of such an approach is strongly dependent on the possibility of delivering reasonable results in a shortened time frame. To a great extent, these restrictions render currently known approaches inapplicable on these devices, giving room for research in the direction of specially designed, efficient on-line algorithms to tackle all the well-known limitations of embedded hardware.

The contribution of this work is the presentation of a novel approach to generate visually appealing, textured 3D models from a set of at least 3 panoramic images on mobile phones without the need for remote processing. Building upon previous work from Pan *et al.* [21], the approach is capable of performing on current smartphone hardware within a time span of several seconds. An intuitive method is used for capturing panoramic images with commonly available small field of view (FOV) cameras which are now present in almost all smartphones. Considering geometrical relationships between individual views, feature correspondences are established robustly and efficiently. These correspondences form the basis for the reconstruction of a scene using a modified space carving approach. The approach is unique in terms of efficiency and, thus, is applied on modern smartphones to demonstrate its suitability for use in an AR scenario. The resulting 3D models are fully textured polygonal models that can be further used for annotation and tracking.

In Section 2 we review related work in the area of reconstruc-

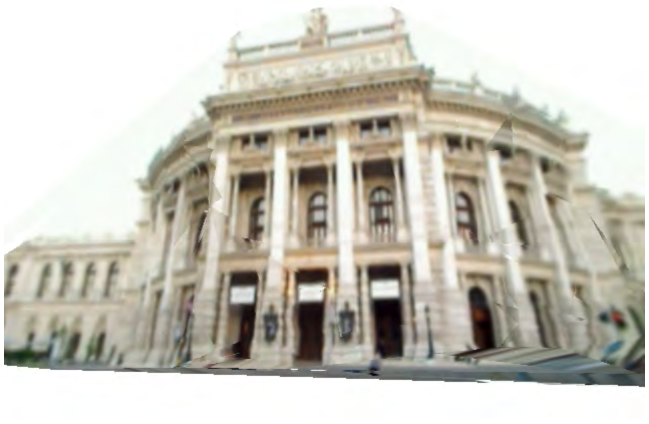
Figure 1: 3D model of a building reconstructed from a set of panoramic images on a mobile phone.

[*]e-mail: qp202@cam.ac.uk
[†]e-mail:arth@icg.tugraz.at
[‡]e-mail:reitmayr@icg.tugraz.at
[§]e-mail:er258@cam.ac.uk
[¶]e-mail:Tom.Drummond@monash.edu

IEEE International Symposium on Mixed and Augmented Reality 2011
Science and Technolgy Proceedings
26 -29 October, Basel, Switzerland
978-1-4577-2184-7/10/$26.00 ©2011 IEEE

tion and modelling. In Section 3 an overview about the entire approach is given, discussing the entire workflow in detail and providing notes on design decisions. Section 4 covers the main part of this work, describing the algorithms used to create 3D models from image sets exhaustively. The results of experimental tests using mobile hardware are given in Section 5. The paper concludes with a summary of the presented work and an outlook on future improvements in Section 6.

2 Related Work

Many different approaches to automatic 3D model reconstruction have been proposed in the last decade, mainly coming from the CV community. The majority of these techniques perform the task in an off-line manner. Solving the task on-line using live video feeds has become possible only recently, mainly due to the use of very powerful hardware. Approaches that can realistically be used on mobile phones have been proposed in the past, but they are only very few in number and limited in scope. Our review of related work is organized accordingly.

2.1 Offline Reconstruction

Snavely *et al.* proposed a system to reconstruct buildings from large collections of images from online community photo collections [26]. Later, Agarwal *et al.* presented an improved version of this system to perform large-scale reconstruction of urban areas using clusters of computers and huge amounts of images [2]. As a result both systems give point cloud reconstructions of the environment. Klopschitz *et al.* presented an approach to robustly reconstruct urban areas from large sets of unordered images [15]. A system fusing orientation and location priors to speed up reconstruction was presented recently by Irschara *et al.* [12]. Other approaches to modelling and reconstruction include for example [8, 23] or [25], amongst others. These approaches are considered to solve the task of modelling as an offline process, regardless of computational and memory demands in most cases.

2.2 Online Reconstruction

An example of a point cloud reconstruction generated online from a live video feed is the work of Klein and Murray [13]. The so-called PTAM system allows for concurrent mapping and tracking with a single camera in small workspaces, triangulating 3D feature points and using a set of keyframes for self-localization and tracking. Several extensions to PTAM have been proposed thereupon. Ventura and Höllerer worked on extracting planes from the sparse point cloud and determining the extents of the planes through dense image matching [32]. Van den Hengel *et al.* proposed a video-based in-situ reconstruction system with little manual interaction, allowing the user to accurately model objects in a short amount of time [29, 30]. Building upon previous work, PTAM is used to provide camera tracks plus a point cloud map and then the system switches into an interactive modelling mode to create meaningful 3D models of the environment. Newcombe and Davison recently demonstrated the dense reconstruction of workspaces from live video based on implicit surface calculations and dense optical flow [19]. The approach gives impressive results, whilst requiring a considerable amount of computational power, however.

Independent from PTAM-based methods, Pollefeys *et al.* described a system performing real-time reconstruction of urban areas using video feeds [22]. Their approach makes use of a multi-camera array and uses large amounts of processing power available from both the CPU and GPU of desktop systems. Bunnun and Mayol-Cuevas proposed a system to interactively build 3D wireframe models of objects [6]. The system uses a combination of a camera and computer mouse as a device for tracking and editing. Pan *et al.* presented the ProFORMA system which automatically and interactively generates 3D models of objects [21]. A 3D point cloud is reconstructed from a live video through on-line reconstruction. From the point cloud, a 3D Delaunay tetrahedralization is computed and used as the underlying structure in a probabilistic space carving approach. The most important aspect of the aforementioned approaches is their real-time or near real-time performance and the suitability to be applied in the context of AR scenarios.

2.3 Mobile Approaches

The most prominent approach to model generation on mobile phones is an improved PTAM version proposed by Klein and Murray [14]. The original PTAM was modified to run on an *Apple iPhone 3G*, and was demonstrated to allow mapping and tracking of workspace-sized areas using solely the computational and memory resources of the mobile phone. Lee *et al.* presented a mobile phone-based 3D modelling framework in which images were captured on the device whilst the 3D model reconstruction was performed on a remote server [16]. The mobile phone can only be considered as a thin client in this respect. Hartl *et al.* described a system to model small objects using a marker target [11]. The approach is based on space carving and the marker target is used to solve the foreground-background segmentation task. The approach works in real-time on a mobile phone and gives reasonable results on, for example, tokens in a board game. Apart from these examples, using mobile phones in any kind of reconstruction task has not yet been investigated widely.

2.4 Discussion and Considerations

The comparably small amount of computational and memory resources makes the use of currently known off-line or on-line approaches unreasonable on mobile phones and tablets. Even if, for some reason, these devices were able to run at a performance comparable to desktop computers for short periods of time, limited battery capacity would remain the main opposing factor.

Although this already describes a severe problem with traditional approaches, there are less obvious difficulties. Concerning video-based approaches, the need for constant tracks of natural features places an almost insuperable challenge using a mobile phone camera. Camera blur and jerky camera motion due to taking steps in the environment causes feature tracking to fail regularly, such that many of the required image correspondences can not be correctly established. Moreover the user is forced to constantly point the mobile phone camera up and towards the object to be reconstructed, at the same time moving around as smoothly as possible and trying to keep the camera as steady as possible. This unfortunately places unrealistic requirements in terms of user-friendliness. In this respect, video-based on-line reconstruction techniques might not be ideally suited for use on mobile phones.

As an alternative, many individual images could be captured using the mobile phone to perform an off-line reconstruction step afterwards. However, this approach also faces several difficulties. The narrow field of view on the mobile phone means that a large number of overlapping images are required to be taken in order to cover a building or scene completely (*c.f.* Figure 2). The complexity of a necessary bundle adjustment step grows very rapidly - $\mathcal{O}\left(n^3\right)$ - with the number of cameras to be estimated, thus increasing the overall computational load considerably. Furthermore, the user has to ensure that there are no gaps in the image set. From a set of images captured it is difficult to predict with certainty if a fully connected reconstruction is possible or not. This is usually circumvented through the use of Internet image databases, *e.g.* Flickr, which hold collections of images of the same place taken by many people. Needless to say that this is not a reasonable option in our scenario.

Many applications on mobile phones that leverage CV methods rely on a powerful remote server for solving the respective processing task. This leaves the mobile phone client as a device for data ac-

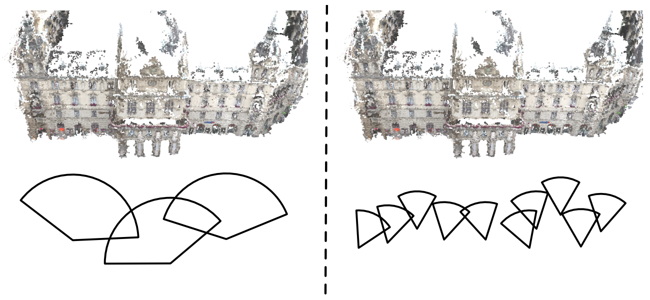
Figure 2: Left: Three wide field-of-view images capture almost the whole environment. Right: More narrow field-of-view images are required to capture the same scope. The dense point-cloud reconstruction was created using the approaches of Klopschitz *et al*. [15] and Furukawa *et al*. [10].

quisition and information visualization as described in the example before. In the context of 3D reconstruction, the amount of image data is substantial. Since data of high quality is needed, transferring large numbers of high-quality images or videos becomes quite costly in terms of data transmission. Moreover in certain areas, transferring large amounts of data from mobile devices may not be possible due to lack of network infrastructure.

Taking all these considerations into account we conclude that a novel and specifically designed approach is necessary to solve the task of 3D modelling and reconstruction on the targeted class of devices. The aim is to develop an approach that is at least one order of magnitude more efficient in terms of computational requirements, compared to desktop reconstruction approaches. Even more importantly, however, is the aspect of user-friendliness concerning the data acquisition conditions. The goal is to allow for a convenient way to capture the required images, at the same time minimizing their number and the required manual user interaction. A final requirement is to provide the user with a reasonable reconstruction result as quickly as possible, *i.e*. within the time frame of several seconds.

3 Approach Description

Since our approach consists of multiple complex components, a short overview is given about the design of our approach and to describe how these parts are interconnected. As the image acquisition algorithm is a module separate from the rest of our reconstruction pipeline, the acquisition procedure is also described.

3.1 Overview

In Figure 3 a flowchart of our approach is depicted. The image acquisition component is used to capture panoramic images. As soon as panoramic image capture begins, feature extraction starts in the background. Once the panoramic image acquisition is complete, image alignment and feature matching occur if more than one image is available. From the matches, epipolar constraints can be calculated using the 5-point pose or the 3-point pose algorithm, depending on the number of currently captured panoramas. From the camera pose estimates and established feature correspondences, 3D points are triangulated. The bundle adjustment step is run after each new image is finished, improving the accuracy of the camera pose estimates and the triangulated 3D points respectively with each additional image. From the camera geometry estimates, additional matches can be established, which in turn increases the number of reconstructed 3D points.

All these steps can be run in parallel to the actual image acquisition algorithm. After all images have been captured, surface reconstruction is performed using space carving, followed by generating

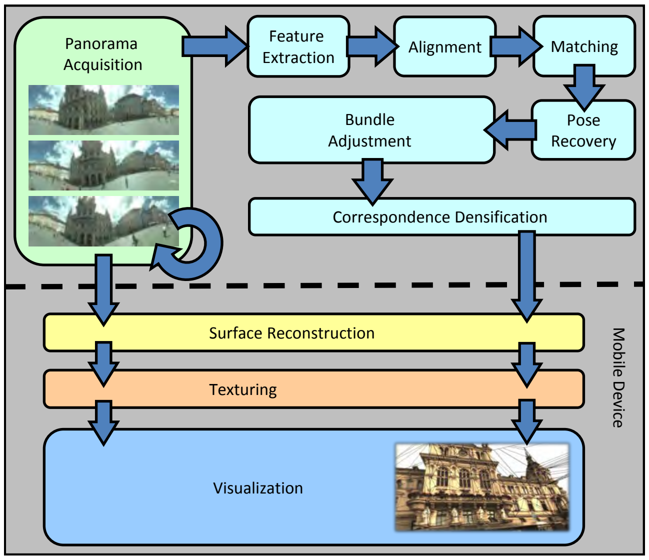

Figure 3: Workflow of our approach running entirely on mobile phone hardware. Most parts of the algorithm can run simultaneously with the panorama capturing application. Solely the surface reconstruction and texturing stage are run serially.

a representation for texturing the models. Finally, the reconstructed model can be visualized on the device itself.

3.2 Parallel Processing Pipeline

Almost all components of our algorithm can perform autonomously, while the image acquisition process is the only part that requires user interaction. Therefore most tasks in the pipeline can be performed in the background without any intervention necessary. This mainly includes the costly feature extraction and matching steps and bundle adjustment. Since mobile phones with multi-core CPUs are becoming available now, our approach can take advantage of this feature immediately. However, more intuitive possibilities exist to cut down the time needed for modelling. For the task of capturing multiple images the user has to change position. While this is done the device's computational resources are usually idle. In our approach these time slots are spent on the aforementioned processing steps. This leads to a very rapid final reconstruction time, as will be shown later.

3.3 Panoramic Image Creation

Panoramic images are created online in real-time whilst panning the camera around the user's location. We use a version of the PanoMT system described by Wagner *et al*. [33]. The mobile device is used to collect wide field-of-view images of the same building/outdoor scene from different locations. For each panoramic image, the program maps frames from the live video onto a cylindrical map whilst tracking the rotation of the device with respect to the map. Tracking and map building work at frame-rate (30Hz), even when using computational resources for other processing tasks. Two panoramic images captured using the system are depicted in Figure 4. Contrary to normal panorama stitching programs, the panorama is built interactively as the user sweeps the device over the view of the environment. Capturing a panorama takes only a few tens of seconds of interaction, as there is no need to carefully align a sequence of overlapping images. The algorithm makes the assumption that the camera is rotated around its optical center during panorama acquisition, which is often violated since the user is usually unable to perform a pure rotation around the device's optical center. Since the scene captured is usually far away compared to the distance the

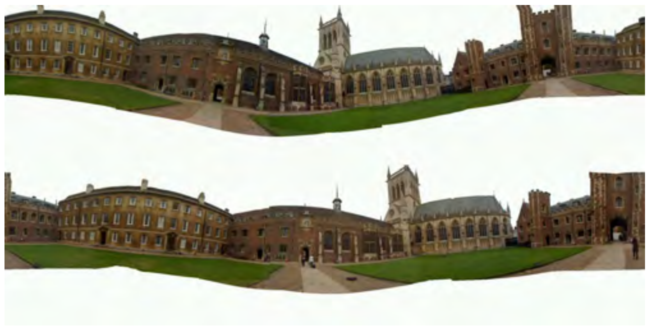

Figure 4: Two sample panoramic images from Cambridge, UK captured with the online panorama mapping algorithm.

optical center moves, however, panoramic images can still be successfully captured.

3.4 Cylindrical Camera Model

The cylindrical camera model used with panoramic images in our approach is formulated as follows. Let $\tilde{\mathbf{x}}_\mathbf{w} = (x_w, y_w, z_w, 1)^T$ be the (homogeneous) 3D coordinate of a point in the world frame and $\mathbf{E_c}$ an element of $SE(3)$ [31] representing the pose of camera c. The 3D coordinate of the point in the coordinate frame of camera c, $\mathbf{x_c} = (x_c, y_c, z_c, 1)^T$ is expressed as:

$$\mathbf{x_c} = \mathbf{E_c}\tilde{\mathbf{x}}_\mathbf{w} \tag{1}$$

The point of intersection, $\mathbf{x_i}$, of the ray cast by $\mathbf{x_c}$ and a cylinder of radius 1 with its axis aligned with the y-axis of camera c is given by:

$$\mathbf{x_i} = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \frac{1}{\sqrt{x_c^2 + z_c^2}} \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} \tag{2}$$

If the centre of the cylindrical image of size $S_x \times S_y$ is taken to be where the positive z-axis intersects the cylinder, and V is and vertical field of view (in radians), the function taking a point on the cylinder in 3D to cylindrical image pixel coordinates $(X, Y)^T$ (origin in top left) is:

$$\theta = (atan2(x_i, z_i)) \pmod{2\pi} \tag{3}$$

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} S_x \left(0.5 + \frac{\theta}{2\pi}\right) \pmod{S_x} \\ S_y \left(0.5 - \frac{y_i}{2\tan\frac{V}{2}}\right) \end{pmatrix} \tag{4}$$

Also, note that a correct rectification of the panoramas to obtain a level horizon is not necessary for reconstruction purposes. The cylindrical camera model maps the image pixels to rays in space and any subsequent computation of epipolar geometry and 3D camera pose is formulated in terms of rays emanating from the camera centers.

4 Rapid Reconstruction from Panoramic Images

The user records several wide FOV images using the panorama mapping algorithm. These images taken at different positions serve as the basis for our reconstruction pipeline. The reconstruction pipeline consists of several stages as described in this section.

4.1 Generating Correspondences

Robust and error-free feature correspondences between the individual panoramic images are required for the subsequent triangulation and reconstruction. We employ a multi-stage procedure to provide robustness against the distortions occurring in the cylindrical projections that lead to many wrong matches in scenes with repeated structure.

4.1.1 Feature Extraction

A large number of different natural feature types have been proposed in the context of reconstruction. Although any feature could theoretically be used in our approach, we experiment with two different feature types, a SURF-like descriptor originally proposed by Bay *et al.* [5], and a custom combination of FAST corners [24] and an 8×8 pixel image patch. This latter feature shall be referred to as a *FAST patch* for the remainder of this paper. SURF Features, like SIFT features [17], provide a degree of robustness to scale and affine transformation, but are faster to compute. Feature extraction speed is critical for rapid reconstruction and thus we use an efficient modified implementation of SURF [3]. Image features can be extracted in the background whilst the panoramic image is captured, without affecting the frame rate of the panoramic image capture. Features are extracted from "cells" (small rectangular regions) in the panorama as each cell is completed. Since cells of the panoramic image that have already been mapped remain unmodified later on, these areas are already processed for feature extraction.

4.1.2 Image Alignment

The panoramic images are centered around the first keyframe used to initialize the capture and thus, the cylindrical images may not be aligned to each other. To make subsequent feature matching more robust and efficient, we calculate a coarse alignment between multiple cylindrical images. Alignment is described only as a rotation around the cylinder axis, assuming a level horizon and similar standing height. We subsample the entire panoramic images to versions with a resolution of 36x4 pixels. This subsampling is created by averaging rectangular regions in the cylindrical images. The sum of squared differences (SSD) of the intensity values over 36 horizontal 1-pixel shifts is then calculated. The alignment rotation is chosen as the shift which generates the lowest SSD. This procedure is used to align all images to the first image through iteratively aligning each new image to the previous one. Using subsampled images of resolution 36x4 allows the images to be aligned within an accuracy of approximately 10°. In devices with digital compasses, the alignment procedure can be replaced by aligning each panorama based on the North direction.

4.1.3 Feature Matching

Exploiting the fact that we are using panoramic images which are roughly aligned, we can greatly constrain the search regions within which matching occurs for efficiency. Matching across aligned images is restricted to 10% of the image width (corresponding to 36° of the 360° panoramic image). Multiple match candidates for each feature are obtained in order to cope with the repetitive features found in many urban environments.

A different method of performing this matching, and which yields a faster matching time, is to store the feature descriptors in a tree. A k-means tree enables very rapid matching, but unfortunately at the expense of being too expensive to build at run-time. A k-d tree is an alternative, but search performance suffers for large dimensions. Thus the cost of building trees for faster searching often exceeds the computational requirements of matching exhaustively in strips in the panorama.

Using a constrained search region when matching between aligned panoramas increases search efficiency, but in certain scenarios limits the minimum depth a point can be successfully matched. In the direction of the baseline between two panoramas, no depth constraints exist, as distance only affects the point's imaged position in the vertical direction, but in the direction perpendicular to the baseline, a change in depth causes a point's projected position to also change in a sideways direction. Therefore, using a search width sets a closest distance in the direction perpendicular to the

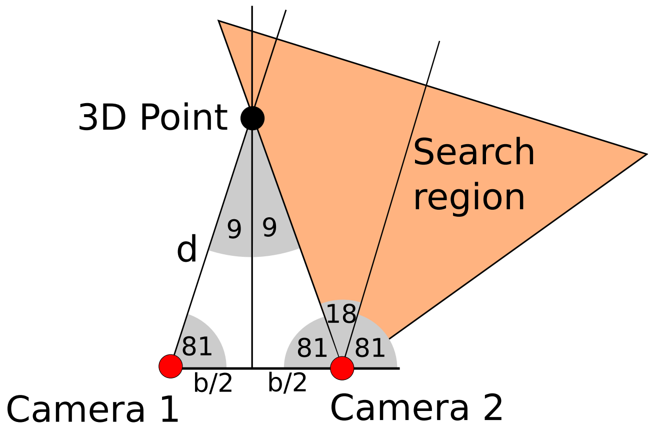

Figure 5: This diagram shows the configuration which generates the worst case minimum depth of a reconstructible point given a search region of 0.1 $\times$ image width, which corresponds to a 36 $^{\circ}$ search angle. The 3D point is equidistant from both cameras and lies on a line which is perpendicular to the baseline (of length b). The minimum distance to the camera in this configuration for a point to be successfully matched is denoted d. Conversely, a point in the image near the extended baseline can be modelled regardless of depth, as its sideways disparity will always lie in the search region.

baseline at which a point can be successfully matched, which is a function of the baseline distance, as shown in Figure 5 .

The worst case point is equidistant from both cameras, and lies on a line perpendicular to the baseline. Using the right angle definition of Sine, the ratio between the length of the baseline, b, and the minimum distance of the point, d, can be calculated.

$$\sin 9^{\circ} = \frac{b/2}{d} \tag{5}$$

$$d = \frac{b/2}{\sin 9^{\circ}} \tag{6}$$

$$d = 3.20b \tag{7}$$

This shows that the minimum distance for matchable points in this configuration is 3.20×baseline, so for a typical 1 meter baseline, points perpendicular to the baseline should be at a distance of at least 3.20 meters away. This limit only applies to points perpendicular to the baseline - points in line with the direction of the baseline can be modelled regardless of depth, as point depth only affects vertical position in this case. For points between these two extreme cases, the minimum reconstructible distance varies between 0 and 3.2× baseline.

Multiple hypothesis matches are generated, whereby all matches within a certain distance in feature space are recorded. Feature distances are computed using sum of squared differences (SSD). Multiple hypotheses matching is very important for many urban environments where there is repetitive texture in the scene. Multiple hypothesis matches are limited to a maximum of 8 hypotheses. The hypotheses are stored in a heap data structure, enabling the 8 lowest scoring hypotheses to be retained efficiently without having to keep track of a fully sorted list of match scores.

After the multiple hypotheses matches are found, we use a subset of these matches from which to draw hypotheses for robust estimation. This is due to the low inlier rate when drawing from all multiple hypotheses requiring too many iterations. The subset from which we wish to draw hypotheses from contains match hypotheses which are deemed "unique." Unique match hypotheses are defined as multiple hypotheses matches which either have only one hypothesis, or have multiple hypotheses where the lowest scored

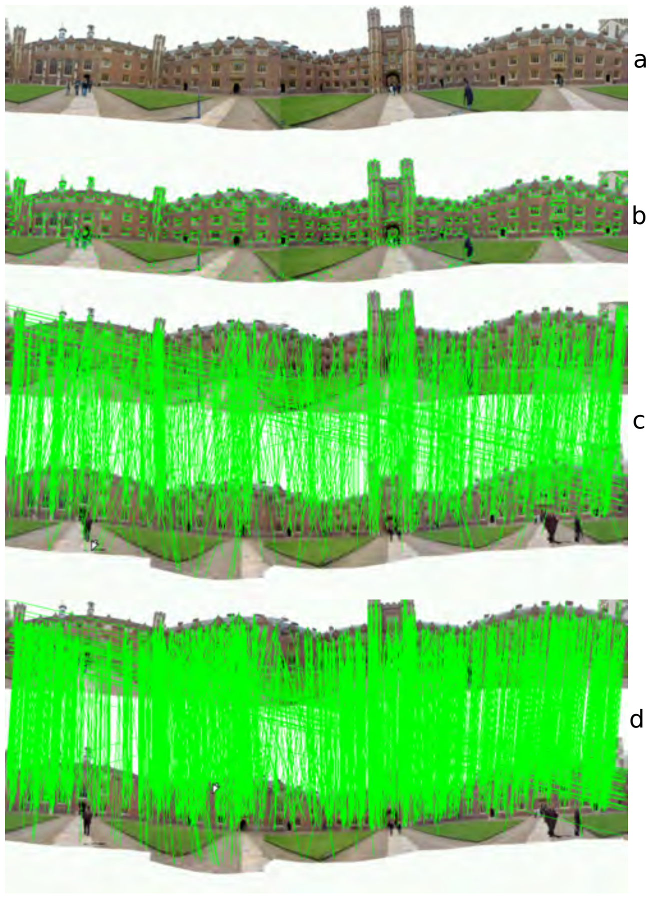

Figure 6: The matching process. (a). A panorama generated using PanoMT. (b). FAST features detected on the panorama. (c). unique matches found between two panoramas. (d). Epipolar inliers generated from all multimatches using unique matches as hypotheses for 5-point pose with PROSAC.

hypothesis is less than 50% the score of the next lowest.

4.2 Epipolar Geometry Recovery

When the first two images are collected, no information is known about the 3D locations of features identified in the images. We attempt to recover the epipolar geometry of the cameras by using 2D feature correspondences and the five-point pose algorithm[20] in conjunction with PROSAC[7] (features are sorted by match SSD). Hypotheses for PROSAC are generated using the unique matches set which was described in the previous section. Once an essential matrix with a high number of inliers is obtained from PROSAC, the matrix is decomposed into 4 possible solutions for the rotation and translation of the cameras (translation up to scale). The correct solution is chosen by taking a small set of inlier correspondences and triangulating their 3D position using each of the 4 poses, then choosing the solution which corresponds to the most points being in the positive direction of the ray for both cameras. Results from feature matching and epipolar geometry estimation can be seen in Fig. 6.

4.3 Bundle Adjustment

Having obtained the camera pose and a set of 2D feature correspondences for two frames, it would now be possible to simply find the closest point of approach of rays cast by the 2D observations to convert the 2D features to 3D landmarks (a 3D point with information about where and in which images it was observed). This approach is, however, limited to two images, and does not provide a way of reducing the re-projection error.

Thus instead, we formulated a triangulation scheme which expresses the probability distribution of the 3D position of a landmark in terms of camera pose and 2D observations only, and in a manner which works for more than 2 frames. A single 2D measurement of a 3D point corrupted by Gaussian noise on the image plane implies that the 3D point lies within a conical probability distribution. We approximate this distribution with a Gaussian. Multiplying the distributions from all 2D measurements together gives a Gaussian probability distribution, the mean of which is the triangulated 3D point position. We take the width of the Gaussian to be the width of the cone at the estimated position of the point, so this scheme must be iterated a few times to get an accurate estimate of the 3D point position and its uncertainty. This concept shown in Fig. 7, and is a form of implicit bundle adjustment [18].

This formulation analytically expresses the position of landmarks as a function of camera positions and observed pixel coordinates, and thus enables us to directly calculate the derivatives of how the reprojection error of landmarks changes with respect to camera pose. This allows us to perform bundle adjustment, the minimisation of re-projection error of landmarks with respect to both camera poses and landmark positions, using Levenberg-Marquadt. An M-estimator [28] is used in the least squares optimisation to make the bundle adjustment robust to outliers. Using this formulation of triangulation means that landmarks are not explicitly represented but are purely functions of camera pose (and fixed 2D observations), so 3D landmark positions never appear in the bundle adjustment, greatly reducing the computational complexity. This directly exploits the primary sparseness inherent in the structure from motion problem, instead of applying the Schur complement to achieve similar results. Classical bundle adjustment methods [27] can also be used to similar effect if using cylindrical camera Jacobians and ported to the mobile platform.

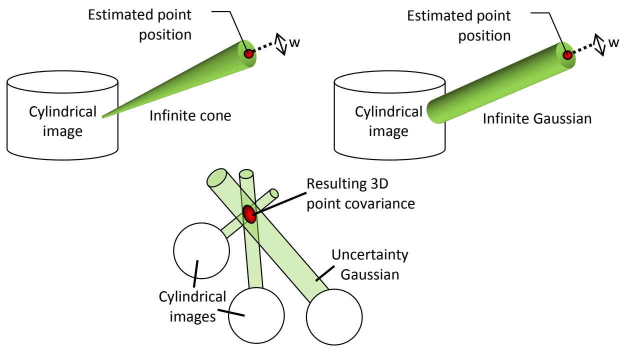

Figure 7: Diagram showing a form of implicit bundle adjustment where points are parameterised purely as a function of camera pose and fixed 2D observations. Top left: Conical uncertainty distribution of a single observation of a point in one cylindrical camera. Top right: Gaussian approximation of the conical distribution of same width as conical distribution at the predicted point depth. Bottom: Top view of multiple Gaussian distributions representing observations of the same point in different cameras, multiplied together to obtain covariance of 3D point.

4.4 Subsequent Pose Estimation and Feature Matching

Pose estimation between the first two images is conducted using the 5-point pose algorithm. After the initial pair, points are triangulated to obtain 3D landmark positions. This allows subsequent images to use 2D-3D correspondences and the 3-point pose algorithm [9] in conjunction with PROSAC to estimate camera pose. All the information available is then bundle adjusted to obtain accurate point locations and camera poses. Using the bundle adjusted pose, epipolar inliers are found for multiple hypothesis matches not associated to a landmark, allowing new landmarks to be created. In the final stage of feature matching, landmarks without a correspondence in the current frame are reprojected into the image and searched for within a 3 pixel radius.

4.5 Surface Recovery

The model output from bundle adjustment is a 3D point cloud, which whilst capturing the geometry of the observed features, is only a sparse representation of the scene. For many augmented reality applications, and indeed a useful visual representation of the scene for the user, a dense 3D model is required. A modified version of the probabilistic space carving algorithm described by Pan *et al*. [21] is used to obtain the surface model, with changes implemented to allow the system to perform inside-out space carving.

The convex hull of the point model is partitioned into tetrahedral voxels via a Delaunay tetrahedralisation using QHull [4]. 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. This process merely partitions the convex hull and thus places tetrahedra over concavities in the scene, and so further processing is required to remove these extra tetrahedra. Tetrahedra are carved away based on landmark visibility, with the probability of a tetrahedron existing being reduced if it occludes a landmark.

Let T_i represent a triangular face of a tetrahedral voxel, c the camera number, k the landmark number and $R_{c,k}$ the ray from the optical centre of camera c to landmark k. Let υ represent the set of all rays with indice pairs (c,k) for which landmark k is visible in camera c. Landmarks are taken as observations of a surface triangle corrupted by Gaussian noise along the ray $R_{c,k}$, centered at the surface of triangle T_i with variance σ^2. Let $x = 0$ be defined at the intersection of $R_{c,k}$ and T_i, and let x be the signed distance along $R_{c,k}$, 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:

$$P(L_k|R_{c,k},T_i) = \int_{-\infty}^{l_k} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-x^2}{2\sigma^2}\right) dx \tag{8}$$

This leads to a probabilistic formulation of tetrahedron carving:

$$P_{exist}(T_i|\upsilon) = \prod_{\upsilon} P_{exist}(T_i|R_{c,k}) \tag{9}$$

$$P_{exist}(T_i|R_{c,k}) = \begin{cases} P(L_k|R_{c,k},T_i) & \text{if } R_{c,k} \text{ intersects } T_i \\ 1 & \text{otherwise} \end{cases} \tag{10}$$

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 not necessary to test all tetrahedra, as surface tetrahedra act as a barrier, shielding tetrahedra below from influencing the surface mesh. In [21] tetrahedra are carved in a recursive manner, starting from those tetrahedra on the convex hull. If a tetrahedron is carved away, then its neighbours are tested for removal. This method works for the outside-in case, where the camera is moved around the outside of an object. When modelling scenes, however, it is often the case that the camera is within the convex hull of the points in the environment, in which case starting the carving process with convex hull tetrahedra may not carve away any tetrahedra at all (as occluding tetrahedra actually only exist within the convex hull). Therefore we generalise the carving process such that it works for both outside-in and inside-out cases.

The first step of the algorithm is to determine which tetrahedra contain camera points. This can be simply done by forming plane equations for each face of each tetrahedron. If the a point corresponding to the camera position lies on the "internal" side of the planes formed from each face of a tetrahedron, then the point is contained inside the volume of the tetrahedron. Tetrahedra which contain a camera point should be removed, and the recursive carving algorithm started at tetrahedra which neighbour the removed tetrahedron. If any cameras are outside the convex hull of the object, then the point lies within the infinite tetrahedron, and thus carving should be performed on its neighbours, which correspond to tetrahedra on the convex hull.

Once recursive carving is complete, the surface mesh can be extracted by marking all triangular faces of tetrahedra with no neighbours. The cylindrical image is mapped onto a cube in order to allow efficient and perspective correct texture mapping of the surface mesh.

5 Experiments

For our experiments we recorded several sets of panoramic images from buildings in Cambridge, UK, Graz, AT and Vienna, AT. As a mobile platform we used a *Nokia N900* smartphone with a 600MHz ARM Cortex A8 CPU and 256MB of RAM. The device runs *Maemo* OS and *FCam* camera drivers from Adams *et al*. [1] were installed to enable greater control of camera parameters. The entire reconstruction runs on the mobile phone platform, producing textured mesh models. For visualization purposes, the textured models were loaded and rendered on a desktop computer using *Geomview*[1].

5.1 Reconstruction Results

Figure 9 shows results from preliminary experiments to study the feasibility of the proposed reconstruction pipeline. The images of the church were captured using a PC-based panoramic image mapper, generating high resolution panoramic images of size 4000x1600 from a 640x480 video stream. The images were collected using a camera attached to a tripod which allowed the camera to rotate roughly around its optical center. These results demonstrated the proposed approach would work with idealised data (high resolution and fixed optical centre).

Figure 10 shows reconstruction results for an image sequence captured on a Nokia N900 smartphone and processed on the device itself. No tripod was used and the panoramas were captured freehand. Panoramic images were captured from a 320x240 video stream, with the final panoramas being of size 2048x512. The first reconstruction set (Figure 10 left middle), uses only 3 panoramas, and recreates the overall shape of the courtyard, closely resembling the ground truth shape as shown in the aerial image from Google Maps. For the second reconstruction set (Figure 10 left center), using 7 images, the reconstructed point cloud is more rectangular and complete due to the increased number of observations for bundle adjustment. The rendered mesh models show that views rendered from novel viewpoints resemble the real scene, as seen from the panoramic image. The generated mesh models are around 500-700KB in size excluding cube map textures for this dataset.

The effect of increasing the number of cameras can be seen in the bottom row of Figure 10. As expected, increasing the number of cameras increases the accuracy and completeness of the model as the additional observations result in lower uncertainties after bundle adjustment.

[1]www.geomview.org

Number of Images	3	4	5
Landmark inliers	499	511	539
Number of Triangles	1013	1120	1249
Save image&features	1.3s	1.3s	1.4s
Read image&features	0.9s	1.0s	1.0s
Align last image	0.5s	0.5s	0.5s
Multimatch last image	2.5s	2.4s	2.6s
3-point pose	2.1s	2.3s	2.6s
Match Densification	0.3s	0.4s	0.4s
Bundle Adjustment	2.1s	3.0s	4.6s
3D Delaunay Triangulation	0.5s	0.6s	0.7s
Space Carving	4.1s	5.2s	6.7s
Render most recent cube map	1.5s	1.5s	1.5s
Total Reconstruction Time:	14.5s	16.9s	20.6s

Table 1: Reconstruction times for increasing input image numbers on the Nokia N900 smartphone, using the FAST patch descriptor (for the same sequence as Fig. 10). Timings are from when the final panoramic image is complete.

5.2 Timing Results

Table 1 shows timing results for different numbers of input images. In this case, FAST patch features were used, although timings are similar for SURF descriptors as descriptor extraction itself, which would cause the performance difference, is performed in the background during panoramic image collection and thus not included in this table. As can be seen, elements contributing to the processing required between collecting panoramic images total around 8-12 seconds for 3-5 images, with the final textured reconstruction taking between 14-21 seconds after the final panoramic image is collected. A split bar chart of the timing results is given in Figure 8, enabling the contribution of each part of the algorithm to the reconstruction time to be seen. Including panorama creation time (∼10-20 seconds per panorama), a reconstruction from 3 panoramic images can be created in around 1 minute and a reconstruction from 5 panoramic images in around 2 minutes.

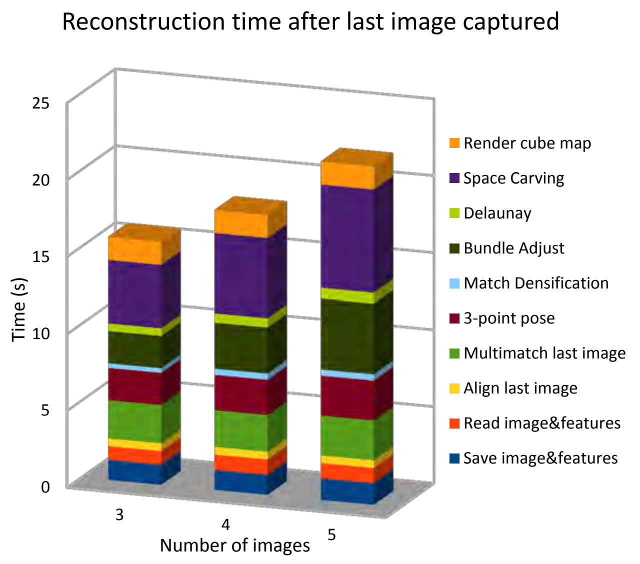

Figure 8: A split bar graph of the information in Table 1, showing how split and overall timings are affected by increasing the number of panoramic images.

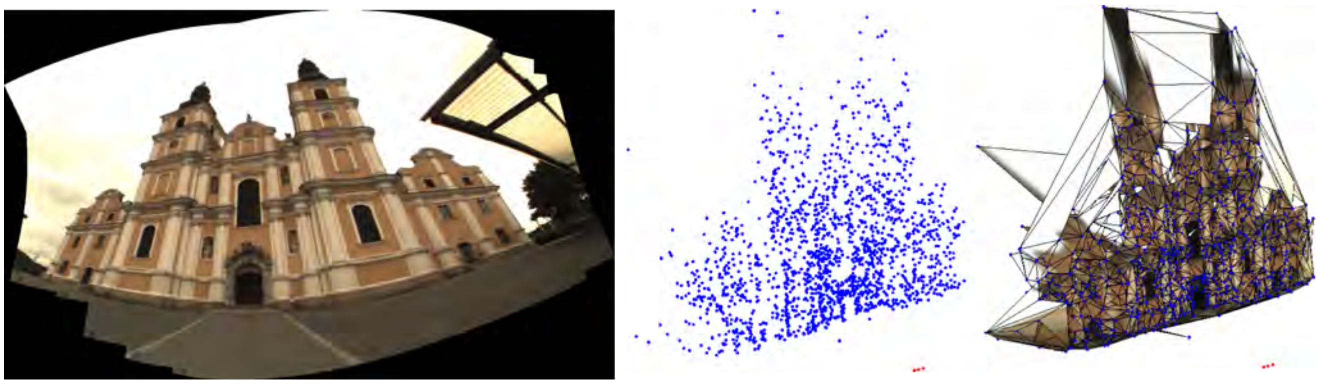

Figure 9: Graz Church. Early preliminary experiments performed as a proof of concept. Left: One of three wide field-of-view images captured on a tripod mounted webcam. Middle: Point cloud from bundle adjustment. Right: Texture mapped surface model.

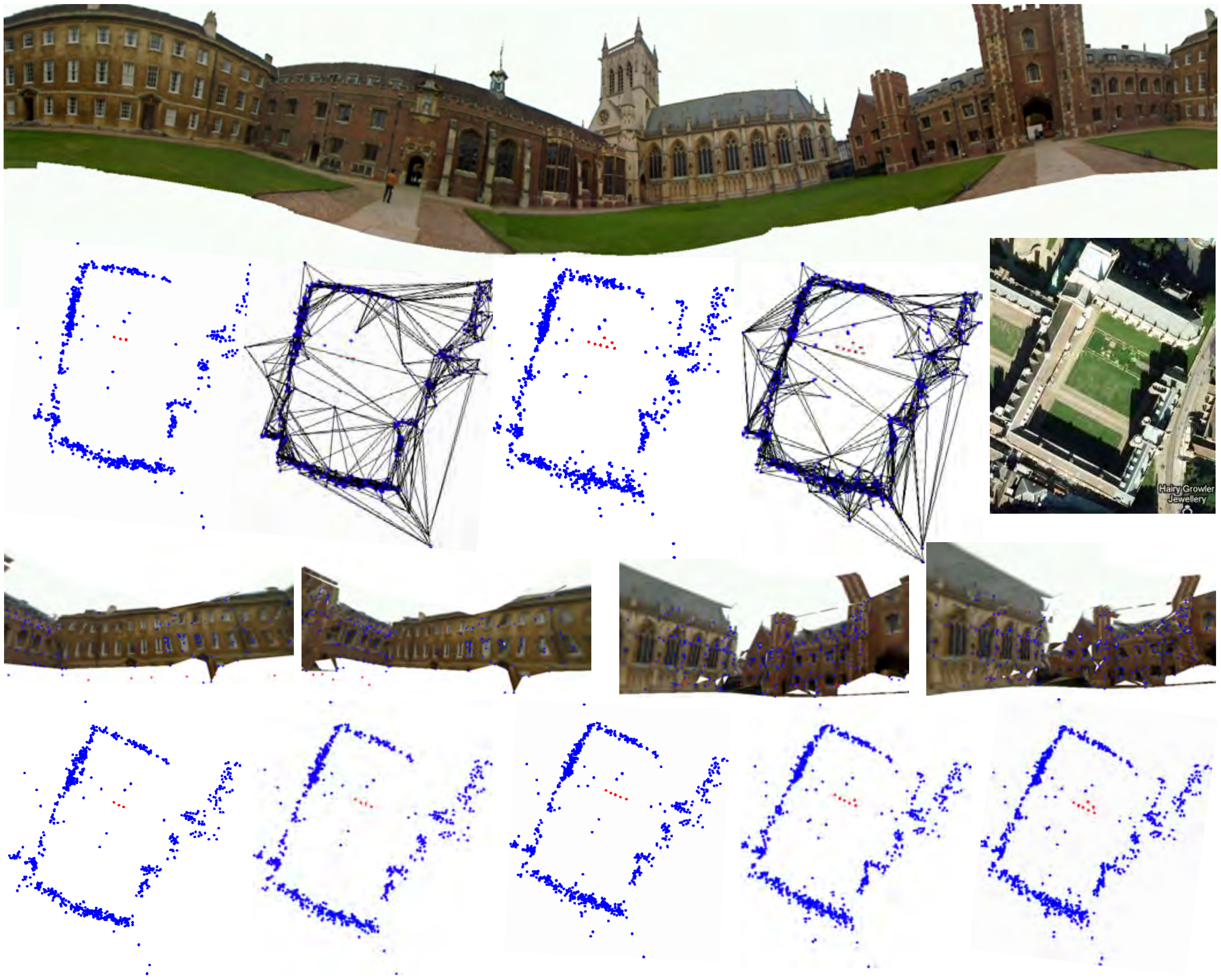

Figure 10: Cambridge, UK. 3D reconstruction results generated on the N900, using FAST patch descriptors. Top: Panorama generated free-hand using PanoMT. 2nd Row: Left to Right: Point cloud and mesh generated from 3 panoramas. Point cloud and mesh generated from 7 panoramas. Blue points represent landmarks, red points represent calculated camera positions. Google Maps aerial view of modelled location. 3rd Row: Rendered novel views of areas in the reconstructed model. Bottom: Reconstructed models using an increasing number of cameras, from 3 (left) to 7 (right).

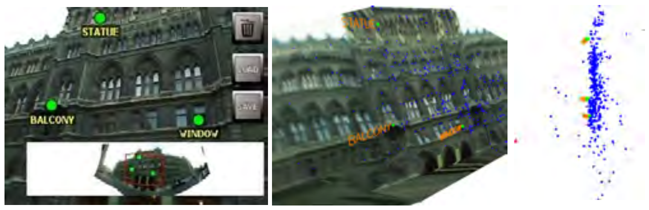

Figure 11: Left: Annotating a panorama with 2D labels during panorama creation [33]. Middle: Using the 3D model generated using our pipeline, 2D annotations made on a panoramic image during panorama creation can be upgraded to 3D annotations. Right: Side view of reconstructed point cloud model showing depth of labels (the textured mesh is hidden for clarity).

5.3 Sample AR Application

In order to demonstrate a sample application for this reconstruction pipeline, the existing use case of 2D panorama annotation as demonstrated by Wagner *et al.* [33] was extended to enable the annotation of 3D locations. Annotating a point on a panorama during panorama creation enables a 2D annotation to be placed on a panoramic image. The annotation is constrained to a ray emanating from the optical centre of the panorama through the pixel location of the annotation. This ray can be intersected with the mesh model generated by our reconstruction pipeline, thus providing a depth coordinate for the originally 2D annotation. Figure 11 shows a view of the panorama creation application during labelling and creation of a panoramic image, as well as the annotations registered in 3D once the 3D model is constructed.

5.4 Feature Comparison

During our experiments, two different types of feature were used. Figure 12 shows images of a scene reconstructed using FAST patches (left) and SURF (right). The distribution of features are shown below the rendered views. As can be seen, their distributions are very different, with FAST corners clustering around certain areas of high intensity change, whilst SURF features are spread much more evenly throughout surfaces. Using each type of feature has advantages and disadvantages. FAST features are very stable, but are not evenly spread out, thus causing loss of detail in certain areas. On the other hand, SURF features have a much better spread, but appear to be less stable. Overall, it was observed that model quality was largely similar, although the extraction of FAST patches was much faster than the extraction of SURF features, and thus a higher density of FAST patch descriptors could be extracted in the background for each cell without affecting the frame-rate of the panorama creation tool.

5.5 Optical centre translation analysis

Translation of the optical centre during collection of a panorama has two effects. One effect is that it can cause the seam of the panorama to overlap, the amount by which is dependent on scene depth and the amount of translation incurred. This induces a horizontal angular error to rays from the optical centre of the cylindrical camera to pixels in the image. A diagram of this effect can be seen in Fig. 13. A typical scene of depth $\sim$10m, with radius of circular optical centre motion $\sim$0.05m leads to an angular error from 0 (in the direction of the first image of the panorama) up to $\pm 0.5^\circ$ or 5 pixels in the image (in the area near the seam). In reality this is lower due to the first image already filling the 66° horizontal field of view of the N900 with no optical centre motion (image is taken instantly), and optical centre motion not always accumulating angular error in the same direction. The second effect of moving the optical centre is the effect of camera position on the ray angle. Taking a point 10 metres away and 10 metre high (worst case near the vertical limit of

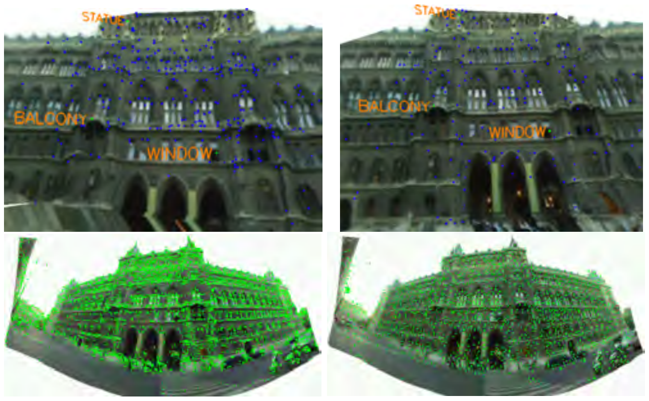

Figure 12: Top: Model reconstructed using FAST patch descriptors (left) and SURF descriptors (right). Bottom: Distribution of FAST features (left) and SURF features (right).

field of view), it can be seen that a deviation of 0.05m to the camera's distance leads to a vertical angular error of 0.15° (0.8 pixels). The main error is thus the horizontal angular error, which could be reduced by capturing the enlarged panorama and loop closing, then resampling the image such that it maps back into the correct sized image (and thus fits into a 360° cylindrical panorama).

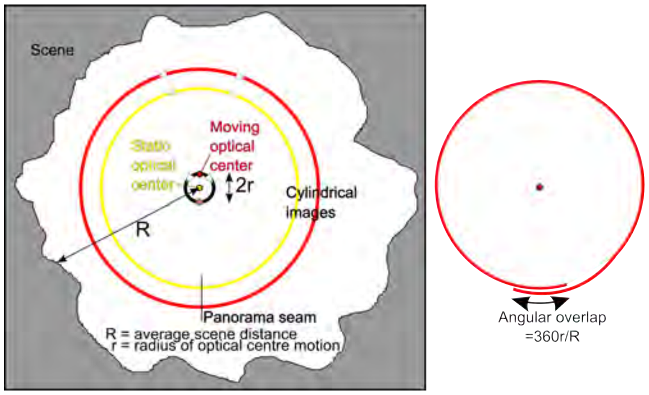

Figure 13: Angular error arising from optical centre motion. Left: Yellow and red cylindrical panoramas captured respectively with a static optical centre and an optical center that moves in a circular motion as the panorama is being captured. The panoramic image is captured clockwise for 180 degrees before returning to the centre and capturing in the anticlockwise direction. Translation in the optical centre leads to an increase in the size of the cylindrical image which is dependent on scene depth and radius of motion. Right: when the red cylindrical image is mapped to the actual size of the panorama (yellow circle), the image is larger than 360 degrees due to the optical centre translating, leading to the panorama seam overlapping.

6 Conclusion

We presented a novel approach to scene reconstruction on mobile devices capable of producing dense textured models just several seconds after panoramic image capture. The area of dense scene reconstruction on mobile phones is relatively unexplored, and we provide some of the first reconstruction results demonstrating what is possible on these devices with very limited computational power.

Our approach employs the use of an easy and user friendly method of capturing input data through the on-line creation of panoramic images, and exploits background processing and idle processing time between image capturing to reduce final computation time. The reduction in data redundancy is exploited in order to be able to generate textured reconstructions in just several seconds on handheld devices, enabling immediate use of these models in Augmented Reality applications. Results from different datasets are shown, ranging from full 360 degree panoramas to wide field of view images of single buildings. We also demonstrate a sample use case of the reconstruction system, enabling the placement of 3D labels in a scene via simple user interaction during panorama creation.

In the future, it would be interesting to be able to fuse the system with additional mobile phone sensor information such as GPS, which would allow images to be registered (noisily) to the world coordinate frame, or the accelerometer, which would enable the detection of significant motion of the optical center, which could cause the algorithm to fail. Another avenue for future work would be the exploration of immediately tracking a scene (in full 6 degrees of freedom) from the rapidly generated mesh model and landmarks.

Acknowledgements

This work was partially funded through the Christian Doppler Laboratory for Handheld Augmented Reality and the project HYDROSYS (EU FP7/DGINFSO grant 224416).

References

[1] A. Adams, E.-V. Talvala, S. H. Park, D. E. Jacobs, B. Ajdin, N. Gelfand, J. Dolson, D. Vaquero, J. Baek, M. Tico, H. P. A. Lensch, W. Matusik, K. Pulli, M. Horowitz, and M. Levoy. The Frankencamera: an Experimental Platform for Computational Photography. *ACM Trans. Graph.*, 29:29:1–29:12, July 2010.

[2] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. *2009 IEEE 12th International Conference on Computer Vision*, (Iccv):72–79, 2009.

[3] C. Arth, D. Wagner, M. Klopschitz, A. Irschara, and D. Schmalstieg. Wide area localization on mobile phones. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pages 73–82. IEEE, 2009.

[4] C. Barber, D. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996.

[5] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Computer Vision–ECCV 2006*, pages 404–417, 2006.

[6] P. Bunnun and W. W. Mayol-Cuevas. Outlinar: an assisted interactive model building system with reduced computational effort. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '08, pages 61–64, Washington, DC, USA, 2008. IEEE Computer Society.

[7] O. Chum and J. Matas. Matching with PROSAC-progressive sample consensus. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 220–226. IEEE, 2005.

[8] N. Cornelis, K. Cornelis, and L. Van Gool. Fast compact city modeling for navigation pre-visualization. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1339 – 1344, 2006.

[9] M. Fischler and R. Bolles. 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] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Clustering Views for Multi-View Stereo. http://grail.cs.washington.edu/software/cmvs.

[11] A. Hartl, L. Gruber, C. Arth, S. Hauswiesner, and D. Schmalstieg. Rapid reconstruction of small objects on mobile phones. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Embedded Computer Vision Workshop*, June 2011.

[12] A. Irschara, C. Hoppe, H. Bischof, and S. Kluckner. Efficient structure from motion with weak position and orientation priors. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Workshop on Aerial Video Processing*, June 2011.

[13] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[14] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '09, pages 83–86, Washington, DC, USA, 2009. IEEE Computer Society.

[15] M. Klopschitz, A. Irschara, G. Reitmayr, and D. Schmalstieg. Robust incremental structure from motion. In *3DPVT10*, 2010.

[16] W. Lee, K. Kim, and W. Woo. Mobile phone-based 3d modeling framework for instant interaction. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1755 –1762, 27 2009-oct. 4 2009.

[17] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[18] Z. Moore, D. Wright, D. E. Schinstock, and C. Lewis. Comparison of bundle adjustment formulations. *American Society for Photogrammetry and Remote Sensing*, 2009.

[19] R. Newcombe and A. Davison. Live dense reconstruction with a single moving camera. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1498–1505. IEEE, 2010.

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

[21] Q. Pan, G. Reitmayr, and T. Drummond. ProFORMA: Probabilistic feature-based on-line rapid model acquisition. In *Proc. 20th British Machine Vision Conference (BMVC)*. Citeseer, 2009.

[22] M. Pollefeys, D. Nistér, J. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. Kim, P. Merrell, et al. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2):143–167, 2008.

[23] 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*, 59:207–232, September 2004.

[24] E. Rosten and T. W. Drummond. Machine Learning for High-Speed Corner Detection. In *European Conference on Computer Vision (ECCV)*, 2006.

[25] G. Schindler, P. Krishnamurthy, and F. Dellaert. Line-based structure from motion for urban environments. In *Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, 3DPVT '06, pages 846–853, Washington, DC, USA, 2006. IEEE Computer Society.

[26] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. In *ACM SIGGRAPH 2006 Papers*, pages 835–846. ACM, 2006.

[27] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment:a modern synthesis. *Vision algorithms: theory and practice*, pages 153–177, 2000.

[28] J. Tukey. Exploratory data analysis. *Reading, MA*, 1977.

[29] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. Videotrace: rapid interactive scene modelling from video. *ACM Trans. Graph.*, 26, July 2007.

[30] A. van den Hengel, R. Hill, B. Ward, and A. Dick. In situ image-based modeling. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pages 107–110. IEEE, 2009.

[31] V. Varadarajan. *Lie groups, Lie algebras, and their representations*. Prentice Hall, 1974.

[32] J. Ventura and T. Höllerer. Online environment model estimation for augmented reality. In *Mixed and Augmented Reality. IEEE/ACM International Symposium on, New York, NY, USA*. Citeseer, 2009.

[33] D. Wagner, A. Mulloni, T. Langlotz, and D. Schmalstieg. Real-time panoramic mapping and tracking on mobile phones. *VR*, 2010.