2005

Efficiently Registering Video into Panoramic Mosaics

Drew Steedly
2University of Massachusetts Amherst

Efficiently Registering Video into Panoramic Mosaics

Drew Steedly[1] Chris Pal[2] Richard Szeliski[1]

[1]Microsoft Research
Redmond, WA 98052
{steedly, szeliski}@microsoft.com

[2]University of Massachusetts, Amherst
Amherst, MA 01003
pal@cs.umass.edu

Abstract

We present an automatic and efficient method to register and stitch thousands of video frames into a large panoramic mosaic. Our method preserves the robustness and accuracy of image stitchers that match all pairs of images while utilizing the ordering information provided by video. We reduce the cost of searching for matches between video frames by adaptively identifying key frames based on the amount of image-to-image overlap. Key frames are matched to all other key frames, but intermediate video frames are only matched to temporally neighboring key frames and intermediate frames. Image orientations can be estimated from this sparse set of matches in time quadratic to cubic in the number of key frames but only linear in the number of intermediate frames. Additionally, the matches between pairs of images are compressed by replacing measurements within small windows in the image with a single representative measurement. We show that this approach substantially reduces the time required to estimate the image orientations with minimal loss of accuracy. Finally, we demonstrate both the efficiency and quality of our results by registering several long video sequences.

1 Introduction

The automatic construction of large, high-quality panoramas from regular hand-held photographs is one of the recent success stories of computer vision, with stitching software bundled with many digital cameras and photo editors. However, approaches for automatic high-quality registration and stitching of video sequences have so far been hindered by high computational costs and thus simplifications in motion models or restrictive assumptions are typically required to make such algorithms run in reasonable time. Registering *all* the frames of a video into a panoramic mosaic [24, 16] (as opposed to just select frames of the video) enables many new creative possibilities. For example, many applications, such as high dynamic range imaging [4, 11] and video textures [22], can be extended to panoramas [1]. Furthermore, registering video frames can facilitate applications such as indexing [7], image stabilization [3], and compression [8].

While existing methods [2, 26] for constructing large panoramas in a "batch" fashion from static images can be quite robust, they are typically not efficient for aligning and stitching all the frames of a high-quality video sequence. While fast techniques do exist for stitching video [17], such methods typically use more restricted motion models and produce final panoramic representations that are less accurate than static image-based batch processing approaches.

In this paper, we present a new automatic method that is much more computationally efficient than existing batch approaches. We demonstrate that we can gain this efficiency without sacrificing registration quality. We use our method to register a number of test video sequences on the order of a thousand frames. For n frames and k key frames, the complexity of matching in our approach is $\mathcal{O}(n + k^2)$ and the complexity of the bundle adjustment is $\mathcal{O}(n+k^3)$. Thus, for many video registration applications, our approach scales well for a thousand frames and beyond. As a result, we can register very long video sequences into panoramic representations and enable many new applications and representations of video.

1.1 Motivating Approaches

Recently, Brown and Lowe demonstrated how panoramas contained in a set of unordered images can be automatically identified, registered and stitched [2]. This approach allows a user to take a number of images with a still camera, automatically identify clusters of images that came from the same underlying panoramic scene, then stitch each cluster into a panorama. Their general approach for panorama recognition can be summarized as follows: First, an "interest point" detector is applied to the image (see [21] for a review of interest point detection methods). Then, invariant features such as those based on Lowe's Scale Invariant Feature Transform (SIFT) [13] are extracted. For each feature, the k nearest neighbors are matched and a connected component approach is used to identify separate panoramas. The images associated with each cluster are then aligned

using bundle adjustment [28]. Finally, the images in each cluster are warped and blended onto a compositing surface [25].

A number of approaches have been proposed for registering and stitching panoramas from video [15, 17, 24]. The way in which such mosaics are constructed, the underlying camera motion models employed, and the details of the algorithms vary considerably. In [9], an affine motion model is used for an image resolution enhancement application. In [15], an 8 parameter perspective transformation model is used. In [17], a simple translation only camera motion model is employed and a "manifold projection" approach is taken. This approach results in a fast algorithm for video stitching in which narrow strips of pixels from the the underlying scene are used to form the composite panoramic image. The approach thus avoids computing the more complex 3D camera motion [10].

With a still camera, users typically only take up to a few dozen images to create a panorama. However, with a video camera, it is easy to generate thousands of images each minute. This causes difficulties when directly applying the techniques from [2] to video sequences. While invariant feature detectors are fairly immune to image warps, in practice they match best to features from images that are not warped with respect to each other. This means the k nearest neighbors of a feature tend to come from the k images with the most overlap.

When estimating the image warps, matches between images with small overlap provide much stronger constraints than matches between images with large overlaps. This results in systematically less accurate image registrations. For example, if the camera pans across the scene in swaths, nearest neighbor matching may result in many matches between temporal neighbors but very few, if any, matches between swaths. The matches between swaths are needed to prevent drift due to error accumulation. These matches can potentially be found by performing $\mathcal{O}(n^2)$ matching between all image pairs, but this is impractical for our goal of creating panoramas from all the frames of a video.

Video sequences are not just unordered sets of images, since adjacent video frames typically have significant overlap. In [19, 20], video stitching is performed by initially only stitching together adjacent frames of the video sequence, thereby making the matching problem linear in the number of images. This ignores matches due to the camera crossing back over its path. By not including these matches, components of the panorama can drift due to error accumulation. It is possible to compensate for this somewhat by interleaving the matching process and alignment process as in [19, 20]. After each new image is aligned to its temporal neighbor, spatial neighbors can be identified and used to refine the orientation estimate of the new image.

In contrast to these approaches, we present an efficient method for registering all frames of a video sequence, which is generally applicable and has been implemented for a number of camera motion models, such as 2D similarities, 3D rotations, affine warps and full homographies. (See [6] for a review of these and other models.) The approach we present here has been used to register on the order of a thousand frames from a video sequence.

In the following sections we present a number of novel contributions we used to construct a working system addressing these video stitching goals. In Section 2, we describe our method for using the ordering of frames in a video sequence to reduce the number of image pair comparisons. This speeds up the matching phase. In Section 3, we represent the matches between a pair of images using a few representative matches, which reduces the time required to align the images. In Section 5 we present efficiency and quality results of our method for aligning video frames and stitching panoramas from video.

2 Efficient Match Structures for Video

In order to avoid matching all pairs of frames, some assumption about the correlation between temporal and spatial adjacency must be made. Ideally, we would like to only search for matches between images that actually overlap spatially. For example, if the images were taken from a pan-tilt-zoom camera or we knew that the capture had been done in a raster-scan manner, we would know approximately where the images were taken relative to each other and would be able to predict which frames overlap. Strong assumptions about the correlation allow the search for matches to be more restrictive but also make the automatic stitching process more brittle. We want to allow as much freedom as possible when capturing panoramas, so these are not acceptable options.

By interleaving the image matching and orientation estimation steps, [19, 20] make the fairly weak assumption that temporally adjacent images are spatially adjacent. They also make the assumption that any loops in the camera path are small enough that the accumulated error, or drift, can be ignored. Even these assumptions are more restrictive than we would like. When filming a 360 degree panorama with a long focal length, the two ends of the panorama are especially susceptible to error accumulation. Also, the position of the endpoints is strongly affected by misestimation of the focal length [12]. This framework does not directly handle breaks in the matching, as would occur with multiple videos of the same panorama. Lastly, interleaving the matching and alignment requires that the images be aligned in the same order as the video. In our framework, we are able to defer aligning the images until after the matching is done, thereby allowing us to align images with more matches before images with fewer matches.

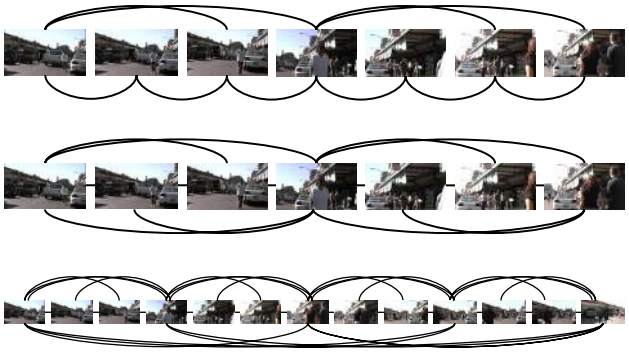

Figure 1: (top) The initial matching structure: video frames are matched to their sequential neighbors and to the last key frame. (middle) As new key frames are detected, forward matches are added from intermediate frames to the new key frame. (bottom) Key frames are densely matched, creating the final key frame mesh.

So as to maintain the robustness of [2], we only assume that *most* temporally adjacent frames are spatially adjacent. We start by looking for matches between all pairs of temporally adjacent images. We use the local matches to select key frames based on the amount of overlap. To do this, our algorithm starts by defining the first and last frame to be key frames. Our algorithm then steps though the video labeling frames as key frames if they do not overlap the most recently labeled key frame by some threshold (25% to 50% for results shown here). The intermediate frames not labeled as key frames are matched to their temporally neighboring key frames and intermediate frames. A key frame mesh is then created by finding matches among pairs of key frames.

The assumption underlying our approach is that neighboring key frames almost completely cover the panorama. In general, though, there will be portions of the panorama that are not covered by any key frames and that will be left out of the matching. As the overlap threshold is set tighter, these potentially unmatched areas become smaller. On the other hand, if the camera path travels over the panorama many times, several key frames can be associated with the same portion of the panorama. This will cause more matches to be tried when generating the key frame mesh than if the camera had panned over the panorama once. Therefore the user pays a computational cost penalty for covering the panorama more than once, although the same can be said for taking more still images than needed to cover the panorama in [2]. The ideas from [19] could be used to prune out key frames that overlap other key frames, but we simply left the extra key frames in.

Our algorithm for video alignment is outlined in Figure 2. We extract Multi-Scale Oriented Patches (MOPs)

Video Alignment Algorithm:

1. For each frame:
   - Extract invariant features at interest points
   - Match to previous frame
   - Match to previous key frame
   - Estimate overlap with previous key frame
   - Mark as a key frame if overlap is too low
2. For each frame:
   - Match to the *next* or "forward" key frame
3. For each key frame:
   - Match to all other key frames
4. Compress match measurements as per Section 3
5. Estimate image orientations from compressed matches using bundle adjustment

Figure 2: Our algorithm for efficiently constructing a sparse match structure, compressing the matches and aligning video frames.

[14] from each frame. RANSAC [5] is used to determine sets of inlier and outlier features based on their geometric compatibility with a pairwise homography as in [2]. The top row of Figure 1 illustrates matching features in successive frames back to the previous intermediate frame and key frame. The middle row shows the additional forward matches from each frame to the *next* key frame and moves the local match arcs for clarity. Finally, the bottom row shows all the matches after the key frame mesh is added in. Importantly, we have found that each of these matching steps are essential to achieve a globally robust registration. In the next section, we discuss efficient ways to estimate the camera orientations from this set of image matches.

3 Match Compression

Once feature matches have been established between pairs of images, they can be used to estimate the camera orientations. Each feature match defines a measurement error that depends on the relative orientations of the pair of images. The relative camera orientations are estimated by minimizing the measurement error in a (robustified) least-squares framework. An interest point detector will typically extract several hundred features from each image, resulting in hundreds of matches between image pairs with large overlaps. If the matched features are well distributed across the im-

age, this strongly constrains the relative orientations of the images. The large number of measurements is a computational bottleneck for many sequences.

We address this by replacing the measurements between a pair of images with a much smaller number of representative measurements. A similar idea was used in [27], where hallucinated point measurements were added as a simple way to incorporate planarity knowledge in structure from motion. In contrast, we want to *reduce* the number of point measurements, so we replace measurements with a smaller number of hallucinated measurements. An example of some original measurements and the replacement measurements is shown in Figure 3. Both the original and new measurements are represented by pairs of image points and 2×2 covariance matrices. The new measurement is set to the centroid of the original points and the covariance is adjusted by summing the inverse covariance matrices. Our match compression approach is thus related to the image patch based method used in [23] where points associated with low texture regions are down weighted or excluded.

By replacing the original measurements with a smaller number of representative measurements, we are changing the shape of the error surface. Therefore, we are paying an accuracy penalty for the reduced computational cost. To minimize the accuracy loss, we merge measurements which span a small portion of the image. Measurements that span a small angular window poorly constrain the parameters of a homography other than the translational component. Representing a group of points by their centroid discards the constraints they provide on the non-translational components of the warp. Only allowing points to merge if they have a small extent reduces the amount of information we are throwing out.

We select measurements to merge by placing a bound on the window size they span in either image. Starting with all the measurements in one group, we recursively split along the largest axis-aligned dimension (in either image) until the bound is met in both images for all groups. For each group, a single measurement is created in each image positioned at the centroid of the group. More sophisticated clustering techniques could potentially satisfy the bound constraint with fewer clusters, but our greedy strategy performs well and takes an insignificant amount of overhead time to compute.

We also considered other ways to reduce the number of measurements considered at each iteration. For instance, each matched pair of images could be aligned and a linear approximation to the error could be used to create a single, linear measurement connecting the two images. There are two benefits to using a few representative 2D points, though. First, we are able to preserve some of the non-linearity of the error surface and do not need to have the images aligned before simplifying our representation. Second, much effort has been put into speeding up the sparse non-linear minimization code to estimate the camera poses [28]. Using our approach, the optimization code does not need to be modified. It is simply a matter of compressing the measurements between the matching and alignment steps.

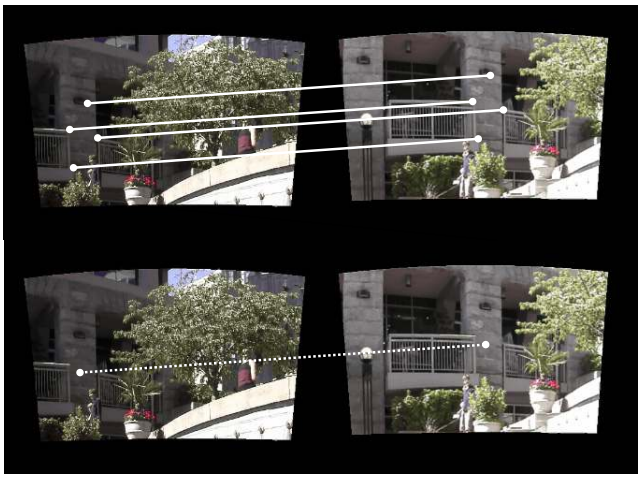

Figure 3: (top) Groups of original measurements spanning a small angular window for matches are replaced by their centroid (bottom). In practice, window sizes that are much smaller relative to the frame size are used.

4 Computational Cost

Using our matching method and compressed measurements, camera orientations that minimize our objective function are estimated using a second order, non-linear technique such as Newton-Raphson [18]. For P image pairs with M_i measurements between pair i, the objective function being minimized is

$$\chi^2 = \sum_{i \in P} \sum_{j \in M_i} \mathbf{e}_{ij}^T \mathbf{\Sigma}_{ij}^{-1} \mathbf{e}_{ij}, \tag{1}$$

where $\mathbf{e}_{ij}$ is the 2D measurement error due to match j in pair i and $\mathbf{\Sigma}_{ij}$ is the 2×2 measurement covariance. $\mathbf{e}_{ij}$ depends on measurement ij and the relative orientations of the images in pair i. We use the following symmetric error function:

$$\mathbf{e}_{ij} = \mathbf{w}(\mathbf{x}_{ij}, \mathbf{p}_i) - \mathbf{w}^{-1}(\mathbf{x}'_{ij}, \mathbf{p}_i), \tag{2}$$

where $\mathbf{w}()$ is the warping function, $\mathbf{x}_{ij}$ and $\mathbf{x}'_{ij}$ are the points in each image being warped and $\mathbf{p}_i$ is the vector of warp parameters. Notice that $\mathbf{p}_i$ represents a "halfway" warp between the images in pair i. In the case of a rotational panorama with a single unknown focal length, this

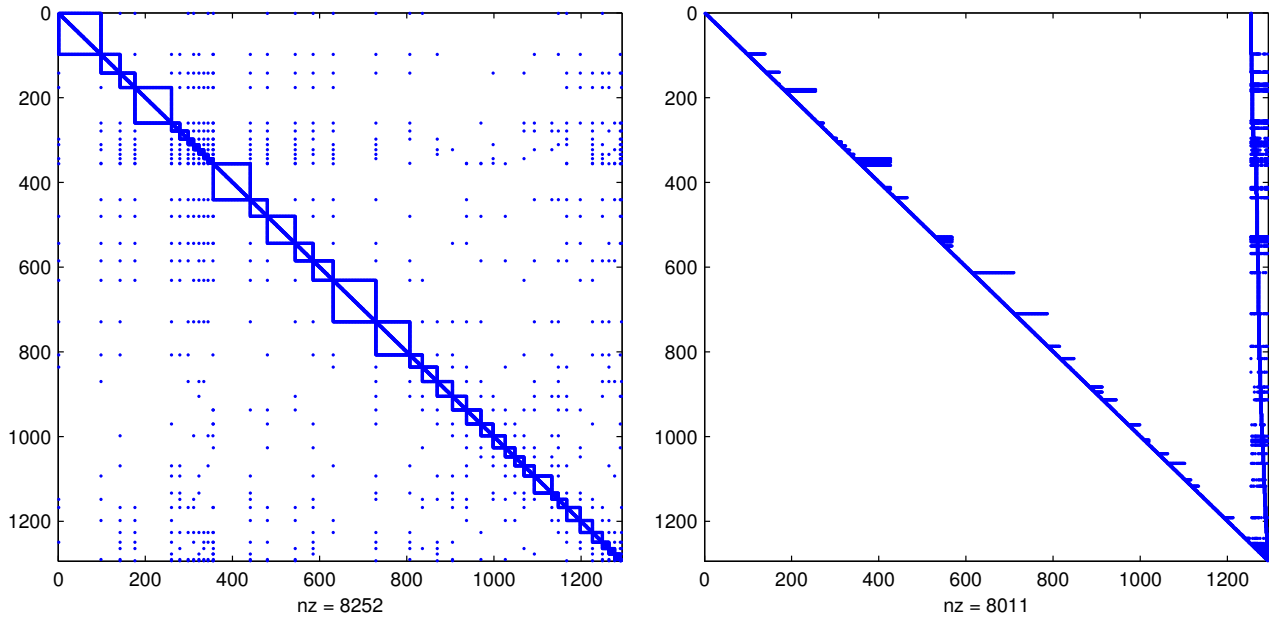

Figure 4: The left plot shows matched frames from the LK sequence and, equivalently, the sparsity pattern of the Hessian. Point (i, j) is filled in if there matches were found between image i and j. The temporal neighbor matches show up as a line down the diagonal. The forward and backward matching to key frames show up as boxes on the diagonal with longer camera pauses generating larger boxes. The key frame to key frame matches show up as sparse off-diagonal points. Since the camera is panning slowly in one direction in the Locks sequence, the key frame to key frame matches between temporally distant frames are outliers, which were rejected during the robust optimization. The right plot shows the sparsity pattern of the factored Hessian after permuting the key frames to the end. The key frames cause fill ins in the bottom right corner and up the right side.

means that the error is calculated on an image plane rotated halfway between the two images with the warp function given by

$$\mathbf{w}(\mathbf{x}_{ij}, [\boldsymbol{\omega}^T, f]^T) = \pi \left(\mathbf{K}(f)\mathbf{R}(\boldsymbol{\omega})\mathbf{K}^{-1}(f) \begin{bmatrix} \mathbf{x}_{ij} \\ 1 \end{bmatrix} \right), \tag{3}$$

where $\pi([x, y, z]^T) = [x/z, y/z]^T$, $\mathbf{R}(\boldsymbol{\omega})$ is the halfway rotation matrix and $\mathbf{K}(f)$ is the calibration matrix

$$\mathbf{K}(f) = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Homographies, 2D similarities and rotation only motion models result in different forms of $\mathbf{w}()$.

At each iteration of the minimization, the gradient and Hessian of the objective function, $\mathbf{g}$ and $\mathbf{H}$ respectively, are calculated and used to solve for an update step. Using a linear approximation for $\mathbf{e}_{ij}$, $\mathbf{g}$ and $\mathbf{H}$ are given by

$$\mathbf{g} = \frac{\partial \chi^2}{\partial \boldsymbol{\theta}} \quad \text{and} \quad \mathbf{H} = \frac{\partial^2 \chi^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T}, \tag{5}$$

and the update step, $\boldsymbol{\delta}$, is obtained by solving

$$\mathbf{H}\boldsymbol{\delta} = -\mathbf{g}. \tag{6}$$

The Hessian is a sparse $Nd \times Nd$ matrix, where N is the number of cameras and d is the number of parameters used to represent each camera (e.g., $d = 3$ for a rotating camera). The computational cost of calculating $\mathbf{H}$ and $\mathbf{g}$ is linear in the number of measurements since each measurement term in (1) only depends on a single pair of images.

For panoramas in which there are measurements between all pairs of images, solving (6) is cubic in N. When there are only connections between a small subset of all image pairs, the sparsity pattern of $\mathbf{H}$ is similar to the left plot of Figure 4. As documented in [28], we can take advantage of the sparsity of $\mathbf{H}$ to reduce the computational complexity of solving (6) using an LU decomposition to only cubic or less in the number of *key frames* and linear in the number of intermediate frames. By permuting the parameters of the key frames to the end, we can limit the number of non-zero entries, or fill ins, when factoring $\mathbf{H}$. The sparsity pattern of the factored Hessian from the Locks sequence is shown on the right side of Figure 4.

The number of matching image pairs is at most quadratic in the number of key frames (if they all overlap each other), but more typically linear; it is also linear in the number of intermediate frames. The number of feature matches per

Figure 5: Panoramic images generated from sequences LK and WF (top to bottom).

image pair is a scalar multiplier on this portion of the computational cost. In many cases, the cost of summing the contributions of the measurements dominates the computational cost of solving the sparse linear system in the motion parameters. Reducing the number of per-image measurements in such cases using feature match compression results in substantial computational savings at little cost in accuracy, as we demonstrate in the next section.

5 Results

To illustrate the quality of our registrations we have stitched together and blended *all* the frames of several video sequences using the techniques presented in this paper. The characteristics of the video sequences are summarized in Table 5. All of the sequences were captured with a 1280×720 video camera.

The final bundle adjustment results we present here used a 3D camera rotation motion models with unknown but fixed focal length, but our approach and system can handle a variety of motion models such as those described in [25]. The input images were then warped onto a spherical surface and composited using a feathered blend for display purposes. Sequences AC1, AC5, AC6, AC8, LK and WF were captured in one swath. Sequences AGP2, GP1, GP4 and GP5 were captured with multiple swaths. The sequences with a single swath were reconstructed with a key frame overlap threshold of 50%. For sequences with multiple swaths, the swath to swath correspondences were often missed with an overlap threshold of 50%. By tightening the overlap threshold to 25%, all of the swaths were connected. In sequence AGP2, there are a few images where only the very top of the cathedral is visible. Ghosting due to their misregistration is visible in the output composite

Seq	Width	Height	# Frames	f	# Swaths
AC1	2039	2399	568	2.27	1
AC5	1357	2050	862	3.94	1
AC6	1319	1883	183	2.34	1
AC8	1372	2998	568	2.30	1
AGP2	2052	2393	1023	2.82	3
GP1	3961	2495	620	3.30	5
GP4	4096	3494	723	6.34	15
GP5	3051	2284	621	3.94	3
WF	2650	1371	1147	2.98	1
LK	4096	1029	1563	2.10	1

Table 1: Summary of the characteristics of output panoramic mosaics and input video used for our tests.

image. Sequence GP4 consisted of 15 horizontal swaths taken with the camera zoomed in ($f = 6.34$). The noticable blurring at the bottom of GP4 is due to motion blur in the original video. A few inter-swath matches were dropped during bundle adjustment at the left and right side of the building, leading to some ghosting on the edges.

We evaluated the accuracy of merging measurements and approximating them with a single point by compressing measurements with a range of bounding box sizes. The results of sweeping the maximum bounding box size from 0% (no compression) to 50% (up to one point in each quadrant of the image) are given in Figure 8. We ran the sweep on three of the sequences that could be optimized in memory and not have to swap to disk. The amount of time taken to optimize the compressed matches is plotted on the left. The timings are normalized by the uncompressed time so that the relative speedups from each sequence can be seen. We evaluated the accuracy penalty by calculating the RMS error of the original, uncompressed measurements using the image orientations estimated from the compressed measurements.

A window size of 20% seems to be a sweet spot where performance increase has leveled off and the error has not started to rise. This corresponds to about 25 matches between fully overlapping images and 12 matches between images that overlap by 50%. For the camera motion model used here, this is probably a good number to choose. However, the window size might need to be smaller for warps such as homographies where the minimum number of matches is larger.

6 Summary and Conclusions

We have presented an efficient method for stitching long video sequences into high-quality panoramic mosaics. The approach presented here was used to achieve the large scale, high quality individual frame registrations required for con-

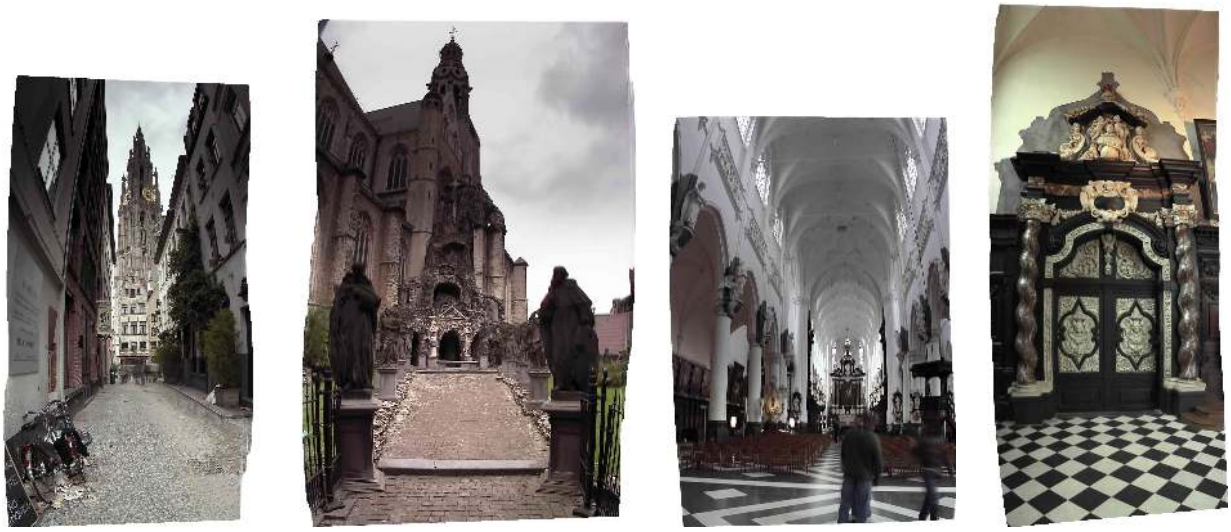

Figure 6: Panoramic images generated from sequences AC1, AC5, AC6 and AC8 (left to right)

Figure 7: Panoramic images generated from sequences AGP2, GP1, GP4 and GP5 (left to right)

structing the panoramic video textures described in [1]. As future work, our approach could be extended to more sophisticated key frame selection methods to ensure optimal scene coverage during more complex camera panning situations. Finally, we believe there are many other promising avenues of exploration enabled by our technique ranging from application in computer graphics, compression and image enhancement.

Acknowledgements We thank Simon Winder and Matt Uyttendaele for assistance with the MOPS matcher and other system integration advice. We also thank Matt for the European city and church video and Aseem Agarwala, Ke Colin Zheng and Michael Cohen for the waterfall and locks video.

References

[1] A. Agarwala et al. Panoramic video textures. *ACM Transactions on Graphics*, 24(3), *Proceedings of SIGGRAPH*, August 2005.

[2] M. Brown and D. Lowe. Recognizing panoramas. In *Ninth International Conference on Computer Vision (ICCV'03)*, pages 1218–1225, Nice, France, October 2003.

[3] P. J. Burt and P. Anandan. Image stabilization by registration to a reference mosaic. *In DARPA Image Understanding Workshop, Monterrey, November*, pages 457–465, 1994.

[4] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. *Proceedings of SIGGRAPH 97*, pages 369–378, August 1997.

[5] M. A. Fischler and R. C. 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, June 1981.

[6] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.

[7] M. Irani and P. Anandan. Video indexing based on mosaic representations. *Proc. of the IEEE*, 86(5):905–921, May 1998.

[8] M. Irani, S. Hsu, and P. Anandan. Video compression using mosaic representations. *Signal Processing: Image Communication, special issue on Coding Techniques for Low Bitrate Video*, 7(4–6):529–552, November 1995.

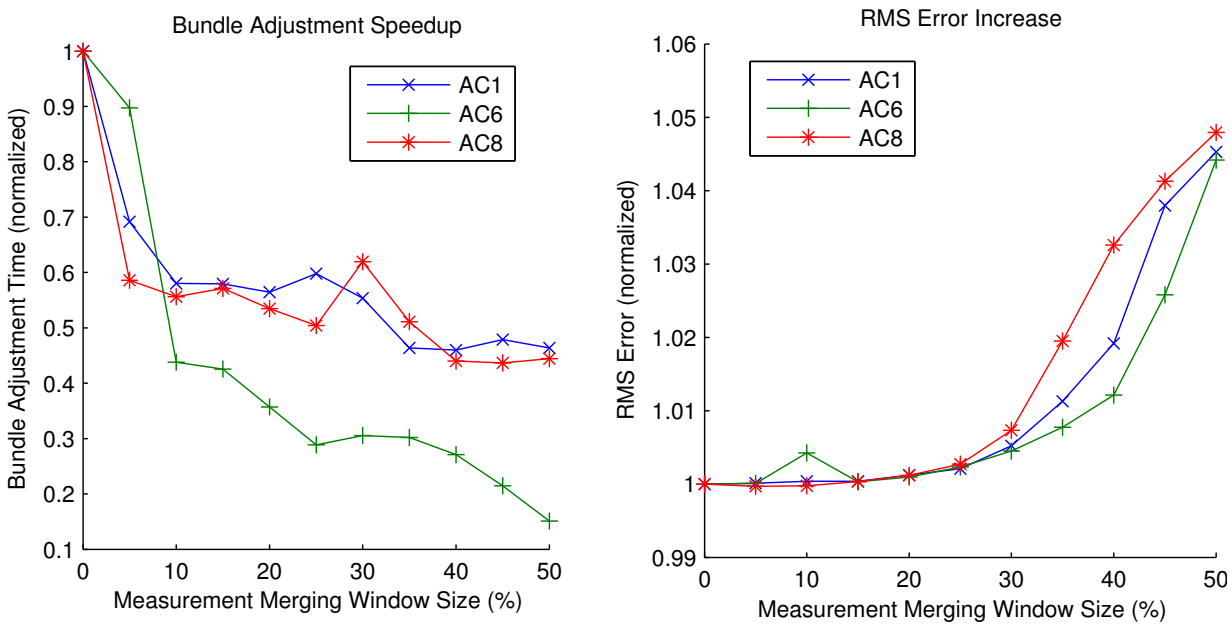

Figure 8: The time (left) taken by the bundle adjuster and RMS error (right) for various maximum window sizes as a percentage of the image size. 50% allows points to be merged if their bounding box width and height are less than half the maximum dimension of the image. 0% does not allow any points to be merged. Each plot has been normalized by the value from optimizing without compression. The uncompressed bundle adjustment times for the sequence AC1, AC6 and AC8 were 58s, 12s and 69s respectively. The uncompressed errors were 1.7, 1.6 and 1.7 pixels respectively.

[9] M. Irani and S. Peleg. Improving resolution by image registration. *Graphical Models and Image Processing*, 53(3):231–239, May 1991.

[10] M. Irani, B. Rousso, and S. Peleg. Recovery of ego-motion using image stabilization. *In IEEE Conference on Computer Vision and Pattern Recognition*, pages 454–460, June 1994.

[11] S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High dynamic range video. *ACM Trans. Graph.*, 22(3):319–325, 2003.

[12] S. B. Kang and R. Weiss. Characterization of errors in compositing panoramic images. *Computer Vision and Image Understanding*, 73(2):269–280, February 1999.

[13] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[14] R. S. M. Brown and S. Winder. Multi-image matching using multi-scale oriented patches. *To appear International Conference on Computer Vision and Pattern Recognition*, 2005.

[15] S. Mann and R. W. Picard. Virtual bellows: Constructing high-quality images from video. *In First IEEE International Conference on Image Processing (ICIP-94),Austin, TX*, pages 363–367, 1994.

[16] M. Massey and W. Bender. Salient stills: Process and practice. *IBM Systems Journal*, 35(3&4):557–573, 1996.

[17] S. Peleg and J. Herman. Panoramic mosaics by manifold projection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pages 338–343, San Juan, Puerto Rico, June 1997.

[18] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, England, 1988.

[19] H. S. Sawhney et al. Videobrush: Experiences with consumer video mosaicing. In *IEEE Workshop on Applications of Computer Vision (WACV'98)*, pages 56–62, 1998.

[20] H. S. Sawhney, S. Hsu, and R. Kumar. Robust video mosaicing through topology inference and local to global alignment. In *Fifth European Conference on Computer Vision (ECCV'98)*, pages 103–119, Freiburg, Germany, June 1998.

[21] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, June 2000.

[22] A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa. Video textures. In *In Proc. SIGGRAPH*, pages 489–498, 2000.

[23] H.-Y. Shum and R. Szeliski. Construction of panoramic mosaics with global and local alignment. *International Journal of Computer Vision*, 36(2):101–130, February 2000. Erratum published July 2002, 48(2):151-152.

[24] R. Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, 16(2):22–30, 1996.

[25] R. Szeliski. Image alignment and stitching: A tutorial. Technical Report MSR-TR-2004-92, Microsoft Research, December 2004.

[26] R. Szeliski and H. Shum. Creating full view panoramic image mosaics and environment maps. *In Proc. of SIGGRAPH*, pages 251–258, 1997.

[27] R. Szeliski and P. Torr. Geometrically constrained structure from motion: Points on planes. In *European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE)*, pages 171–186, Freiburg, Germany, June 1998.

[28] B. Triggs et al. Bundle adjustment — a modern synthesis. In *International Workshop on Vision Algorithms*, pages 298–372, Kerkyra, Greece, September 1999. Springer.