Video Alignment to a Common Reference

Rahul Dutta, Bruce Draper, and J. Ross Beveridge

Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523 U.S.A.
{duttar}{draper}{ross}@cs.colostate.edu

Abstract

Handheld videos include unintentional motion (jitter) and often intentional motion (pan and/or zoom). Human viewers prefer to see jitter removed, creating a smoothly moving camera. For video analysis, in contrast, aligning to a fixed stable background is sometimes preferable. This paper presents an algorithm that removes both forms of motion using a novel and efficient way of tracking background points while ignoring moving foreground points. The approach is related to image mosaicing, but the result is a video rather than an enlarged still image. It is also related to multiple object tracking approaches, but simpler since moving objects need not be explicitly tracked. The algorithm presented takes as input a video and returns one or several stabilized videos. Videos are broken into parts when the algorithm detects the background changing and it becomes necessary to fix upon a new background. Our approach assumes the person holding the camera is standing in one place and that objects in motion do not dominate the image. Our algorithm performs better than several previously published approaches when compared on 1,401 handheld videos from the recently released Point-and-Shoot Face Recognition Challenge (PASC). The source code for this algorithm is being made available.

1. Introduction

Two kinds of motion tend to dominate in videos taken by a stationary person with a hand-held camera. First, there is intentional motion such as panning or zooming. Second, there is unintentional motion, in the form of unwanted shakes and jitters. Some of the best and most recent work on video stabilization seeks to remove shakes and jitters while smoothing the presumably intended motion. This is typically done by first identifying common features in the scene and estimating the frame-to-frame movement of these features. Then, algorithms do the following: 1) estimate the original camera motion, 2) fit a model to capture the smoothed presumably intended camera motion, 3) solve for transformation matrices for each frame warping them to form a new stabilized video without jitter and apparently smooth camera motion. An excellent example of recent work following this approach is by Grundmann et al. [6] where they demonstrate in the context of YouTube a robust algorithm for stabilizing videos so as to produce a final video that appears as though it was shot by a professional cinematographer.

There are circumstances, often associated with video analysis, where the goal is to stabilize a video with respect to its background and consequently remove apparent camera motion entirely. There are many reasons to remove camera motion. For example, stabilized videos enable background modeling [16] and make object motion more apparent to both to human and automated observers. Certainly there are domains where the goal of removing camera motion is not practical, for example cameras mounted on moving vehicles. In such domains, more complicated solutions involving multiple object tracking and camera modeling are required. However, there are many scenarios where the simpler approach of stabilizing relative to a fixed background is appropriate.

We have recently begun working with a data set where this is the case. In the Point-and-Shoot Face Recognition Challenge (PaSC) [15], handheld videos show people carrying out activities in a scene. The videos were shot by a person who is holding the camera while standing in one place, and frequently they pan the camera in order to keep the subject near the center the video. At the same time, the person in the video typically does not move so far in the world as to change completely the background. Under these conditions, it is reasonable and desirable to stabilize the videos with respect to the background. Results presented below for the

PaSC handheld video demonstrate our algorithm generally does a good job of removing camera motion.

Like much of the prior work on stabilization, our algorithm incorporates detection of salient features [6], Lucas and Kanade tracking [6, 11], RANSAC [6] for testing potential feature correspondences, and finally motion modeling in terms of frame-to-frame alignment transforms [6]. However, to the best of our knowledge, two aspects of our approach are novel. First, our algorithm introduces a staged multi-frame mechanism for introducing new salient features into the tracking procedure. Second, the same multi-frame mechanism makes decisions about whether a salient point is part of the stable background based upon measurements over multiple frames.

In particular, the transformation matrix that registers frame t to the background is computed using features originally detected in frame $t-2$. These features are tracked to frame $t-1$ using Lucas-Kanade and their motion is compared to the camera motion at $t-1$. Features that move consistently with the camera are potential background points are are forwarded to frame t, whereas features with inconsistent motion are presumed to be foreground features and are dropped. As a result, the transformation matrix at time t is estimated by applying RANSAC and motion modeling to a set of features that have already been selected as background points and have persisted for at least two time steps. Similarly, the transformation at time $t+1$ is calculated from features originally extracted at time $t-1$. This additional level of feature selection adds robustness to the procedure. We show below that this procedure minimizes registration errors better than previous techniques.

Another difference relative to most prior work follows directly from our goal of stabilizing relative to a fixed background. In our approach, the final step of generating the stabilized video establishes a common reference frame and maps all frames back to this reference. In cases where the background change significantly, i.e. by more than 50%, the original video is broken and then each part is stabilized to a different fixed reference. This process of mapping videos to a common reference frame is related to image mosiacing [3].

The rest of this paper is organized as follows. Section 2 reviews related work on video stabilization and image mosaicing. Section 3 presents our new video stabilization algorithm. Section 4 presents an empirical evaluation comparing our method to alternative methods based upon prior video stabilization and image mosaicing algorithms in the literature.

2. Related Work

Broadly speaking, the aim of Video Stabilization is to remove unwanted motion. For example, it is almost impossible for a person to hold a camera and not introduce small but rapid movements, i.e. jitters. Stabilization algorithms often estimate a smooth camera motion path from video frame data [11]. For example, Grundmann et al. [6] (Youtube) presented a robust method of finding an L1-optimal Camera path in order to generate stabilized videos. This algorithm is based on a Linear Programming framework that finds optimal partitions of the smooth camera path. Path modeling includes fitting portions of the path to constant, linear and parabolic motion models. A crop window of fixed aspect ratio is moved along that optimal path to include salient points or regions while minimizing an "L1 smoothness constraint". The goal is a stabilized video such as might have been shot by a professional cinematographer using expensive physically stabilized cameras. Such videos are pleasing for human viewers.

2.1. Overview - Common Aspects

Work on image alignment and image mosaicing is in some respects more closely related to our goal of creating stabilized backgrounds than is the recent work on video stabilization for smoothed camera motion trajectories. However, both bodies of work share common aspects. For example, a reliance upon extracting matchable features in successive frames. In particular, features like harris corners[3], SIFT features or Good features[12, 4] can be used for feature selection. Matching techniques like correlation of a window centered about the feature point[3], or tracking methods like Lucas Kanade Tracker[12] or Kalman Filters[4] are used to track features from frame to frame.

Frame-to-frame motion models also play a key role. Transformations with differing numbers of degrees of freedom (DOF) are common, including similarity transforms (4 DOF)[13], affine transforms (6DOF)[7] or homography transforms(8 DOF)[4, 3, 12]. These transformations are usually estimated from features matched between pairs of video frames. In practice, a balance must be struck between the ability to model more complex forms of motion versus the need to find more matching features in order to constrain additional degrees of freedom. Further, higher DOF motion models, in part due to their much larger space of possible solutions, are sometimes susceptible to settling on unrealistic frame-to-frame motions.

Another key problem is deciding which matched features to use. When a scene contains independently moving objects, then not all feature matches between frames are associated with the dominant motion. In particular, feature points on an independently moving object will, if included in the calculation of the dominant motion, throw off the estimate. Therefore, it is common to describe feature pairs as being either inliers or outliers relative to motion estimation. To separate outliers from inliers, either RANSAC[3] or LMeS(Least Median of Squares)[12] are often used.

2.2. Specific Prior Efforts

Real-time scene stabilization in video was demonstrated by Hansen et al.[7]. Their VFE-100 system employed a multi-resolution iterative scheme using Laplacian pyramid images. During every iteration, optical flow is estimated using cross correlation of the current image and the previous image at a particular pyramid level. A linear motion model is fit to the optical flow and the previous image is warped with that model. In the next iteration the flow is estimated between the warped previous image and the current image at a higher resolution level of the pyramid.

Morrimoto and Chellappa [13] describe a fast and robust implementation of a 2D electronic image stabilization system. The method selects features on the horizon by thresholding and dividing the Laplace of the image into vertical zones and selecting the topmost feature of every zone. The selected features are tracked from f_{t-1} to frame f_t by a multi-resolution scheme also involving Laplacian pyramids. The feature point at every pyramid level in frame f_t is searched over the window centered about the point in frame f_{t-1} and the point which returns the minimum SSD is the best match. The estimate obtained at a coarse level is used to search for the minimum SSD at a finer level of the Laplace pyramid. Finally the Least square solution is used to estimate the similarity matrix from the point correspondences. The motion matrices are combined from the reference frame to the current frame and the current frame is warped with the accumulated motion model.

A good example of an image mosiacing algorithm employing these techniques is that of Capel and Zisserman[3]. Their system used a window-based localized correlation score to match the harris corners between two consecutive frames. RANSAC was used to discard outlier points and to estimate the homography that best defined the matched inliers. Finally the estimates of point correspondences and the homography were refined using a non-linear optimizer by minimizing the euclidean distance between the original feature points and the corrected feature points of the correspondences. The cost function was minimized using the Levenberg-Marquardt algorithm.

Censi et al. [4] approach image mosaicing with feature tracking. Their algorithm tracks good features[17] in every subsequent frame of a sequence using a linear Kalman filter. The predicted position of the feature point is obtained from the predicted state of the Kalman filter and the neighborhood of this predicted position is searched for the minimum SSD (sum of square difference) error to find the corresponding feature point. The system uses a robust rejection rule x84[5] to identify outliers. The residual of every feature is calculated and any feature whose residual differs by more than 5.24 MAD from the median residual is discarded as an outlier.

There are excellent examples of image mosaics being used in wide area surveillance. Mei et al.[12] present a background retrieval system which detects Good Features in a reference frame and then tracks them over the subsequent frames using a Kanade-Lucas-Tomasi[17] tracker to obtain the frame-to-frame correspondences. The homography is estimated from these correspondences and a Least Median of Squares(LMeS) algorithm is used to remove the outliers. A mixture of Gaussians(MoG)[16] background modeling procedure is then used to separate a stable background from points-of-interest.

Heikkila et al. [8] also propose an automatic image mosaic method for wide area surveillance. Their method extracts SIFT features from the incoming images and then constructs a mosaic. RANSAC was used to reject the outliers and the parameters of the homography are refined by using the Levenberg-Marquardt algorithm for minimizing a geometric cost function defined over the inliers.

Another excellent and most recent example of Image Alignment is SIFT Flow of Liu et al.[9]. The system uses a pyramid based discrete flow estimation algorithm to match the SIFT descriptors between two images and defines a neighborhood for SIFT flow. A dual layer loopy belief propagation is used to minimize an objective function at every pyramid level, which constrains the SIFT descriptor to matched along the flow vector and also constrain the flow vectors of adjacent pixels to be similar. This system is primarily used to estimate correspondence between images of different scenes categories, however it has also been used for registration of same scene satellite images.

2.3. Alignment to a Common Reference

The alignment of frames in a video with respect to a reference frame will result in a video where the frames appear motionless except the borders where the content changes[2]. Two issues arise when aligning to a reference frame.

- Accumulating errors in motion estimation can lead to unacceptable errors relative to the reference frame.
- The amount by which a new frame overlaps the reference frame can grow too small. Man and Picard[10] solve this problem by splitting the frames into subsets which can be best registered.

Comparing algorithms that align video to a common reference involves both evaluating the breaks and also the quality of the aligned subsets of frames. Morimoto and Chellappa [14] propose a fidelity measure which corresponds to the Peak Signal to Noise Ratio (PSNR) between stabilized consecutive frames. The PSNR is the function of MSE (Mean squared error) which is the average departure per pixel from the desired stabilized result. Due to moving object in our videos, we propose in Section 4 a related measure less sensitive to differences due to moving objects.

3. Stabilization Algorithms

The algorithms described here take one video as input and return one or more videos, each of which is stabilized with respect to a stationary reference frame. In other words, the goal is for the background to remain fixed throughout each returned video. Input videos should only be broken when camera motion so alters the objects visible in the background that it is no longer possible to align with a common reference. The criteria for when to break a video is discussed further in Section 3.3.

Four algorithm variants are described here. Each carries out broadly the same three operations: 1) feature extraction and feature mapping between consecutive frames, 2) frame-to-frame motion estimation from the correspondences, and 3) motion compensation. These steps in general are described in Sections 3.1 and 3.2. Then, the algorithms themselves are developed in Sections 3.4 and 3.5.

3.1. Frame-to-Frame Motion Estimation

In general, camera motions are estimated from feature point correspondences. Features points are extracted, selected, and matched across frames, and then affine motions between consecutive frames are calculated from these tracked points. For frames f_t and f_{t-1} the affine transform may be expressed as:

$$\begin{bmatrix} x_t \\ y_t \\ 1 \end{bmatrix} = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ 1 \end{bmatrix}$$

This is a fairly standard choice of motion model. The 6 degrees of freedom (DOF) affine has 6 unknown parameters, and therefore 3 point correspondences generate six linear constraint equations. Thus, a match between any three features in one frame to the next typically defines an alignment transformation. We also experimented with 8 DOF homographies, but on the video data shown below we found the results were worse than when using the 6 DOF transform.

Also following common practice, the **RANSAC** algorithm is used to find an affine transformation supported by a majority of features. In every iteration of RANSAC, 3 point-wise correspondences are selected at random and then used to estimate the affine motion matrix. The estimated motion matrix is used to check how many points lie in the consensus set. If the percentage of points that fit the estimated affine model is more than a threshold we stop and declare the model as good. The points that fit this model are known as inliers and the rest are known as outliers. Once we have obtained the inliers from RANSAC we perform a linear least square solution to the inlier points to obtain the final affine matrix.

The RANSAC algorithm presumes point-wise correspondence between feature points in two frames. There are two ways of finding corresponding matched feature points used in the algorithms which follow. One is based upon feature similarity, in other words matches that measure similarity between feature points expressed in a feature space. For example, Heikkila et al. [8] used SIFT features to describe SIFT points. The other is to use a tracker to move points forward from frame $t-1$ to frame t. Mei et al. [12] use the KanadeLucasTomasi feature tracker (KLT) [18]. The algorithms below use an iterative pyramidal implementation of KLT based on optical flow to provide robustness to large displacements[1].

3.2. Motion Compensation

In our approach, one frame is taken as a reference frame and the incoming frames are registered to this reference frame. For example, if H_i represents the affine transform between frames i and $i-1$ and if the reference frame is H_0 , then the frame i can be mapped to the 1st frame by the composition of the transformations: $H_{1\cdots i} = H_1 H_2 H_3 \cdots H_i$ Given the motion model of frame i with respect to the reference frame, we then warp frame i using the inverse of $H_{1\cdots i}$. Warping a frame means projecting every pixel coordinate of the source image by the motion matrix to a new coordinate in the destination image.

To acquire an estimated pixel value in the destination image, sampling is done in reverse, going from destination to source. That is, for each pixel (x, y) of the destination image, the functions compute coordinates of the corresponding "donor", pixel in the source image and copy the pixel value: $dst(x,y) = src(f_x(x,y), f_y(x,y))$ where $f_x(x,y)$ and $f_y(x,y)$ are warping functions. Since $f_x(x,y)$ and $f_y(x,y)$ are seldom integers, bilinear interpolation is used to obtaining new interpolated pixel values. If $f_x(x,y)$, $f_y(x,y)$, or both, fall outside the source image, then $dst(x,y)$ is set to zero.

3.3. Criteria for Breaking Videos

Our algorithms will introduce breaks in response to two events. First, if an excessive amount of panning is detected such that the majority of a new frame is not visible in the reference frame, then a break is created. Such a break is triggered by monitoring the fraction of pixels in the current frame that lie outside the reference frame when mapped back to the reference frame using the accumulated motion model $H_{1\cdots i}$. In particular, our algorithms are typically set to break a video if more than 50% of the pixels in the current frame lie outside the reference frame.

The second trigger for a break is excessive scaling in the transformation matrix. This trigger makes sense in our domain because we are working with cameras that pan but do not zoom in or out during a single video. This trigger is implemented by monitoring the determinant of the accumulated motion model $H_{1\cdots i}$. If it drifts too far from 1.0,

i.e. it starts scaling the background relative to the reference frame, then a break is created. In general for our algorithms, if the determinant falls outside the range 0.95 to 1.05 then the video is broken.

We've observed that the determinant of the motion matrix lies very close to 1.0 for frame-to-frame estimates. Specifically, the determinant always remains within $1 \pm \epsilon$ where ϵ is on the order of 10^{-4}. However, we've also observed cases where errors accumulate in the accumulated motion model $H_{1 \ldots i}$, as previously discussed in Section 2.3, and starting over with a new reference becomes necessary.

Our tests have also revealed that using a Homography transform tends to cause more skew and perspective errors and testing for these errors in turn leads to more breaks. We initially tried stabilizing videos using the techniques in [8, 12], which used homography transforms. However, we found this led to a large number of video breaks, and subsequently changed the implementations of the baselines below to use affine models instead.

3.4. Baseline Algorithms

The two baseline algorithms are roughly modeled after the work by Heikkila et al. [8] and Mei et al. [12]. In **Baseline-1**, based upon Heikkila's approach, new SIFT Features are extracted for each new frame, and the motion model is determined by the best matching correspondence between SIFT features in adjacent frames. In **Baseline-2**, based upon Mei's approach, SIFT features are extracted from the first two frames and RANSAC is used to select those that are inliers, i.e. represent background motion. These inliers are then tracked forward in time using the KLT algorithm.

3.5. Proposed Algorithm

The new algorithm introduced here uses a two staged process to refresh a set of SIFT feature points that are tracked using the KLT algorithm. The motivation behind this design may in part be found by examining weaknesses in the two baselines just defined. The baseline-1 algorithm suffers two faults. One is the computation needed to create pairwise correspondences from scratch in each successive frame. Worse, however, starting over with new features in every frame opens up opportunities for mistakes. This tendency is something we've observed in practice and which becomes evident in the empirical evaluation below.

The baseline-2 algorithm suffers a different fault. It depends on the initialization step where the points to be tracked are established based on the first two frames of video. As tracking proceeds, the set of points being tracked shrinks in size. Points get dropped when the KLT algorithm cannot establish with confidence a new position in a new frame. As the scene changes, there is no mechanism for refreshing the points being tracked.

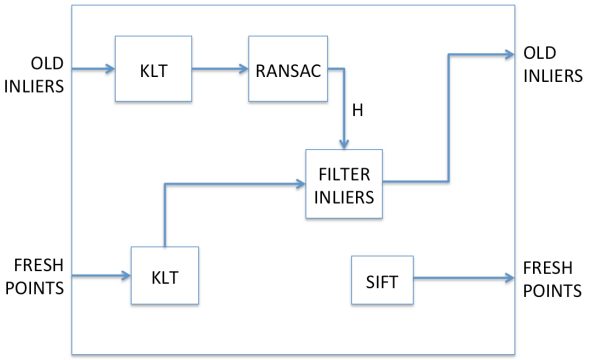

Figure 1: Sketch of a single frame in the Robust Staged RANSAC Tracking Algorithm

Our algorithm solves both of these problems through an iterative staging process which constantly refreshes the pool of SIFT features being tracked. Figure 1 shows the algorithm for a single frame. Two sets of feature points enter a frame on every iteration: 1) a set of **inliers** extracted from two frames previous to the current frame and 2) a set of **fresh points** extracted from the previous frame. Inliers t from the previous frame are used by RANSAC to estimate the motion of the current frame with respect to the previous frame. Once the motion is estimated, the inliers are discarded. The staged fresh points are then tested for compatibility with the estimated motion between the current frame and the previous frame, and the points that are consistent with the motion are passed on to the next frame as the new set of inliers. A new candidate of SIFT features are extracted from this current frame and passed as fresh points to the next frame. We describe this new algorithm as the Robust Staged RANSAC Tracking (RSRT) algorithm.

To initialize the tracking algorithm in Figure 1, we need a bootstrap frame. As shown in Figure 2, SIFT features are extracted from frame 0. The KLT Algorithm updates the position of these features in frame 1 and RANSAC computes the motion, separating inliers from outliers. The inliers are then passed to frame 2 as *inliers*. A new set of SIFT features are extracted from frame 1 and passed as staged fresh points to frame 2. From frame 2 onward, the procedure for every frame goes on exactly as depicted in figure 1.

Figure 3 shows an example comparing tracking with Baseline-2 (top row) with tracking using the RSRT algorithm (bottom row). Blue feature points are inliers as declared by the tracking algorithm; green points are outliers. Frames 82 and 83 are shown because the Baseline-2 algorithm fails here, and the failure is indicative of other failures we've observed on the PaSC videos. Many factors may contribute to a failure, but one obvious factor is the

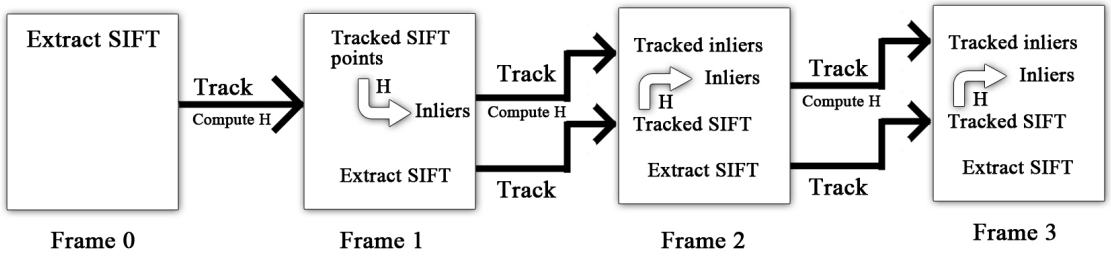

Figure 2: Sketch of the Robust Staged RANSAC Tracking Algorithm with the bootstrapping

(a) Frame 82

(b) Frame 83

(c)Frame 82

(d)Frame 83

Figure 3: Illustration of tracking error made by Baseline-2 algorithm, top row, that is not made by the RSRT algorithm. Blue feature points are inliers and green points are outliers. Note the Baseline-2 algorithm confuses the moving persons arm with the stationary background.

absence of feature points on the right side of the sofa using the Baseline-2 algorithm. This is a direct consequence of that algorithm's inability to add new points as the camera pans to the right.

We have observed one source of weakness in the RSRT algorithm as described. Namely, highly textured objects such as the sofa in Figure 3 can capture almost all the high quality SIFT features, leaving too few to represent the remainder of the background. Therefore, another refinement is added. Instead of allowing the SIFT algorithm to place features, features are densely distributed across the image using a grid pattern. This variant will be describers as the RSRT with dense grid algorithm (RSRT-DG).

4. Results

The two baseline algorithms (Baseline-1 and Baseline-2) and the two new algorithms (RSRT and RSRT-DG) are compared relative to two performance measures. The first measure is how many breaks are inserted into the stabilized videos. The second measure, called the stabilization error score, is a robust pixel-to-pixel difference between succes-

sive frames of stabilized videos.

The stabilization error score is a robust version of the fidelity measure based on Mean squared error used in [14]. It is defined as the sum of the absolute differences between *background* pixels in consecutive pairs of frames, where the background pixels are defined to be 50% of pixels with the smallest frame to frame differences. In other words, for every pair of images we sort the frame-to-frame pixel differences by magnitude and take the sum of all the differences below the median. Pixels that lie outside the video frame are ignored. This error is normalized by the number of pixels which are part of video content in either of the two frames. Although slightly complex, this error score checks how well the background is registered while ignoring pixel differencesthat are due to moving objects. These normalized error scores are accumulated for consecutive frames unless there is a break: pairs of frames across a registration break are ignored.

All four algorithms were run on the 1401 hand-held videos of Point and Shoot Challenge(PASC) dataset [15]. Figure 4 presents four curves summarizing how the mean number of video breaks relates to the stabilization error with different algorithm configurations. By different configurations we mean different thresholds used in creating a break in the video. The circled measurement points indicate default algorithm configurations. As mentioned in section 3.3, we check whether the determinant of the accumulated motion model $H_{1\ldots i}$ falls outside the range 0.95 to 1.05 so that errors does not accumulate in the accumulated motion model. If this range is made smaller, the number of breaks increases, but the frames between the breaks are better registered and have lower overall errors. Similarly with a broader range, the breaks decrease but the overall error scores increase. Both variants of RSRT perform better than the baselines relative to the number of breaks and mean stabilization error at all four thresholds. The dense grid variant of RSRT outperforms the SIFT-based version of RSRT.

Figure 5 shows that RSRT-DG is the most robust frame alignment technique, with the smallest accumulated error and fewest breaks. The breaks that appear in RSRT-DG are mostly due to panning. This results can also be observed in tables 6 and 7.

5. Conclusion

This paper presents a video alignment technique to stabilize videos relative to a fixed background. Our staged multi frame video alignment method efficiently tracks inliers while ensuring that the number of tracked points does not decrease over time. As a result, frame to frame motion is estimated more robustly than in previous techniques.

Our technique is not suitable in domains where the camera is constantly moving or zooming. In such domains, the goal of registering frames to a fixed reference frame is inappropriate. However, as shown here our technique performs better than other image alignment techniques in

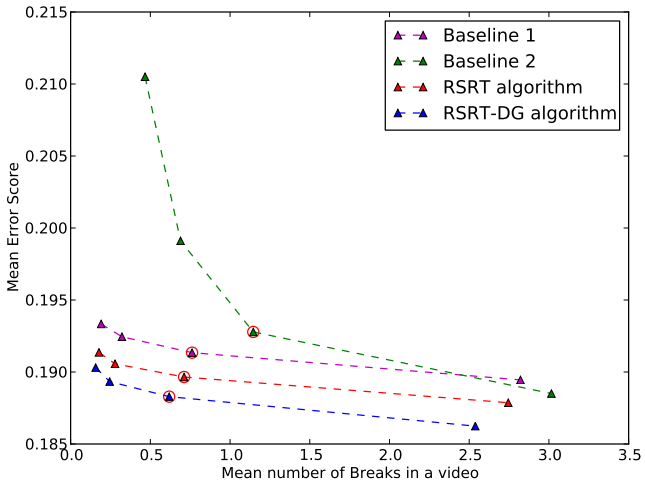

Figure 4: Plot showing how the stabilization error relate with number of breaks for different configurations for each algorithm

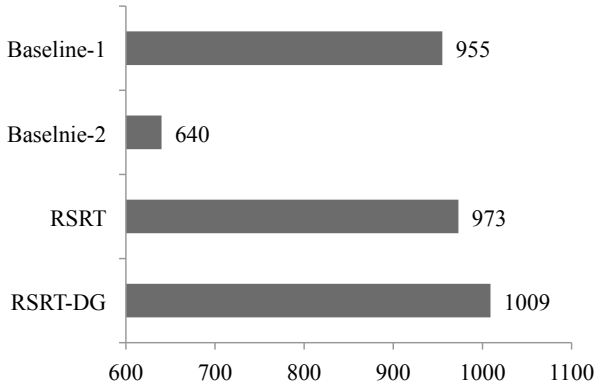

Figure 5: Bar plot showing number of stabilized videos with zero breaks for each algorithm at the default configuration

	Baseline 1	Baseline 2	RSRT	RSRT-DG
Min	0.01103	0.01106	0.01092	0.01067
1st Quartile	0.14070	0.14310	0.13920	0.13880
Median	0.18340	0.18330	0.18190	0.18180
3rd Quartile	0.23190	0.23130	0.22940	0.22780
Max	0.51310	0.55300	0.51220	0.50830
Mean	0.19130	0.19280	0.18970	0.18830

Figure 6: Table showing the summary of Errors for each algorithm over all 1401 videos at default configuration

	Baseline 1	Baseline 2	RSRT	RSRT-DG
Min	0.0000	0.0000	0.0000	0.0000
1st Quartile	0.0000	0.0000	0.0000	0.0000
Median	0.0000	1.0000	0.0000	0.0000
3rd Quartile	1.0000	2.0000	1.0000	1.0000
Max	21.000	15.0000	16.000	14.000
Mean	0.7616	1.146	0.7116	0.6181

Figure 7: Table showing the summary of Break for each algorithm over all 1401 videos at default configuration

videos where much of the camera motion is unintentional. In our technique we have used the first frame as the reference frame. A further challenge would be to dynamically select a reference frame which optimizes the number of breaks and the stabilization error. The source code of this video alignment technique is publicly available at http://www.cs.colostate.edu/vision/rsrt_software/

6. Acknowledgments

Funding for this work was provided by Defense Advanced Research Projects Agency (DARPA) as a part of MIND'S EYE program. We want to thank P. Jonathon Phillips of NIST and Professor Geof Givens of Colorado State University for their help.

References

[1] J. Y. Bouguet. Pyramidal implementation of the lucas-kanade feature tracker, 1999. Tech. Rep., Intel Corporation, Microprocessor Research Labs. 4

[2] M. Bruder, G. A. Roitman, and B. Cernuschi-Frias. Robust methods for background extraction in video. *Argentine Symposium on Artificial Intelligence*, pages 83–95, 2012. 3

[3] D. Capel and A. Zisserman. Automated mosaicing with super-resolution zoom. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 885–891. IEEE, 1998. 2, 3

[4] A. Censi, A. Fusiello, and V. Roberto. Image stabilization by features tracking. In *Image Analysis and Processing, 1999. Proceedings. International Conference on*, pages 665–667. IEEE, 1999. 2, 3

[5] P. J. R. Frank R. Hampel, Elvezio M. Ronchetti and W. A. Stahel. *Robust Statistics: the Approach Based on Influence Functions*. Wiley Series in Probability and mathemetical statisics, 1986. 3

[6] M. Grundmann, V. Kwatra, and I. Essa. Auto-directed video stabilization with robust l1 optimal camera paths. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 225–232. IEEE, 2011. 1, 2

[7] M. Hansen, P. Anandan, K. Dana, G. Van der Wal, and P. Burt. Real-time scene stabilization and mosaic construction. In *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, pages 54–62. IEEE, 1994. 2, 3

[8] M. Heikkilä and M. Pietikäinen. An image mosaicing module for wide-area surveillance. In *Proceedings of the third ACM international workshop on Video surveillance & sensor networks*, pages 11–18. ACM, 2005. 3, 4, 5

[9] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):978–994, 2011. 3

[10] S. Mann and R. W. Picard. *Video orbits of the projective group: A new perspective on image mosaicing*. Citeseer, 1995. 3

[11] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum. Full-frame video stabilization with motion inpainting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(7):1150–1163, 2006. 2

[12] X. Mei, M. Ramachandran, and S. K. Zhou. Video background retrieval using mosaic images. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 441–444, 2005. 2, 3, 4, 5

[13] C. Morimoto and R. Chellappa. Fast electronic digital image stabilization. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 3, pages 284–288. IEEE, 1996. 2, 3

[14] C. Morimoto and R. Chellappa. Evaluation of image stabilization algorithms. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 5, pages 2789–2792. IEEE, 1998. 3, 7

[15] P. J. Phillips, J. R. Beveridge, D. Bolme, B. A. Draper, G. H. Givens, Y. M. Lui, S. Cheng, M. N. Teli, and H. Zhang. On the existence of face quality measures. 1, 7

[16] C. Stauffer and W. E. L. Grimson. Learning patterns of activity using real-time tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):747–757, 2000. 1, 3

[17] C. Tomasi and T. Kanade. *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ., 1991. 3

[18] C. Tomasi and J. Shi. Good features to track. *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition*, pages 593–600, 1994. 4