3D RECONSTRUCTION OF ARCHITECTURAL SCENES FROM UNCALIBRATED VIDEO SEQUENCES

Jan-Michael Frahm[1], Marc Pollefeys[2], Brian Clipp[1], David Gallup[1], Rahul Raguram[1], ChangChang Wu[1], and Christopher Zach[1]

[1]Department of Computer Science
The University of North Carolina at Chapel Hill
CB 3175, Chapel Hill, NC 27599 USA
{jmf,bclipp,gallup,rraguram,ccwu,cmzach}@cs.unc.edu

[2]Department of Computer Science
ETH Zürich
Universitatstrasse 6, CH-8092 Zürich, Switzerland
marc.pollefeys@inf.ethz.ch

KEY WORDS: structure from motion, 3d reconstruction, self-calibration, bundle-adjustment, gpgpu

ABSTRACT:

In this paper we present a system for three-dimensional reconstruction of architectural scenes from uncalibrated videos. These videos might be recorded using hand-held cameras, downloaded from the internet or taken from archival sources. Because we do not require prior knowledge of the camera's internal parameters such as focal length, center of projection, and radial distortion we can deal with videos from uncontrolled sources. We present fast algorithms for 2D feature tracking on the GPU, real-time robust estimation and bundle adjustment. We also demonstrate a new type of feature, the viewpoint invariant patch (VIP), in its application to loop detection and closing.

1 INTRODUCTION

In recent years, the topic of creating 3D reconstructions of buildings and landmarks from image and video data has received much attention. Some of these systems use video data in conjunction with GPS/INS information and produce detailed 3D models in the form of textured polygonal meshes, in real-time (Pollefeys et al., 2008). Other systems have been developed that leverage internet photo collections and produce sparse reconstructions of various landmarks, which may then be used for visualization (Snavely et al., 2008, Li et al., 2008). Applications such as Google Earth and Microsoft Virtual Earth have been very successful in delivering effective visualizations of large scale models based on aerial and satellite imagery to a broad audience. Enormous amounts of video and image data are collected every day, using a variety of recording devices. Both the sheer volume of data, along with the large variation in recording devices, pose significant challenges for systems that collect, process and visualize this information.

In this paper, we introduce a system for three-dimensional reconstruction of architectural scenes from uncalibrated videos. These videos might be recorded using hand-held cameras, downloaded from the internet or taken from archival sources. Because we do not require prior knowledge of the camera's internal parameters such as focal length, center of projection, and radial distortion we can deal with videos from uncontrolled sources. Our approach uses computationally efficient components and computation strategies to achieve real-time processing for various stages of the reconstruction pipeline. While the system in (Pollefeys et al., 2008) made use of calibrated cameras along with GPS/INS information, the current system is capable of handling uncalibrated cameras, and uses purely visual data.

Reconstruction from uncontrolled video poses several challenges, since cameras are typically in auto-exposure mode, which breaks the standard assumption of image brightness constancy. In addition, the radiometric calibration of the camera is typically unknown, implying that the appearance change of the scene could potentially be non-linear. Since we do not use precalibrated cameras, the intrinsic parameters of the cameras are unknown and can potentially even vary over time. Given the incremental nature of video-based structure-from-motion algorithms, systems tend to accumulate drift, which can only be compensated through image based topology detection. The system we present tackles all these challenges with efficient techniques to produce an initial video based 3D reconstruction. Following this, video-based topology detection is performed in near real-time, to identify loops in the video sequence. The final step of global error mitigation through efficient bundle adjustment is performed as a post-processing on the full extracted model, or large partial models.

The paper is organized as follows: Section 2 is an overview of our system; Section 3 briefly presents related work; Section 3 describes the various modules of the reconstruction pipeline. Section 4 contains results and Section 5 concludes the paper.

2 SYSTEM OVERVIEW

In this section, we provide a brief overview of the 3D reconstruction system. A more detailed description of the various modules follows in later sections of the paper.

1. **2D tracking of salient image features**: This constitutes the first module of the processing pipeline, where we obtain tracked feature correspondences over consecutive video frames, potentially belonging to the same 3D point. This module runs on the GPU and is capable of tracking 1000 features at speeds exceeding 200Hz on recent GPU hardware (Zach et al., 2008). The tracker module also estimates the gain of the camera, since for outdoor sequences, the ability to compensate for brightness change is often valuable.
2. **Estimation of camera parameters**: We seek to operate with video data acquired from uncontrolled sources, and thus employ a simple technique to recover the parameters of the camera. Given that the quality of the reconstruction depends primarily on the focal length, we use the technique from (Mendonca and Cipolla, 1999) to estimate this parameter, while assuming standard values for the remaining intrinsics (i.e., principal point in the center of the image, zero skew, and unit aspect ratio). Once this has been obtained, we compute a sparse reconstruction of a segment of the video and perform bundle adjustment to obtain an estimate of the radial distortion. This can be thought of as a preprocessing step, performed once on a subset of the sequence, after

which we assume that the intrinsics are fixed. The entire video sequence is then processed using the estimated camera parameters to obtain the final 3D model.

3. **3D camera pose estimation**: Given the camera intrinsics, the full video sequence is fed into the reconstruction pipeline, starting with 2D tracking. Given 2D feature correspondences between consecutive frames, we can estimate the pose of the camera by using visual odometry techniques (Nister, 2004, Haralick et al., 1994). In addition, we employ a fast and accurate technique for robust estimation (Raguram et al., 2008), which provides the ability to reliably estimate camera pose over a wide range of inlier ratios, in real-time.

4. **Stereo depth estimation**: The stereo module computes depth maps from the camera pose information and the images, using a fast GPU implementation of an advanced multi-view plane sweeping stereo algorithm. Our algorithm has provisions to deal with nonfronto-parallel surfaces, occlusions and gain changes.

5. **Depth map fusion**: Following stereo, this module combines multiple depth maps to reject erroneous depth estimates and remove redundancy from the data, resulting in a more accurate and smaller in size set of depth maps.

6. **Model generation**: This module creates a triangular mesh for each fused depth map and determines the texture mapping. It also removes duplicate representations of the same surface and fills some of the holes.

7. **Viewpoint Invariant Patch Extraction and Matching**: The viewpoint invariant patch (VIP) (Wu et al., 2008) constitutes an extension of wide baseline features into three dimensions. Model matching with VIPs forms the basis for robust loop detection and closing in our reconstruction system. Given a video sequence taken of the outer walls of a building that overlaps at the ends, we can generate a series of locally accurate 3D models of the building using the steps outlined above. These 3D models can then be matched using VIP features to find the loops in the video.

8. **Bundle adjustment**: The 3D models generated by the modules described above are obtained by using relatively local information from the images. Thus, the reconstructed models and the estimated camera poses do not accurately match the measurements in the images. In order to obtain results with the highest possible accuracy, an additional non-linear optimization step, generally referred to as bundle adjustment, is applied at various stages in the processing pipeline.

3 3D RECONSTRUCTION SYSTEM

3.1 Focal Length Estimation

In order to allow for flexibility with respect to producing reconstructions from uncontrolled video sources, we employ self calibration to recover the camera intrinsics. We perform this as a preprocessing step, which is done once on a subset of the video sequence. A commonly used technique for self calibration is to first build up a projective reconstruction and then upgrade this to a metric reconstruction by imposing constraints on the intrinsic camera parameters, such as constant intrinsics (Faugeras et al., 1992, Triggs, 1997) or assuming that some intrinsics are known and others vary (Pollefeys and Gool, 1999, Pollefeys et al., 1999). We circumvent the issue of maintaining a projective reconstruction by employing a simpler technique based on (Mendonca and Cipolla, 1999), which leverages properties of the essential matrix to recover camera intrinsics. In particular, we recover the focal length of the camera (which is assumed to be constant over the length of the video), while assuming the other intrinsics are known.

Table 1: Estimated focal lengths for two sample sequences, before and after bundle adjustment.

	Using calibration pattern	Self-calibration (before bundle)	Self-calibration (after bundle)
Focal length (sequence 1)	1316.1	1417.9	1347.2
Focal length (sequence 2)	1339.9	1092.6	1353.8

The technique works by translating constraints on the essential matrix into constraints on the intrinsics, thus allowing a search in the space of intrinsic parameters in order to minimize a cost function related to the constraints. By considering a segment of video containing at least n frames, satisfying $nn_k + (n-1)n_f \geq 8$ (where n_k is the number of known intrinsics and n_f is the number of unknown, but fixed, intrinsics), we compute a set of fundamental matrices by matching points pairwise. Since for a pair of views i, j, the essential matrix is related to the fundamental matrix as $E_{i,j} = K_j^T F_{i,j} K_i$, constraints on the essential matrix are translated into constraints on the calibration matrices K_i and K_j. It is then possible to establish a cost function to be minimized in the entries of the calibration matrices K_i, $i = 1, ..., n$. Additional details regarding this technique may be obtained from (Mendonca and Cipolla, 1999).

Note that the procedure described above does not account for radial distortion, which in practice significantly affects the quality of the reconstruction. We use bundle adjustment (described in Section 3.3) in order to perform efficient non-linear optimization of intrinsic camera parameters, and also to account for the non-linear radial-distortion of the camera lens. Sample results for focal length estimation on two sequences are shown in Table 1. After the computation of the intrinsic camera parameters and the two-view relations a sparse Euclidian reconstruction is established, as described below. This provides a camera path that globally accumulates drift over time but is locally virtually drift free. The system deploys this fact to estimate local geometry that is a correct approximation of the local 3D scene.

3.2 Initial Sparse Structure From Motion

After the preprocessing step that estimates the focal length and remaining camera calibration parameters, we assume a fully internally calibrated camera in the following steps to robustify the estimation process.

3.2.1 GPU-Accelerated KLT Tracking With Camera Gain Estimation Tracking of simple corner-like features in a video sequence (Lucas and Kanade, 1981, Shi and Tomasi, 1994) is still often applied due to its simplicity and its excellent runtime performance. In particular, augmented reality applications (e.g. using the OpenVIDIA framework (Hill et al., 2004, Fung et al., 2005)) and high performance scene reconstruction from streamed video (Pollefeys et al., 2008) require very fast detection of interest points and subsequent search for potential correspondences from a video source.

For best run-time performance we employ a GPU-based implementation of feature detection and tracking. In addition, we relax the brightness constancy assumption between images and allow slowly varying camera exposure settings in the image sequence. Assuming a linear camera response function, the varying brightness of pixels can be corrected by multiplication with a single

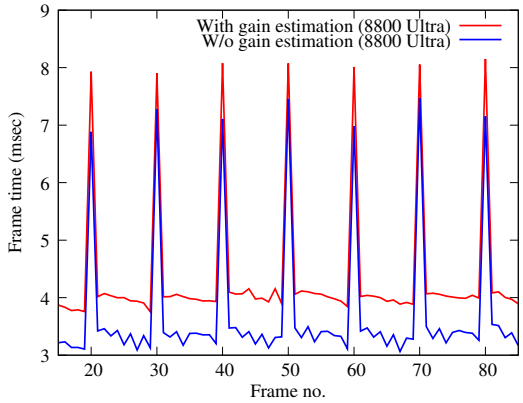


Figure 1: Observed frame times for tracking up to 1000 features on a Geforce 8800 Ultra (with and without simultaneous gain estimation).

gain ratio value globally valid over the full image. The method proposed in (Kim et al., 2007) for simultaneous tracking and camera gain estimation relies on sequential and therefore CPU-based computations. A full data-parallel approach for simultaneous tracking and gain determination and the respective GPU implementation is presented in (Zach et al., 2008). In summary, the estimation of a global gain ratio parameter between images is replaced by a parallel estimation of a gain value for each feature track. The estimated gain changes existing in parallel are globally regularized to obtain one consistent estimate for all feature tracks. This method essentially corresponds to a block Jacobi approach for a data-parallel and iterative solution procedure for linear systems of equations. Lost tracks or newly appearing features are frequently detected by computing the respective cornerness measure. Putative new features are subject to a spatial non-maximum suppression scheme, and feature positions are extracted by a hierarchical compaction approach (Ziegler et al., 2006).

Tracking of up to 1000 features with simultaneous gain estimation can be done at frame rates exceeding 200Hz on recent GPU hardware. Fig. 1 displays the measured frame times on a Geforce 8800 Ultra with and without simultaneous gain estimation enabled. The regularly occuring peaks correspond to increased runtime during feature redetection. In many outdoor sequences the ability to compensate for brightness changes is very valuable. The accuracy of the gain changes reported by our feature tracker in comparison to the camera reported values is visualized in Fig. 2.

3.2.2 3D camera pose estimation using ARRSAC Given the 2D feature correspondences across video frames, the next step in the processing pipeline is the estimation of camera poses. In this system we rely purely on visual input, and thus the camera poses are obtained up to scale. Visual odometry is a well-studied problem, and we use efficient implementations of various techniques to perform pose estimation. In particular, the camera tracker is initialized using the 5-point algorithm (Nister, 2004), which computes the relative pose of three views, given the 2D correspondences between them. Following this, 3D points are triangulated, and subsequent camera poses are estimated from 2D-3D correspondences by the 3-point method (Haralick et al., 1994).

To provide robustness to noise and outliers we use ARRSAC (Raguram et al., 2008), which is a real-time RANSAC framework which provides reliable robust estimation within a fixed time-budget. The technique combines the breath-first paradigm of preemptive RANSAC (Nister, 2003) with a depth-first strategy to quickly discard contaminated hypotheses (Matas and Chum, 2005). While traditional RANSAC techniques face the problem of having to generate a very large number of hypotheses for low inlier ratios, we adopt a non-uniform sampling strategy (Chum and Matas, 2005) to preferentially generate better hypotheses earlier on in the sampling process. Since operating within a fixed time budget effectively places an upper limit on the maximum number of hypotheses that may be evaluated, incorporating prior knowledge into the sampling process allows us to perform reliable estimation in real-time, even for very low inlier ratios. In practice, ARRSAC operates well within the enforced time budget, with estimation speeds ranging between 55-350 Hz, depending on the inlier ratio.

3.3 Sparse Bundle Adjustment

The initial sparse model generated by the methods described in the previous sections is obtained by using relatively local information from the images. Therefore, the reconstructed sparse 3D model and the estimated camera poses do not accurately match the measurements in the images, i.e. the overall reprojection error is typically larger than one pixel. In order to obtain results with the highest possible accuracy, an additional refinement procedure, generally referred as bundle adjustment, is necessary. Bundle adjustment is a non-linear optimization method, which not only tackles the error in camera calibration and their respective poses, but also refines the inaccuracies in the 3D points and the 2D tracks. It incorporates all available knowledge—initial camera parameters and poses, image correspondences and optionally other known applicable constraints—, and minimizes a global error criterion over a large set of adjustable parameters, in particular the camera poses, the 3D structure, and optionally the intrinsic calibration parameters. In general, bundle adjustment delivers 3D models with sub-pixel accuracy, e.g. an initial mean reprojection error in the order of pixels is typically reduced to 0.2 or even 0.1 pixels mean error. Thus, the estimated two-view geometry is better aligned with the actual image features, and dense 3D models show a significantly higher precision. The major challenges with a bundle adjustment approach are the huge numbers of variables in the optimization problem and (to a lesser extent) the non-linearity and non-convexity of the objective function. The first issue is addressed by sparse representation of matrices and by utilizing appropriate numerical methods. Sparse techniques are still an active research topic in the computer vision and photogrammetry community (Dellaert and Kaess, 2006, Engels et al., 2006, Ni et al., 2007, Kaess et al., 2008).

Our implementation of sparse bundle adjustment[1] largely follows (Dellaert and Kaess, 2006) by utilizing sparse Cholesky decomposition methods in combination with a suitable column reordering scheme (Davis et al., 2004). For large data sets this approach is considerably faster than bundle adjustment implementation using dense matrix factorization after applying the Schur complement method (e.g. (Lourakis and Argyros, 2004)). Our implementation requires less than 20min to perform full bundle adjustment on a 1700 image data set (which amounts to 0.7sec per image). This timing result refers to full bundle adjustment for the complete data set, which is only required in combination with loop detection. In many applications a "streaming" version of bundle adjustment only optimizing the recently added views and 3D points is sufficient. Such a reduced bundle adjustment optimizing the 10 recently added poses and the respective 3D structure achieves 50 frames per second.

Since only weak calibration information is available from camera self-calibration, it is an absolute necessity to adjust the camera intrinsics as well. Often, the zoom of the camera does not vary appreciably within the sequence of interest and accordingly

[1] available in source at http://cs.unc.edu/∼cmzach/opensource.html

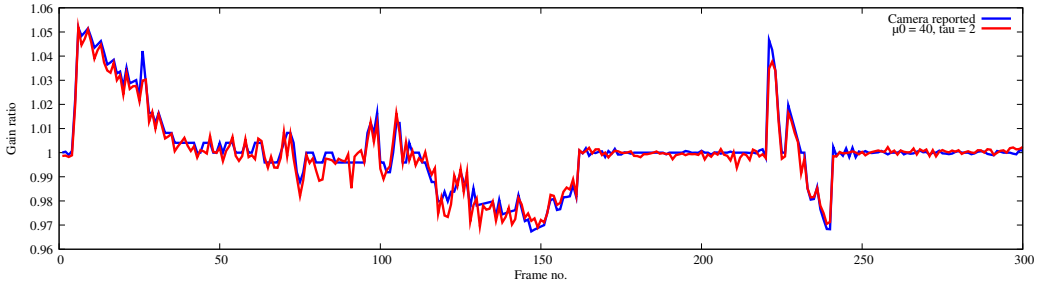


Figure 2: Camera reported ground truth gain ratios (blue) and estimated ones returned by the proposed feature tracking approach (red).

we can assume fixed intrinsics for the camera (or set of cameras). Enforcing common intrinsic parameters where appropriate is highly beneficial for the quality of the final reconstruction. In the case where the camera parameters change over the duration of the sequence, we can optimize separate sets of parameters for each camera, which potentially increases the noise sensitivity of the estimation process.

3.4 Dense Model Generation

After the initial sparse bundle adjustment we have a set of camera poses which have a high degree of local accuracy but still suffer from long term error accumulation. In order to reduce this, we must find areas of the video that view the same scene structure at distinct parts of the video. The simplest of these cases is when the camera moves in a loop such as recording the facades around a building. Once we have detected the same structure in two parts of the video we can use a final bundle adjustment to reduce the accumulated camera error and arrive at a maximum a-posteriori approximation of the scene structure and camera path.

Detecting loops in our system involves first generating locally accurate estimates of the dense scene geometry. In contrast to sparse geometry which only estimates the scene structure for a few distinctive scene features, dense estimation attempts to find the scene depth for every pixel in a set of key frames. The key frames are selected such that combining their depth maps provides a continuous coverage of the scene geometry.

Dense estimation is broken into two steps, both of which leverage the parallel computation power of the GPU to significantly speed up computation. The two stages are a plane sweeping, multi-view stereo algorithm which generates depthmaps and a depthmap fusion algorithm. The stereo algorithm works by back projecting a set of n images taken from known camera poses onto a plane in the scene. Where the plane intersects the scene structure all of the back-projected images will be locally similar but where the scene structure does not intersect the plane the back-projected images will differ significantly. By moving this back-projection plane through the scene we generate a correlation volume which gives us an estimate of the scene depth over a volume. Taking the minimum in this volume along the rays of one central camera gives us a depthmap for that central camera. We calculate these depthmaps in a sliding window through the video sequence using the GPU, taking advantage of the parallel nature of the plane sweeping stereo algorithm to speed the computation.

The second step in dense model generation is depth map fusion. This step takes a set of depthmaps which overlap, giving multiple measurements of the same scene structure. While plane sweeping stereo is fast it is also susceptible to noise. However, by looking for consensus between multiple depthmaps we can arrive at a final depthmap which is much more accurate than any of the individual stereo depthmaps. Consensus is measured based on occlusion relationships. More details on depthmap fusion can be found in (Merrell et al., 2007). Following depth map fusion we extract and match viewpoint invariant patches to detect loops.

3.5 Viewpoint Invariant Patch Extraction and Matching

The viewpoint invariant patch (VIP) is an extension of wide baseline features (SIFT (Lowe, 1999), MSER (Matas et al., 2002)) into three dimensions. These features are designed to find image regions that "look" the same from a variety of viewing directions (by "look", we mean have a similar feature descriptor). While typical wide baseline features are extracted in the image, VIP features are extracted on textured 3D surface patches. Extraction of VIP features starts with a multiple-view reconstruction of a scene which gives scene depth and texture from at least one viewpoint. From the scene depth locally quasi-planar regions are found and texture is applied to a plane fit to the local geometry. This textured surface is then projected into a camera which is fronto-parallel to the planar surface and uses an orthographic projection. We refer to this normalized view as an ortho-texture.

Once we have an ortho-texture we calculate whether the ortho-texture contains a scale space maxima in a way similar to SIFT. We then calculate the dominant gradient direction in the ortho-texture which, combined with the surface normal, defines the rotation of a local coordinate system attached to the surface. The depth of the scene point then gives the relative translation to the camera center of the VIP's local coordinate frame. Additionally, the size of the scale space maxima fixes the feature's relative scale to any other feature. Finally, we calculate a descriptor for the VIP feature based on the ortho-texture. Because the VIP feature is so information rich only one feature match is required to find the alignment between two 3D models. Given two 3D models of the same scene we can find a relative similarity transform between the two models using one feature correspondence because each feature includes its own local coordinate frame. This greatly speeds up model alignment and matching. Further we can find the model alignment in a hierarchical fashion, starting with alignment in scale, then rotation and finally translation.

Repetitive features can actually help in aligning models. For example, a brick wall is highly repetitive. Even if two bricks are incorrectly matched based on their descriptors the relative scaling of the bricks will still match the relative scaling of the model. Rotation also has this same property in many architectural scenes. Two windows that are mismatched will probably have the same orientation in space as the rest of the windows on a structure. After aligning scale and rotation it is much easier to find the correct common translation between two models. Model matching is the basis for robust loop detection and closing in our reconstruction system. Given a video sequence taken of the outer walls of a building that overlaps at the ends we can generate a series of locally accurate 3D models of the building using structure from motion. These 3D models can then be matched using VIP features to find the loops in the video. Later we show the results of one such loop detection and closing.

Figure 3: Representative frames from the 'Baity Hill' video sequence.

Efficient feature matching is also a concern for our system. We match VIP features based on their descriptor. A large scene such as a city center will contain many thousands of VIP features. Matching all those descriptors to each other would have a quadratic computational cost. We take a more efficient approach by using a vocabulary tree to find likely matches in a manner similar to (Nister and Stewenius, 2006, Fraundorfer et al., 2007). A vocabulary tree is built by applying K-means clustering to the high dimensional features in a hierarchical way. Starting with all features they are clustered into ten clusters in our implementation. Each of these ten clusters is then broken into another ten clusters. To find feature matches we determine if they map to the same leaf of the vocabulary tree. With one-hundred thousand clusters in a tree with a branching factor of ten and height 5 we reduce the comparisons necessary to find two features which are likely to match from n (the number of total features to compare) to 50. This offers a significant cost savings.

3.6 Final Bundle Adjustment and Dense Model Generation

Following VIP detection and matching we do another sparse bundle adjustment with the addition of the measurements of the VIP feature matches to the minimization. This bundle adjustment gives us a final sparse model of the scene and camera poses. We then do a final dense model generation using the same plane sweeping stereo and fusion as described previously to arrive at our final resulting model of the architectural scene.

4 EXPERIMENTAL RESULTS

In this section, we show some sample results obtained using the 3D reconstruction pipeline described in this paper. We have successfully processed videos at a number of resolutions and framerates, using uncalibrated cameras. The 'Baity Hill' video sequence (Figure 3) consists of images of resolution 1024x768, collected at 30 frames per second. Figure 4 show 2D tracks (in green) produced by the 2D tracker module running on the GPU. These 2D tracks, along with the estimated focal length from self-calibration are then used to generate an initial sparse reconstruction. Figure 5 shows the sparse reconstruction for the 'Baity Hill' sequence, obtained using the estimated focal length. The figure shows the model before bundle adjustment is performed. It should be noted that this model shows a large deviation from ground truth, mainly due to inaccurate focal length, along with radial distortion in the images. Figure 6 shows the sparse reconstruction after bundle adjustment, and it can be observed that this reconstruction is much closer to ground truth. After bundle adjustment, the model in Figure 6 has a mean reprojection error of < 0.2 pixels. Figure 7 shows some views of the textured 3D model obtained from the 'Baity Hill' sequence.

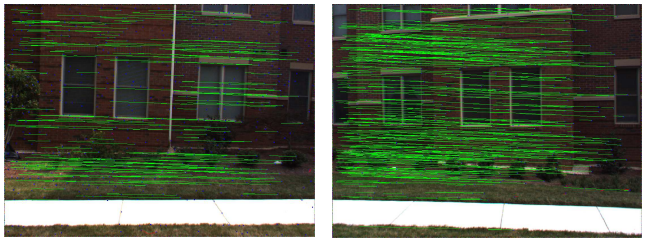


Figure 4: 2D feature tracks (in green) are shown superimposed on example video frames.

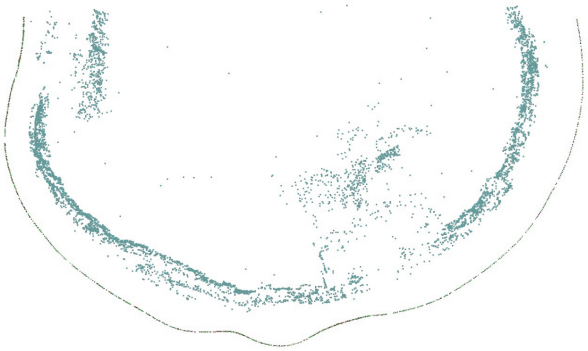
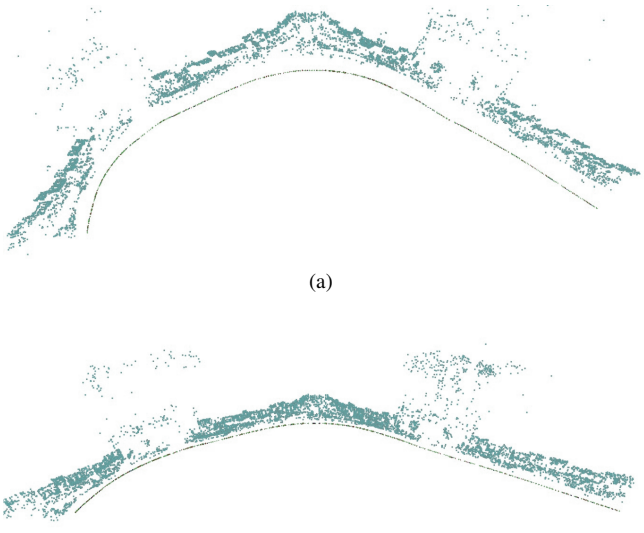


Figure 5: Initial sparse reconstruction (top view) with estimated focal length (before bundle adjustment) for the 'Baity Hill' video sequence, showing triangulated 3D points along with the recovered camera poses.



(a)

(b)

Figure 6: Two views of the sparse reconstruction of the 'Baity Hill' sequence after bundle adjustment to optimize intrinsic camera parameters as well as non-linear radial-distortion. Note from the top-view in (a) that the reconstruction is much closer to ground-truth as compared to that from Figure 5.

(a)

(b)

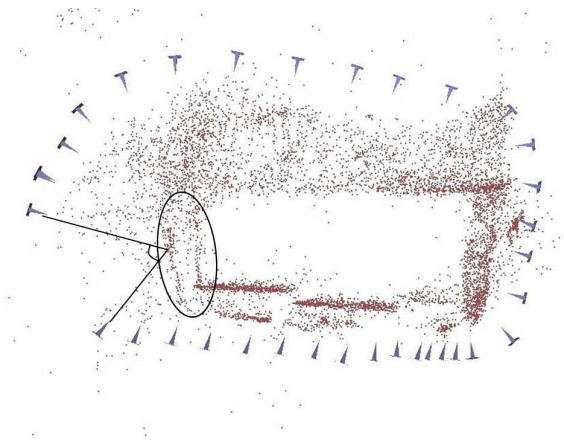
Figure 7: Final textured 3D model obtained from the 'Baity Hill' sequence.

In order to demonstrate loop-closing, we show results from the 'South Building' sequence, which consists of 1024x768 images captured using a Point Grey Research Ladybug camera. Some frames from the video are shown in Figure 8. Note that the first and last images in Figure 8 represent frames at either end of the video sequence, forming a loop around the building. The dimensions of the building are approximately 37x16 meters and the camera's path completes the loop by crossing at an angle of approximately 90 degrees. Matching correspondences across this wide angle using previous techniques for wide baseline matching is difficult or even impossible. However, using VIP patches we were able to complete the loop, generating an accurate 3D point model of the building. Figure 9(a) shows the top view of an initial sparse reconstruction of the 'South Building' sequence. Due to the accumulated drift in the pose estimation process, the obtained sparse reconstruction does not succeed in correctly recovering all facades of the building (note the area circled in the figure). However, following the loop completion procedure described previously, it can be seen that we obtain a much more accurate reconstruction (refer Figure 9(b)).

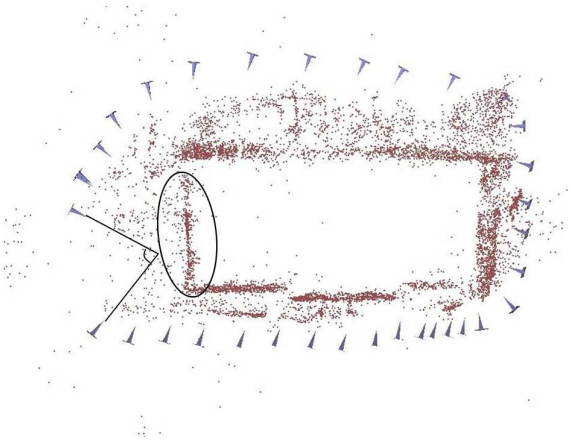
5 CONCLUSIONS AND FUTURE WORK

This paper presents a system for 3D reconstruction of architectural scenes from uncontrolled video sources. The reconstructions can be viewed from novel viewpoints or used to take relative measurements. Given an object of known scale in the scene we can also give absolute measurements. Our system is capable of generating models on the scale of single buildings. Our group continues to work on methods to scale our algorithms to work an much larger scenes. Currently, bundle adjusting very large scenes forms the main bottle-neck. In fact, given a set of camera poses we can calculate the scene geometry in real time. We are optimistic that this bottle-neck will be reduced in the future.

Figure 8: Representative frames from the 'South Building' video sequence.



(a)



(b)

Figure 9: Top view of the sparse reconstruction of the 'South Building' sequence.

ACKNOWLEDGEMENTS

This work has been supported by NSF Career Award No. 0237533 and The David and Lucille Packard Foundation.

REFERENCES

Chum, O. and Matas, J., 2005. Matching with PROSAC - progressive sample consensus. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Davis, T. A., Gilbert, J. R., Larimore, S. and Ng, E., 2004. A column approximate minimum degree ordering algorithm. ACM Transactions on Mathematical Software 30(3), pp. 353–376.

Dellaert, F. and Kaess, M., 2006. Square root SAM: Simultaneous location and mapping via square root information smoothing. International Journal of Robotics Research.

Engels, C., Stewénius, H. and Nistér, D., 2006. Bundle adjustment rules. In: Photogrammetric Computer Vision (PCV).

Faugeras, O. D., Luong, Q.-T., and Maybank, S. J., 1992. Camera self-calibration: Theory and experiments. In: European Conference on Computer Vision (ECCV).

Fraundorfer, F., Stewenius, H. and Nister, D., 2007. A binning scheme for fast hard drive based image search. Computer Vision and Pattern Recognition, IEEE Computer Society Conference on 0, pp. 1–6.

Fung, J., Mann, S. and Aimone, C., 2005. OpenVIDIA: Parallel GPU computer vision. In: Proceedings of the ACM Multimedia 2005, pp. 849–852.

Haralick, R., Lee, C., Ottenberg, K. and Nollei, M., 1994. Review and analysis of solutions of the three point perspective pose estimation problem. Int. Journal of Computer Vision 13, pp. 331–356.

Hill, R., Fung, J. and Mann, S., 2004. Reality window manager: A user interface for mediated reality. In: Proceedings of the 2004 IEEE International Conference on Image Processing (ICIP2004), pp. 24–27.

Kaess, M., Ranganathan, A. and Dellaert, F., 2008. iSAM: Incremental smoothing and mapping. IEEE Transactions on Robotics.

Kim, S. J., Gallup, D., Frahm, J.-M., Akbarzadeh, A., Yang, Q., Yang, R., Nister, D. and Pollefeys, M., 2007. Gain adaptive real-time stereo streaming. In: Proc. Int. Conf. on Computer Vision Systems (ICVS).

Li, X., Wu, C., Zach, C., Lazebnik, S. and Frahm, J.-M., 2008. Modeling and recognition of landmark image collections using iconic scene graphs. In: European Conference on Computer Vision (ECCV).

Lourakis, M. and Argyros, A., 2004. The design and implementation of a generic sparse bundle adjustment software package based on the Levenberg-Marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH.

Lowe, D. G., 1999. Object recognition from local scale-invariant features. In: ICCV, pp. 1150–1157.

Lucas, B. and Kanade, T., 1981. An iterative image registration technique with an application to stereo vision. In: International Joint Conference on Artificial Intelligence (IJCAI), pp. 674–679.

Matas, J. and Chum, O., 2005. Randomized RANSAC with sequential probability ratio test. In: IEEE International Conference on Computer Vision (ICCV).

Matas, J., Chum, O., Urban, M. and Pajdla, T., 2002. Robust wide baseline stereo from maximally stable extremal regions. In: British Machine Vision Conference, pp. 384–393.

Mendonca, P. R. S. and Cipolla, R., 1999. A simple technique for self-calibration. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Merrell, P., Akbarzadeh, A., Wang, L., Mordohai, P., Frahm, J.-M., Yang, R., Nister, D. and Pollefeys, M., 2007. Fast visibility-based fusion of depth maps. In: IEEE International Conference on Computer Vision (ICCV).

Ni, K., Steedly, D. and Dellaert, F., 2007. Out-of-core bundle adjustment for large-scale 3D reconstruction. In: IEEE International Conference on Computer Vision (ICCV).

Nister, D., 2003. Preemptive RANSAC for live structure and motion estimation. In: IEEE International Conference on Computer Vision (ICCV).

Nister, D., 2004. An efficient solution to the five-point relative pose problem. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 26, pp. 756777.

Nister, D. and Stewenius, H., 2006. Scalable recognition with a vocabulary tree. In: In CVPR, pp. 2161–2168.

Pollefeys, M. and Gool, L. V., 1999. Stratified self-calibration with the modulus constraint. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 21, pp. 707–724.

Pollefeys, M., Koch, R. and Gool, L. V., 1999. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. Int. Journal of Computer Vision 32, pp. 7–25.

Pollefeys, M., Nister, D., Frahm, J.-M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S.-J., Merrell, P., Salmi, C., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewenius, H., Yang, R., Welch, G. and Towles, H., 2008. Detailed real-time urban 3d reconstruction from video. Int. Journal of Computer Vision.

Raguram, R., Frahm, J.-M. and Pollefeys, M., 2008. A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus. In: European Conference on Computer Vision (ECCV).

Shi, J. and Tomasi, C., 1994. Good features to track. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Snavely, N., Seitz, S. M. and Szeliski, R., 2008. Modeling the world from internet photo collections. Int. Journal of Computer Vision 80, pp. 189–210.

Triggs, B., 1997. Autocalibration and the absolute quadric. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Wu, C., Clipp, B., Li, X., Frahm, J.-M. and Pollefeys, M., 2008. 3d model matching with viewpoint invariant patches (vips). In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Zach, C., Gallup, D. and Frahm, J., 2008. Fast gain-adaptive KLT tracking on the GPU. In: CVPR Workshop on Visual Computer Vision on GPU's (CVGPU).

Ziegler, G., Tevs, A., Theobalt, C. and Seidel, H.-P., 2006. On-the-fly point clouds through histogram pyramids. In: 11th International Fall Workshop on Vision, Modeling and Visualization 2006 (VMV2006), pp. 137–144.