Thesis

Video Alignment to a Common Reference

Submitted by

Rahul Dutta

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2015

Master's Committee:

Advisor: Bruce A. Draper
Co-Advisor: Ross Beveridge

Chris Peterson

Abstract

Video Alignment to a Common Reference

Handheld videos often include unintentional motion (jitter) and 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 background change and it becomes necessary to fix upon a new background. We present two techniques in this thesis. One technique stabilizes the video with respect to the first available frame. Another technique stabilizes the videos with respect to a best frame. 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 previously published approaches when compared on 1,401 handheld videos from the recently released Point-and-Shoot Face Recognition Challenge (PASC).

Acknowledgements

This thesis would not have been possible without the guidance of my advisor Dr. Bruce Draper and my co-advisor Dr. Ross Beveridge. I would like to thank Dr. Draper for giving me the opportunity to work in the vision lab at CSU, for being such a great mentor and providing me direction with my research. I would also want to thank Dr. Peterson for being in my committee and providing me invaluable feedback. You all have been very helpful when I needed it.

I'd like to thank my vision group lab mates and friends Hrushi, Prady, Som, Jatin and Wimroy who have been more like my family at CSU. I would also like to thank my other colleagues of the vision lab - Mo, Maggie and Hao. Working with them helped me learn a lot. My close friends- Anant, Aritra, Nand, Neelam and Sounak helped me and provided me advice at times of need.

Finally I'd like to thank my parents, my sister Riya and my aunt Sumitra. Their prayers, blessings and sacrifices let me follow my dream and I would not be here without them.

Table of Contents

List of Tables

List of Figures

CHAPTER 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. The goal of this thesis is to remove both kinds of motion, producing videos in which the background doesn't move.

Most recent works [9, 15, 17] on video stabilization aim to remove shakes and jitters while smoothing the presumably intended motion. These algorithms most commonly perform the following steps: 1) estimate the original camera motion by determining the frame-to-frame movement, 2) fit a model to capture the smoothed camera motion and 3) solve for transformation matrices for each frame, warping them to form a new stabilized video without jitter but with apparently smooth camera motion. An excellent example of recent work following this approach is by Grundmann et al. [9], 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.

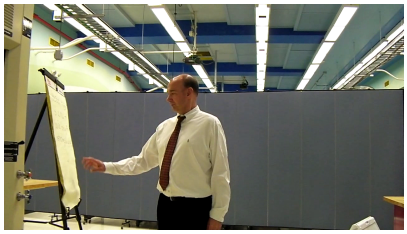
Another possible goal in video analysis is to stabilize a video with respect to it's background and consequently remove apparent camera motion entirely. There are many reasons to remove camera motion. For example, stabilized videos enable background modeling [22, 1] and make object motion more apparent to both human and automated observers. Another reason is to filter the foreground motion from the camera motion can be used as a front end for motion segmentation.

Sometimes it is not possible to remove pan or zoom, for example cameras mounted on moving vehicles like Unmanned Aerial Vehicles. In such domains, more complicated solutions involving multiple object tracking or motion segmentation are required [28, 12]. 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) [21], as shown in Figure 1.2, 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. Frequently they pan the camera in order to keep the subject near the center of the video. The panning motion is typically not large enough, to change completely the background in the scene. Under these conditions, it is reasonable and desirable to stabilize the videos with respect to the background.

Figure 1.1 provides an example video from the PASC dataset used in this thesis. It shows 4 frames from 4 videos, each showing the same subject but with different backgrounds. The person in video 024633d3328, shown in the upper left corner of Figure 1.1, enters the scene on right, moves toward the table on the far left, picks up the phone and then walks back to right. The person in video 02463d3463, shown in the top right corner, follows the same trajectory as in the previous video, but the activity performed is "writing on a paper board". The subject in video 02463d3613 also follows a similar path "shooting a basketball". The video 02463d3561,shown in the lower left corner shows the subject enter the scene from left, pick up a newspaper from a desk on far right and then leave the scene. The experimental design of the PASC dataset is described in more detail in section 5.1.

(02463d3328 frame 200)



(02463d3463 frame 100)

(02463d3561 frame 100)

(02463d3613 frame 114)

FIGURE 1.1. Some background scenes of PASC Dataset

Like much of the prior work on stabilization, our algorithm incorporates detection of salient features [9], point tracking [9, 17], RANSAC [9] for testing potential feature correspondences, and finally motion modeling in terms of frame-to-frame alignment transforms [9]. The major contribution of my thesis as compared to other algorithms is the introduction of a staged multi-frame mechanism named Robust staged RANSAC Tracking (RSRT). Two benifits of this mechanism are 1) new salient features are continuously introduced into the tracking procedure, 2) a two stage filtering process for selecting stable background points. In particular, recognizable features extracted from every frame are staged as the algorithm shifts to the next frame. These staged features are tracked, and only those consistent with the camera motion estimate are retained. This camera motion is estimated from features that are already 2 frames old and are established as background points. Then, these staged features that are compatible with the camera motion, take the place of the features that are 2 frames old. In this way our algorithm provides robustness in estimating the camera motion.

Another difference compared 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 picks a common reference frame and maps all frames back to this reference. In cases where the background change significantly, the original video is broken and each segment is stabilized to a different fixed reference. This process of mapping videos to a common reference frame is related to image mosiacing [5] as discussed in the literature review. The Figure 1.2 illustrates the stabilization process. The imagery turned black in the frames is due to lack of corresponding data when the frame was mapped to the reference frame.

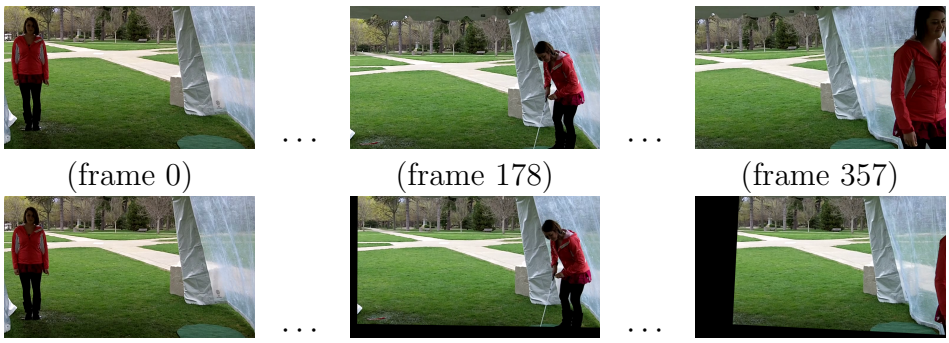


FIGURE 1.2. The top row shows the frames of the handheld video and the bottom shows the corresponding stabilized frames of the video

Having created the RSRT algorithm for stabilizing against the fixed background, there still remains the question of what background should be chosen as the canonical background. One clear choice is the first frame of the video. There are also other choices. Different versions of RSRT algorithm emerged based on the preference of the fixed background: 1) The First Frame RSRT algorithm where we align the video segment to the first reference frame of that segment. 2) Best Frame RSRT algorithm where we determine the best reference frame by processing the whole information from the video and map the other frames to this reference. The first choice is more feasible than the second choice for streaming videos where we need a stabilized frame for every incomming frame. Results presented in Chapter 5 for the PaSC

handheld video demonstrate our algorithm generally does a good job of removing camera motion and also maximizing the amount of background visible.

The rest of this thesis is organized as follows. Chapter 2 reviews related work on video stabilization and image mosaicing. Chapter 3 presents the RSRT Algorithm and our video stabilization algorithm based on the first frame frame. Chapter 4 discusses the Best frame RSRT Algorithm. Chapter 5 presents an empirical evaluation comparing our method to alternative methods based upon prior video stabilization and image mosaicing algorithms in the literature. And Chapter 6 presents a summaryd work and conclusions.

CHAPTER 2

Literature Review

2.1. Related Work

Broadly speaking, the aim of Video Stabilization is to remove unwanted shaky motion. For example, it is almost impossible for a person to hold a camera and not introduce small but rapid movements, i.e. jitters. Video Stabilization algorithms often estimate a smooth camera motion path from video frame data [17]. Matsushita et al. [17] propose a robust approach that produces full frame stabilized videos with enhanced visual quality. First the global motion is estimated between adjacent frames. Then local motion is estimated by computing optical flow between frames after applying a global transformation, using only the common coverage areas between the frames. The local motion field is then used to help fill in missing image areas for non-planar and dynamic scene regions. This technique, known as motion inpainting, is used for video completion. Finally the system uses a motion deblurring technique that transfers sharper pixels to corresponding blurry pixels.

Liu et al. [15] describe a technique for transforming hand-held motions to directed camera motions by simulating 3D camera movements. First, a standard structure from Motion (SFM) system is used to produce a sparse set of 3D points describing the 3D trajectory of the camera motion. Then the system fits a linear or quadratic camera path to the data. Finally, the warp of each input frame into its corresponding output frame is computed using a least squares optimization method, such that the warped frames follow the original 3D camera motion without deforming the content in the frame.

Another novel example, Grundmann et al. [9] (Youtube) presents 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. Like other algorithms, it computes the original camera motion by tracking features between adjacent frames. The algorithm constrains the camera path such that it is composed of 1) a constant path, 2) a path with constant velocity and 3) a path with constant acceleration. Path modeling therefore includes fitting portions of the optimized camera path to constant, linear and parabolic motion models. Apart from minimizing L1 smoothness constraints, the crop window of fixed aspect ratio is moved along the optimal path to satisfy the following constraints: 1) The crop window transformed by the optimal path should be included in the window transformed by the original camera path, 2) the new camera path should preserve the intent of the video and 3) the optimal camera path should include salient points or regions. In Liu et al. [15] and Grundmann et al. [9] algorithms, 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.

The work presented in this thesis is concerned with video stabilization because the goal is to produce a video as though taken by a camera mounted on a tripod. However it breaks from video stabilization because our goal of aligning the video to a fixed background is different from the goal of video stabilization. A different subdiscipline of computer vision addresses image mosaicing. The fundamental goal of image mosaicing aims to create large images by registering sequences of images that partially overlap to common reference frames. Thus the notion that the background frame is fixed and the video frames are aligned to it is very similar to image mosaicing. Some more examples of image mosaicing are described in section 2.1.2.

Video stabilization and image mosaicing both presume an underlying sequence of four steps as nicely outlined by Szeliski et al. [23] which are more elaborated in section 2.1.1.

1) Select distinctive features that can be matched efficiently across frames.
2) Pick the best representation of frame alignment (direct vs sparse pixel-pixel comparisons).
3) Various computational methods to compute the parameters of the alignment (Motion models).
4) Different ways to compute a globally consistent set of alignments over all frames.

2.1.1. Overview - Commonly Used Elements. One common aspect among stabilization and mosaic algorithms is a reliance upon extracting localized matchable features in successive frames. For example, Harris corners[5], SIFT features or Good features[18, 6] can be used for feature selection. Matching techniques like correlation of a window centered about the feature point[5], or tracking methods like Lucas Kanade Tracker[18] or Kalman Filters[6] 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 (4DOF)[19], affine transforms (6DOF)[10] or homography transforms(8 DOF)[6, 5, 18]. 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 additional pairs of matching features in order to constrain additional degrees of freedom. Further, higher DOF motion models are sometimes susceptible to settling on unrealistic frame-to-frame motions, in part due to their much larger space of possible solutions.

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[5] or LMeS(Least Median of Squares)[18] are often used.

2.1.2. Specific Examples of Prior Work. Real-time scene stabilization in video was demonstrated by Hansen et al.[10]. Their VFE-100 system employ a multi-resolution iterative scheme processing Laplacian pyramid images in a coarse to fine search strategy in order to estimate affine motion models. During every iteration, optical flow is estimated using cross correlation of the Laplacian of the current image and the Laplacian of 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 Laplacian image and the current Laplacian image at a higher resolution level of the pyramid.

Morrimoto and Chellappa [19] 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 [5]. Their system used a window-based localized correlation score to match the Harris corners between two consecutive frames. RANSAC was used to discard outlier matches and to estimate the homography that best defined the matched inliers. Finally, the estimates of point correspondences and the associated homography were refined to "corrected" point correspondences by minimizing the Euclidean distance between the original feature points and the corrected feature points of the correspondences. The cost function was minimized over all frames using the Levenberg-Marquardt algorithm to obtain a set of consistent homographies.

Censi et al. [6] approach image mosaicing with feature tracking. Their algorithm tracks "Good features"[24] 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. Once the correspondences are obtained, the 8 DOF homography is computed using a least squares method. The system uses a robust rejection rule named x84[7] to identify outliers. In particular 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.[18] 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[24] 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)[22] background modeling procedure is then used to separate a stable background from points-of-interest.

Heikkila et al. [11] also propose an automatic image mosaicing method for wide area surveillance. Their method extracts SIFT features from the incoming images and then constructs a mosaic. Feature correspondences are found by using the Euclidean distance between SIFT descriptors. RANSAC was used to reject the outliers and the parameters of the homography motion model are refined using the Levenberg-Marquardt algorithm to minimize a geometric cost function defined over the inliers.

Another excellent and recent example of image alignment is SIFT Flow by Liu et al.[14]. The system uses a pyramid based discrete flow estimation algorithm to match the SIFT descriptors between two images in a coarse-to-fine fashion. A dual layer loopy belief propagation technique is used to minimize an objective function at every pyramid level. The objective function consists of three terms: 1) data term which matches the SIFT descriptor along the flow vector, 2) smoothness term which constrains the flow vectors of adjacent pixels to be similar, and 3) displacement term which constrains the flow vectors to be small. This system is primarily used to estimate the correspondence between images of different scene categories. However, it has also been used for register satellite images.

2.1.3. Alignment to a Common Reference. The alignment of frames in a video with respect to a single common reference frame will result in a video where the frames appear motionless except for along the borders where the content changes[4]. 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[16] 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 [20] 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 objects in our videos, we propose in Chapter 5 a related measure less sensitive to differences due to moving objects.

CHAPTER 3

Video Alignment to the First Frame

3.1. Stabilization Algorithms

In video analysis, when our goal is to perform background substraction for **streaming handheld** videos, it is necessary to align the incomming frames to a reference and then extract the foreground from the aligned frame. In such scenarios where we need to process continuous incomming frames, picking the **first available** frame as the reference seems more logical. The First Frame Video Alignment algorithms described here take one video as input and return one or more videos, each of which is stabilized with respect to its first frame as a reference. The first frame of the video segment is selected as a reference and every incoming frame is mapped to this reference. In other words, the goal is for the background to remain fixed throughout each returned video segment. 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 the first reference frame. The criteria for when to break a video is discussed further in Section 3.1.3.

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 3.1.1 and 3.1.2. Then, the algorithms themselves are described in Sections 3.2 and 3.3. Our algorithm presents a consistent way to map or track features between consecutive frames.

3.1.1. Frame-to-Frame Motion Estimation. In general, camera motions are estimated from feature point correspondences. Feature 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:

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

The affine motion model is a fairly standard choice assuming that the features lie on objects that are almost at the same depth. The 6 degrees of freedom (DOF) affine transformation 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 less accurate, due to over-fitting, than when using the 6 DOF transform.

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, then 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 pair-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. [11] used SIFT features to describe SIFT points and used the euclidean distance in the SIFT feature space to map the SIFT points. The other is to use a tracker to move feature points forward from frame $t-1$ to frame t. Mei et al. [18] use the Kanade-Lucas-Tomasi feature tracker (KLT) [25]. The proposed algorithms described in section 3.3 and one of the baseline algorithms described in section 3.2 below use an iterative pyramidal implementation of KLT based on optical flow to provide robustness to large displacements[3].

3.1.2. 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: $\mathit{destination}(x,y) = \mathit{source}(W_x(x,y), W_y(x,y))$ where $f_x(x,y)$ and $f_y(x,y)$ are warping functions. Since $W_x(x,y)$ and $W_y(x,y)$ are seldom integers, bilinear interpolation is used to obtain new interpolated pixel values. If $W_x(x,y)$, $W_y(x,y)$, or both, fall outside the source image, then $\mathit{destination}(x,y)$ is set to zero.

3.1.3. Criteria for Breaking Videos. In this online version of stabilization, 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 (first) 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 first frame using the accumulated motion model $H_{1\ldots i}$. 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 is reasonable 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 $H_{1\ldots i}$. 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 frame-frame motion matrix lies very close to 1.0 for frame-to-frame estimates with the exception of cases where we have sudden distorted or blurry frames or very large changes in view (panning) where the background scene changes completely between two consecutive frames. In those rare cases, we have to break the video. Mostly, 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 2.1.3, and starting over with a new reference becomes necessary.

3.2. Baseline Algorithms

The two baseline algorithms are roughly modeled after the work by Heikkila et al. [11] and Mei et al. [18]. In **Baseline-1**, based upon Heikkila's approach, new SIFT Features are extracted for each new frame. The features of two adjacent frames are mapped by seeking the ratio of two closest matches and the motion model is estimated from the feature correspondences with RANSAC rejecting the outliers. In **Baseline-2**, based upon Mei's approach, good features are extracted from the first frame and are then tracked forward in time using the KLT algorithm. The outliers of the feature correspondences are rejected by a Least Median of Squares (LMeS) algorithm and the motion model is estimated from the inliers.

3.2.1. Problems with Baseline Algorithms.

The baseline-1 algorithm suffers two faults. One is the computation at the cost of creating pairwise correspondences from scratch in each successive frame. The second is that, starting over with new features in every frame creates mistakes since there is a tendency of the system to identify wrongly the foreground points as background points as seen in Figure 3.3. 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.

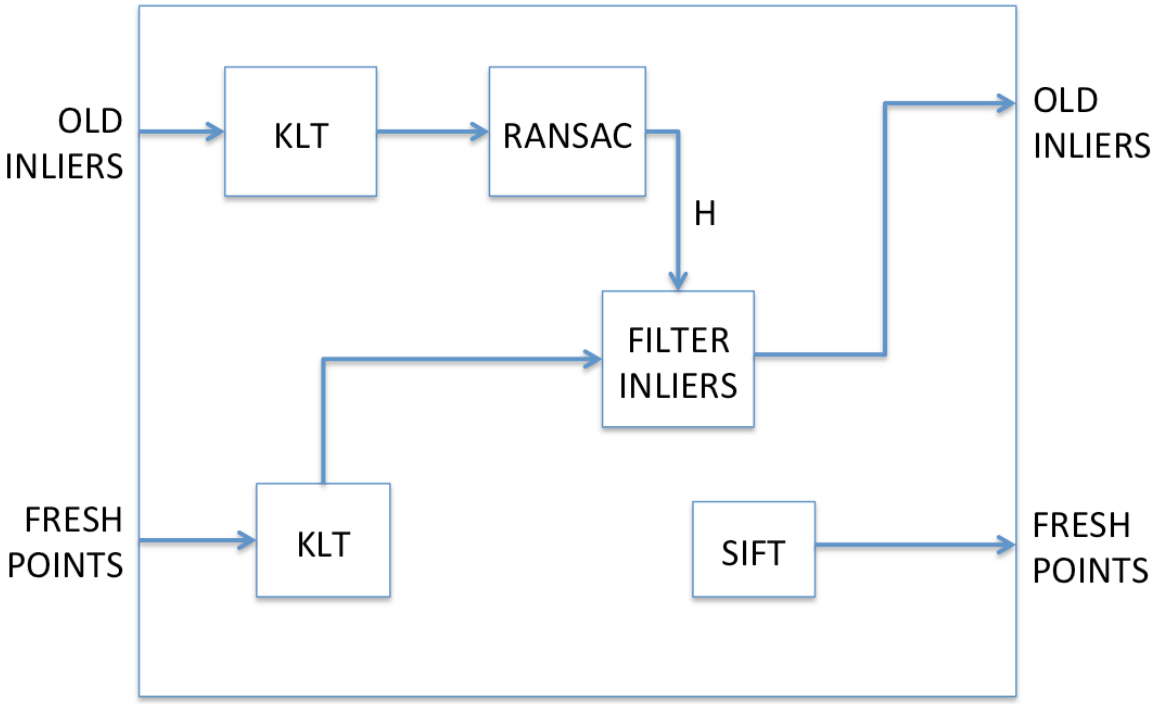


FIGURE 3.1. Sketch of a single frame in the Robust Staged RANSAC Tracking Algorithm

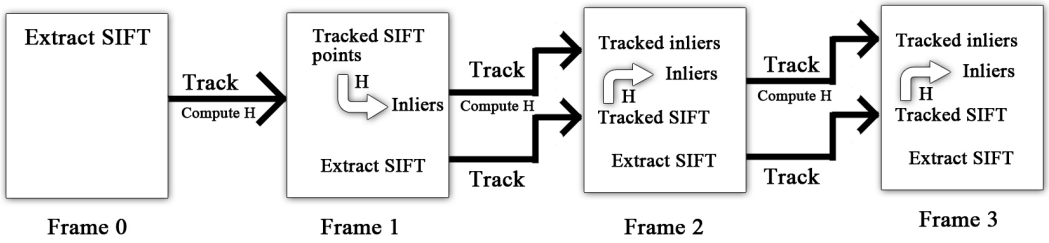


FIGURE 3.2. Sketch of the Robust Staged RANSAC Tracking Algorithm with the bootstrapping

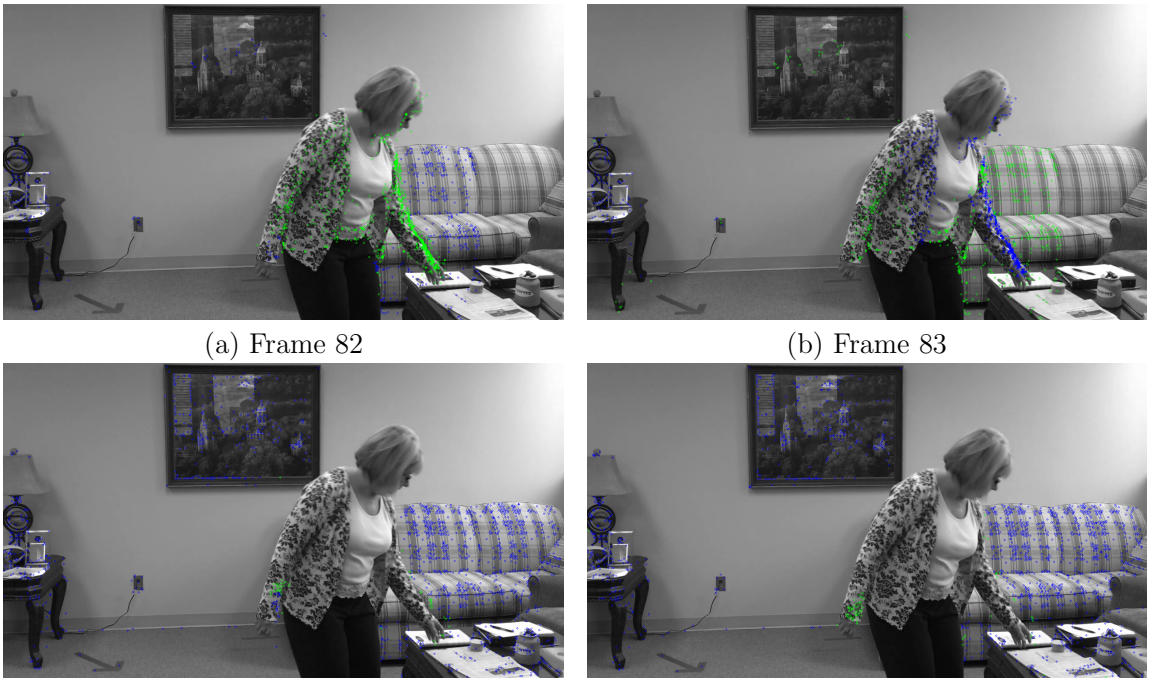
3.3. PROPOSED ALGORITHM

The new algorithm introduced here uses a two-stage process to refresh a set of SIFT feature points that are tracked using the KLT algorithm. The motivation behind the design of our algorithm can be found by examining the weaknesses in the two baselines described in section 3.2.1.

Our algorithm solves both of these problems through an iterative staging process which constantly refreshes the pool of SIFT features being tracked. Figure 3.1 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 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, these old 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.

The new points extracted in every frame ensure that total number of points tracked over time, never decrease below the threshold. The comparison of the motion of a feature point with the estimated camera motion at every stage makes sure that we are mostly tracking the background points. So at every stage, the old inliers are points extracted two frames prior to the current frame and are already filtered as background points. We describe this new algorithm as the Robust Staged RANSAC Tracking (RSRT) algorithm.

To initialize the tracking algorithm in Figure 3.1, we need a bootstrap frame. As shown in Figure 3.2, 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. These inliers are then passed to frame 2 as *established background points*. 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 is repeated as depicted in Figure 3.1.



(a) Frame 82 (b) Frame 83

(c)Frame 82 (d)Frame 83

FIGURE 3.3. 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.

Figure 3.3 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 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 3.3 can capture almost all the high quality SIFT features, leaving too few to represent the remainder of the background. Therefore,

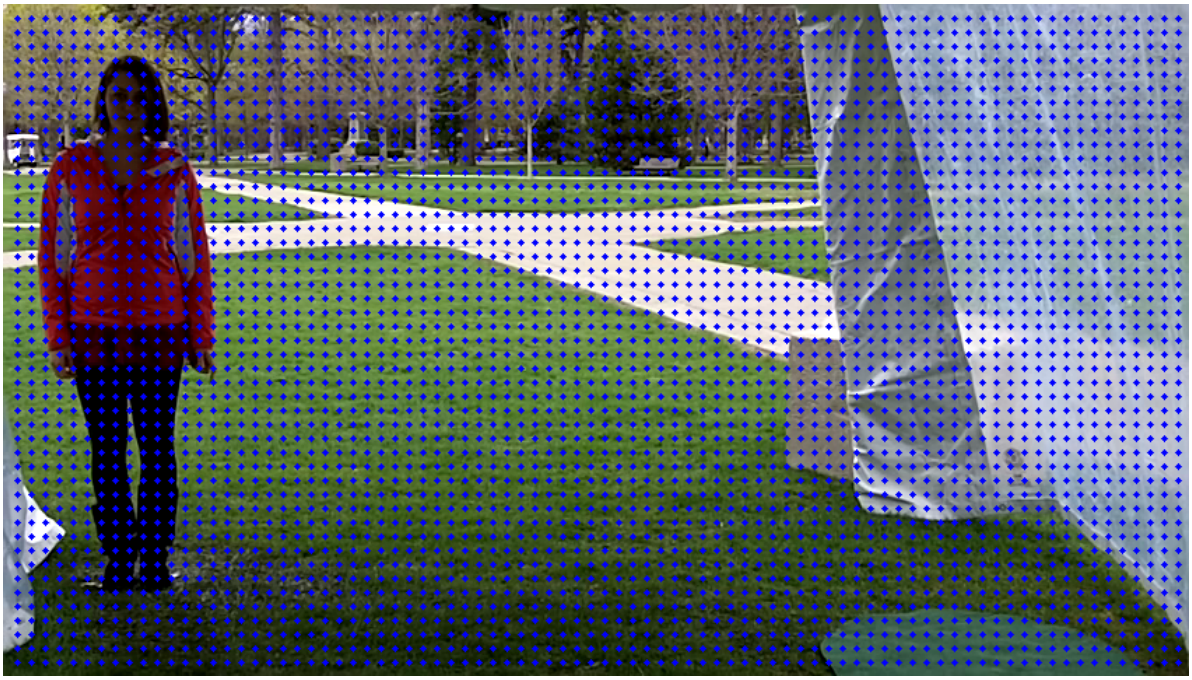


FIGURE 3.4. Dense grid extracted from a frame

another refinement is added. Instead of allowing the SIFT algorithm to place features, features are densely distributed across the image using a grid pattern. We consider a dense grid of unifromly sized square cells. Then we choose the bottom right corner of every square cell as a feature point in that cell as show in Figure 3.4. By default we select the cell width and height as 10. The space between two horizontal or vertical feature points is the cell width or cell height. The spacing between two adjacent feature points can be changed by adjusting the cell size. Another option would have been to select a SIFT feature point in the grid cell. But this would result in more processing time. Since our aim is to uniformly spread out the feature points over the frame, a grid of uniformly spaced points suffice. This variant will be described as the RSRT with dense grid algorithm (RSRT-DG). The First frame RSRT and RSRT-DG are the default versions of RSRT and RSRT-DG.

CHAPTER 4

Video Alignment based on Best Frame

Till now we have been discussing the way of registering frames to the first reference frame and breaking videos into segments when there is no sufficient overlap between the incoming frame and the reference frame. This is the only feasible choice for real time video alignment. Given a complete video however, we can dynamically select a best fit reference frame that can minimize the breaks and maximize the overlap of the frames with respect to that selected reference frame.

To understand the problem, it is helpful to have an example in mind. We consider a video of a person pacing back and forth in front of a doorway. An amateur handling the camera, might very well follow the person in such a way that the person always remains at the center of this camera view. Let's assume we want the same video, but with the door and all the background associated with it becoming stable and fixed. Under such a circumstance, the choice of the best background is pretty obvious. We deliberately created an example where the doorway represented the focal point of the background. Thus the proper choice for the image that should serve as the reference geometry for the fixed camera position is the one that places the door at the center.

In this particular example, if the background is made stable with the door at the center, the amount of total image area over the course of the video that will turn black or that will lie outside the reference because there was no corresponding data, will be minimized (an illustration of the pixels lying outside the reference frame was shown before in Figure 1.2). This concept of minimizing the number of pixels that lies outside the reference frame for

lack of having data is essential to understanding our strategy for selecting the best camera geometry.

Here we are not concerned with cases where cameras are changing their viewpoint to such an extent that the entire background changes, or even changes multiple times. We're worried about cases where a good portion of the background remains visible throughout the course of the video, and all were left trying to decide algorithmically is what camera reference frame to select so that over the entire extent of the processed and stabilized video maximum amount of background remains in view as possible.

The problem of maximizing the overlapping portion of the frames with respect to the background can thus be described as minimizing the amount of imagery turned black. In sections 4.1 and 4.2 we discuss three techniques to pick a reference frame to minimize the number of breaks and maximize the overlapping portions. For frame to frame registration, we use the same RSRT algorithm as discussed in section 3.3 and compute the cumulative motion models corresponding to every frame. However, the criteria for breaking videos is a little different from the alignment process where we pick the first frame as reference frame. We still trigger a break based on the determinant of the frame-frame motion matrix. But for detecting pan, we use a new technique discussed in section 4.5.

4.1. Selection of the best Geometry - Approach 1

In our first approach we transform every pixel of every frame by the cumulated motion models (that we have obtained from our RSRT algorithm) corresponding to every frame.

$$\hat{p}^i_{x,y} = H_{1\cdots i} \times p^i_{x,y}$$

where $H_{1\cdots i}$ is the cumulated motion model corresponding to frame i,

$p^i_{x,y}$ is a pixel point x,y in frame i,

$and \hat{p}^i_{x,y}$ is pixel point transformed by the cumulated motion model.

These transformed pixel points ($\hat{p}^i_{x,y}$) of all the frames are plotted in a 2D space as shown in Figure 4.1. The yellow points in Figure 4.1 represent the transformed pixels. To maximize the overlap, we aim to determine an imaginary rectangle with the same dimension as the video frame, enclosing the maximum number of transformed pixel points in the 2D space. This rectangle is selected as the reference frame. In Figure 4.1 the blue rectangle represents the position of the first frame and the red rectangle represents the reference rectangle computed by this approach.

To determine the reference rectangle, we find the axis which captures the maximum variance of all the transformed pixel points ($\hat{p}^i_{x,y}$) in the 2D space using Principal Component Analysis. We calculate the covariance matrix from the 2D points in space and perform the

Singular Value Decomposition of the covariance matrix to obtain the eigenvectors.

$$P = \begin{bmatrix} \hat{X}_1^1 & \cdots & \hat{X}_N^1 & \cdots & \hat{X}_1^F & \cdots & \hat{X}_N^F \\ \hat{Y}_1^1 & \cdots & \hat{Y}_N^1 & \cdots & \hat{Y}_1^F & \cdots & \hat{Y}_N^F \end{bmatrix}^T$$

$$Cov(P) = PP^T$$

$$PP^T = \begin{bmatrix} \sum_{p=1}^{N} \sum_{i=1}^{F} \hat{X}_p^i \hat{X}_p^i & \sum_{p=1}^{N} \sum_{i=1}^{F} \hat{X}_p^i \hat{Y}_p^i \\ \sum_{p=1}^{N} \sum_{i=1}^{F} \hat{X}_p^i \hat{Y}_p^i & \sum_{p=1}^{N} \sum_{i=1}^{F} \hat{Y}_p^i \hat{Y}_p^i \end{bmatrix}$$

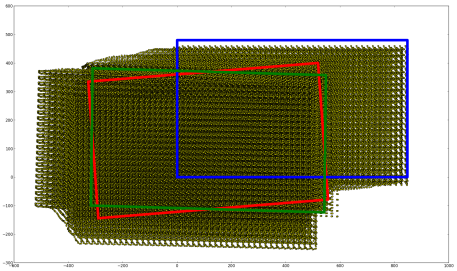
$$PCA = SVD(COV(P))$$

$$SVD : PP^T = R\lambda R^{-1}$$

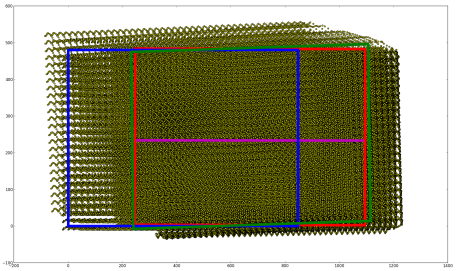
The columns of R represent the eigenvectors and the diagonal values in λ represent the eigenvalues corresponding to the eigenvectors. The eigenvector corresponding to the maximum eigenvalue is the axis capturing the maximum variance of the pixel points. The eigenvector with the second largest eigenvalue is perpendicular to the first eigenvector and captures the second largest variance.

The maximizing rectangle is drawn with its width along the first eigenvector such that the center of the rectangle coincides with the mean of all the transformed pixel points ($\hat{p}_{x,y}^i$). The height of the rectangle is perpendicular to the 1st eigenvector. The width and height of the rectangle are same as frame width and frame height respectively. In the Figure 4.1, the maximizing rectangle, shown in red, is drawn along the 1st eigenvector which is shown in pink.

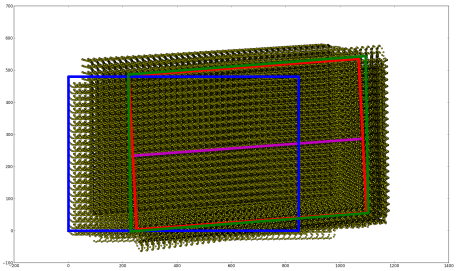
4.1.1. Drawback. The main drawback of this approach is that sudden sharp angular jitters can rotate the rectangle. The PCA approach tries to find the axis which will maximize the total variance. And while doing it, the outlier frames can result in unwanted rotation of the maximizing rectangle. Thus to prevent those sharp jittery frames from contributing towards the reference frame, other approaches were proposed.



(a) Video 05809d1439



(b) Video 06051d1126



(c)Video 06557d516

Figure 4.1. Illustration of Approach 1.

4.2. Selection of the best Geometry - Approach 2 and 3

In Approach 2, instead of considering all the pixels of every frame, we consider only three pixel points lying at the vertices of an equilateral triangle at the center of the first frame as shown in Figure 4.2. As the camera moves from frame-to-frame, the three points on the triangle also translate. These three points on the triangle are transformed by the cumulated motion models ($H_{1\ldots i}$) corresponding to every frame (i), to obtain a temporal sequence of points for every vertex. This produces three independent paths of points in a 2D space

as shown in Figure 4.3. These three paths roughly represent the movement of camera in the 2D space where each point on the path represent the translated position of the camera corresponding to a frame, with respect to the original position. In approach 2, for every path we select the mean point (the point that is equidistant to all other points on that path) as the reference point. By connecting the mean points from those three paths, we obtain the Mean reference triangle. This mean reference triangle corresponds to the mean frame.

For approach 3, we compute the median point of the three independent paths and thereby select the median reference triangle.

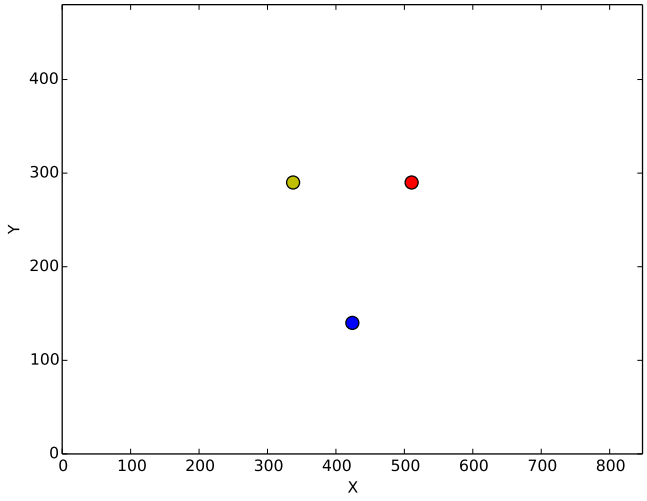


FIGURE 4.2. Plot showing the three vertices of the reference equilateral triangle on the first frame

4.3. Represent the Motion Models with respect to Reference

The cumulated motion models $H_{1\cdots i}$ for every frame are determined, with the first frame as a reference frame by our RSRT algorithm. Next, we need to determine the cumulated motion models with respect to the determined reference rectangle (as in case of Approach

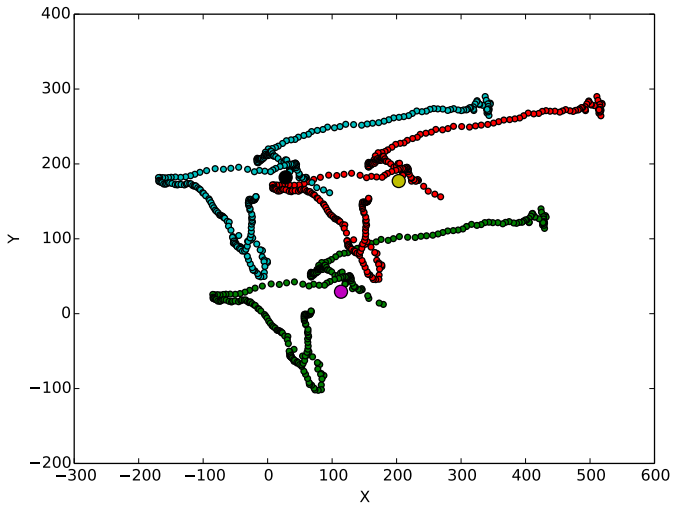


FIGURE 4.3. Plot showing the tracks of three vertices as transformed by the cumulated motion models

1) or reference triangle (as in case of Approaches 2 and 3). First, we calculate the reference motion model that relates the reference frame to the original frame.

Approach 1:

$$\begin{bmatrix} R^1_{ref} \\ R^2_{ref} \\ R^3_{ref} \\ R^4_{ref} \end{bmatrix} = H_{ref} \times \begin{bmatrix} R^1_0 \\ R^2_0 \\ R^3_0 \\ R^4_0 \end{bmatrix}$$

where R^i_0 are the 4 coordinates of original rectangular frame

where R^i_{ref} are the 4 coordinates of reference rectangle

Approach 2 and 3:

$$\begin{bmatrix} T^1_{ref} \\ T^2_{ref} \\ T^3_{ref} \end{bmatrix} = H_{ref} \times \begin{bmatrix} T^1_0 \\ T^2_0 \\ T^3_0 \end{bmatrix}$$

where R^i_0 are the 3 coordinates of the triangle in the first frame

where R^i_{ref} are the 3 coordinates of reference triangle

Now the new cummulated motion models H^i_c can be expressed in terms of H_{ref} and $H_{1\cdots i}$ as:

$$H^i_c = H_{1\cdots i} \times H^{-1}_{ref}$$

These new cumulated motion models (H^i_c) can be used to warp the respective frames with respect to the reference frame.

4.4. Local Search about the Reference frame

The evaluation criteria for measuring the quality of a reference frame is the total number of image pixels that lie outside the reference frame as shown in Figure 4.4. This is defined in more detail in chapter 5. The fewer pixels outside the reference frame, the better. The goal is to minimize the number of pixels lying outside the reference. In theory this objective function is a non-linear and non-convex function. So we cannot use a linear or a quadratic optimizer to minimize this function. We can use optimization techniques like hill climbing or Gradient Descent. Here we use a steepest descent hill climbing approach.

To increase the overlapping portion of the video with respect to the reference, a greedy local search optimization technique (shown in Figure 4.5) on the reference motion model

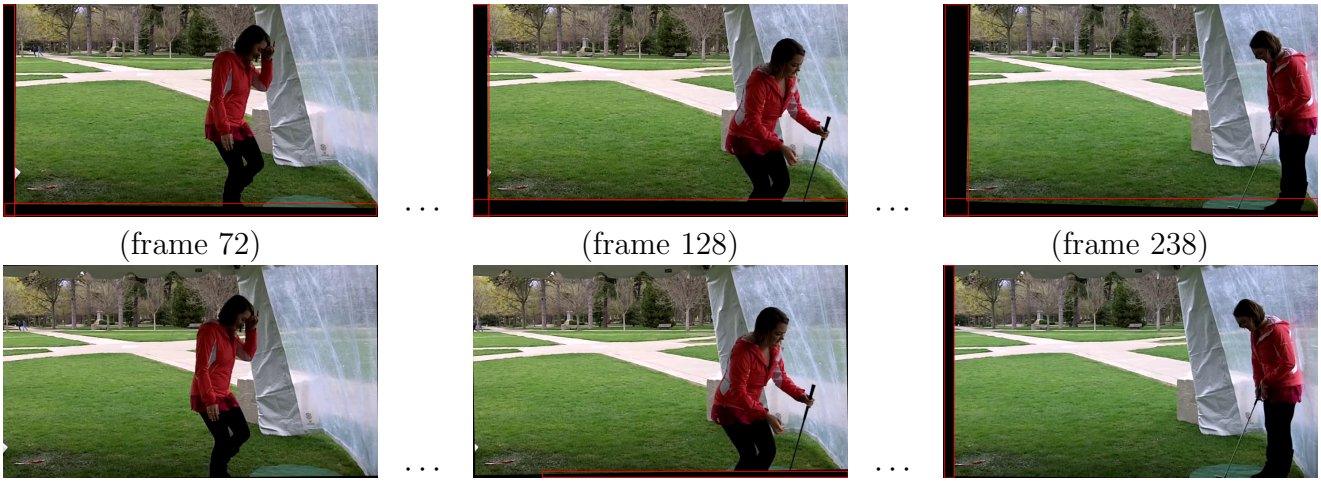


FIGURE 4.4. The top row shows the frames of the stabilized video by first frame RSRT and the bottom row shows the corresponding stabilized frames of the video by best frame RSRT. The pixels enclosed inside the red box are the ones that lie outside the reference.

is employed to obtain a reference model that minimizes the number of pixels outside the reference. The local search algorithm starts from a candidate solution and then iteratively moves to a neighbor solution in the search space. Here the candidate solution is the reference motion model H_{ref} obtained by the methods discussed in sections 4.1 and 4.2. The search space is the x and y translational components of the affine reference motion model. So if we denote the reference model as:

$$H_{ref} = \begin{bmatrix} a & b & x \\ c & d & y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

then at every iteration we search the 8 neighbors of (x, y) i.e. $(x+1, y+1) \cdots (x-1, y) \cdots (x-1, y-1)$. Every neighbor is at a step size 1 away from the current translational components. This step size can vary from coarser values like 16 and 8 to finer values like 2 and 1. The criterion for selecting the neighbor is the total number of pixels lying outside the reference frame over the course of the stabilized video once mapped to the new reference frame.

The neighbor with the minimum criterion value is selected as the solution and becomes the candidate for the next iteration. For instance, if the solution is $(x, y+1)$, then for next iteration the 8 neighbors of $(x, y+1)$ is searched for minimum. When no improving configurations are present in the neighborhood, we say the local search has reached local optimum or minimum and the solution is the new reference motion model.

One way to implement this local search, is to iteratively update the translational components by a single step size till we reach a minimum. This can take a long time since the solution can lie far from the starting candidate solution. For this reason, a more time efficient approach is required. To speed up the process, we use an iterative local search technique where step sizes are of different scales moving from large to small. First, we search with a higher scale of step size for an optimum and reach the solution at that scale. Then, we search the neighborhood with a step size of lower scale about the candidate solution, obtained at the end of local search from the previous scale to locate a more accurate solution. We keep on reducing the scale till it reaches 1. To elaborate, for the first iteration, we use a step size at a scale of 8 for searching the neighbors about every pixel. For example, the neighbors of (x, y) are $(x+8, y+8) \cdots (x-8, y) \cdots (x-8, y-8)$ at this scale. Once we obtain a solution using a step size of scale 8, in the next iteration we perform the local search using a step size of scale 4 about the solution obtained at scale 8. The process is terminated at step size of scale 1.

Now such iterative local search of various scales can provide various local minima for a non-convex function as shown in Figure 4.6b. However, we have tested on 4 videos to verify that a local search starting from scale 8 and a default local search starting from scale 1 both reach the same minima, signifying that the objective function can be locally approximated as a convex function. Even if this assumption does not hold, our aim is to search for a region

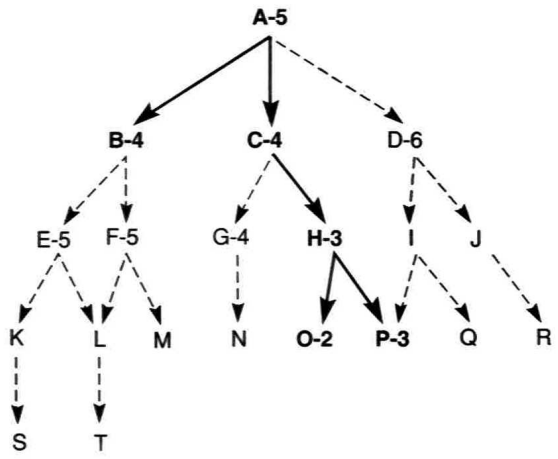
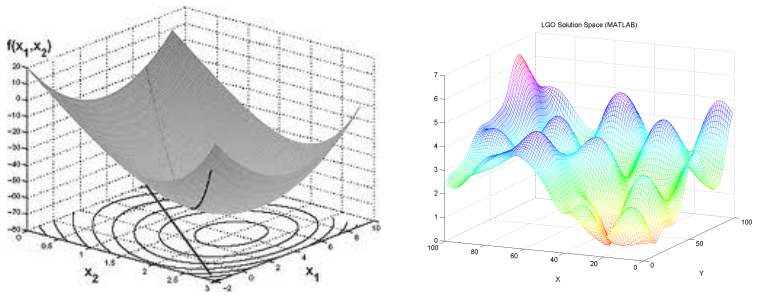


FIGURE 4.5. Illustration of local search (Taken from http://www.cis.temple.edu/ pwang/3203-AI/Lecture/Search-3.htm). The search starts from A whose score is 5, and greedily moves to its neighbor C who has the minimum score among A's neighbors. Next we search the neighbors of C and move to the neighbor H which has the minimum value. We continue this till we reach a local minimum.

having overall lower range of the objective function, during the search at a higher scale. Once we reach a minimum at the higher scale, then we gradually decrease the scale. At the lower scale, the idea is to search for deeper valleys in a more smaller region around the minimum obtained from the previous scale, in the hope of finding a better approximation to the previous minimum. We also use an optimization technique called memoization to store the criterion values for the neighbors so that, criterion is not evaluated twice for a same pixel shift.

4.5. Splitting the Video into segments

In the alignment method based on the first frame, we used the determinant value of the cumulated motion models to determine the criteria for excessive pan. However, for best frame video alignment approaches, when we are processing all the video frames, we can use the information from the whole video, to determine where to break the video. For this



(a) Convex Function (b) Non Convex Function

FIGURE 4.6. Example of Convex Function (Taken from http://metacademy.org/roadmaps/rgrosse/deeplearning/version/25) and a Non-convex function (taken from http://www.gams.com/solvers/solvers.htm

purpose, we use the temporal sequence of vertex points of an equilateral triangle generated as in section 4.2. The sequence of points gives us a fairly good idea of the path of moving camera. The distance between two farthest positions of the camera is used as a criteria for the break of videos. The distance between two farthest points in the sequence of points of a vertex corresponds to the distance between two farthest positions of the camera. Specifically if the distance between two farthest points is more than than some threshold, then we break the video into two segments. Since we have three independent paths for three vertices, we consider the path having maximum distance between its two farthest points.

The complexity of the brute force algorithm to determine the two farthest points from a sequence of points is $O(N^2)$. In order to reduce the complexity to $O(N \log N)$, we determine the convex hull of the sequence and then determine the diameter of the convex hull. The brute force algorithm for determining the Convex Hull is $O(N^2)$. The Graham Scan algorithm [8] by R. L. Graham performs the convex hull computation in $O(N \log N)$. The Graham Scan algorithm first sorts the the set of points in Q in $O(N \log N)$ and then scans the points in linear time ($O(N)$) to build the convex hull.

Once we have obtained the convex hull, our next goal is to obtain the largest distance between two points on the convex polygon. Shamos in his thesis "Computational Geometry" [13] presents an algorithm to compute all the anti-podal pairs in $O(N)$ and the diameter can then be obtained by going through the list, and outputting the pair with maximum distance. We use the Shamo's algorithm to compute the diameter of the convex polygon. Toussaint [26] later suggested that Shamos algorithm is similar to rotating a pair of calipers around the polygon.

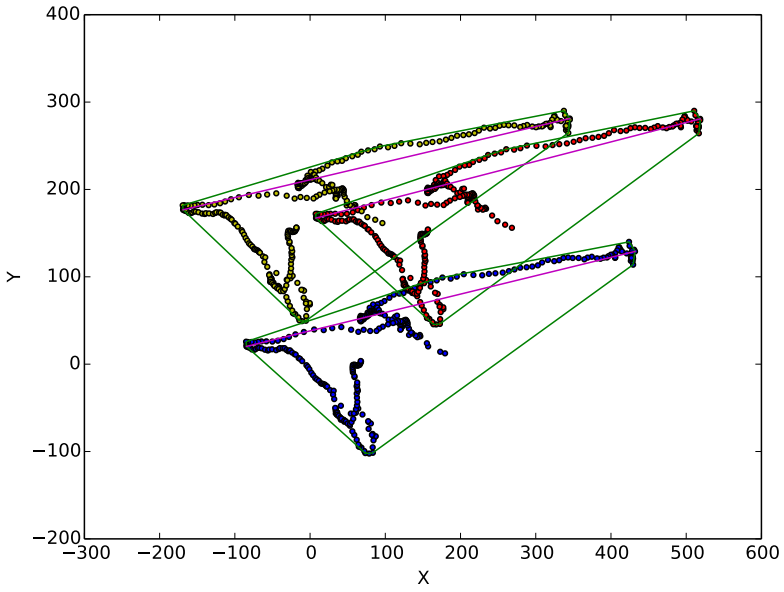


FIGURE 4.7. Illustration of generating the convex hull of the tracks of three vertices as transformed by the cumulated motion models from example shown in Figure 4.3 using the Graham Scan algorithm and computing the diameter of the polygon using vertex and edge antipodals.

4.5.1. DETERMINING WHERE TO SPLIT THE VIDEO. We have described the criteria for splitting videos based on pan. However, we have to determine the frame about which we will split the video. For this purpose, we have used the sequence of points corresponding to a vertex of the equilateral triangle. There are two approaches to this. In our first approach

we find a point in the sequence that is closest to the mean or average of all the points in that sequence as shown in Figure 4.8. In our second approach, we compute the middle point of the starting and the ending point of the set and determine a point in the sequence that is closest to it as shown in Figure 4.9. Since every point in that sequence corresponds to the temporal position of a frame, the selected point refers to the frame, where the split occurs. Once every segment is created, the criteria for splitting is checked again for that segment and again split into sub-segments if necessary.

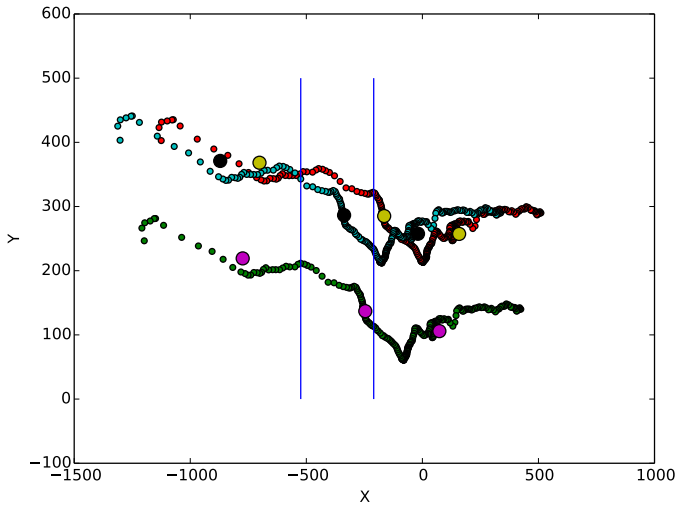


FIGURE 4.8. Figure showing the breaks determined by picking the mean position of the camera positions

In this Chapter we have presented three approaches to select a reference frame. However we will evaluate the mean and median approach, as defined in section 4.2, to generate the reference in section 5. We will perform local optimization on both of these two approaches and observe how the number of pixels lying outside the reference, changes for both the approaches. For determining the position to split the videos, we use the mean position of the camera positions, as described in section 4.5.1, in the evaluation section.

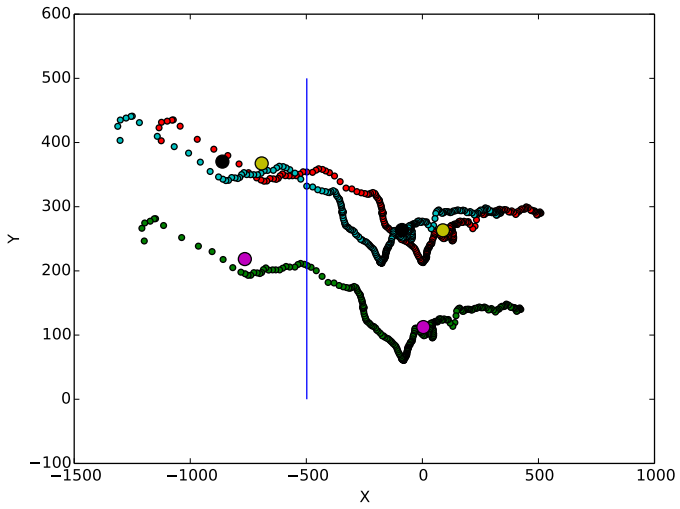


FIGURE 4.9. Figure showing the breaks determined by picking the middle position of the starting postion and ending positions of the camera

CHAPTER 5

Experiments

5.1. The Point and Shoot Face Recognition Challenge Dataset

Inexpensive "point-and-shoot" camera technology combined with social network technology motivates the us to use face recognition technology. The Point and Shoot Face Recognition Challenge (PaSC) [2] was created for two reasons:

- Create a challenge problem that would encourage people to develop better technologies for Face recognition.
- To evaluate and analyse the Face recognition algorithms.

The still portion of PaSC contains 9376 images of 293 people. The PaSC also includes 2802 videos of 265 people carrying out simple actions. Video and still frame data was collected according to an experiment design that systematically varied sensor, location, pose and distance from the camera. Subjects were instructed to carry out actions in order to avoid scenarios where a person looks directly into the video camera for a prolonged time, reducing the task to frontal image recognition with multiple stills. Different actions were scripted for different locations and days.

The 2802 PASC videos are divided into 1401 control videos and 1401 handheld videos. Control videos are shot by a camera mounted on a tripod. Handheld videos are taken by a person who is holding the camera while standing in one place. Sometimes they pan the camera in order to keep the subject near the center the video. In all cases, however, the videos show a person entering the scene relatively far from the camera, carrying out an action, and then leaving the field of view. Typically the person is relatively close to the camera when they leave the field of view, but they do not look directly at the camera. Since

the goal of our algorithm is stabilizing videos by registering frames to a reference, we use the 1401 handheld videos for evaluation purposes.

In my thesis I have presented two methodologies to pick the background. One is the method of picking the background as the first frame which we have described in detail in Chapter 3. The Chapter 4 described a second method which is to pick a best background with greatest overlap. The results of these two methods of picking the background or reference frame are described. For the second method, we evaluate two techniques of picking the best frame.

5.2. Results using First Frame Video Alignment

The two baseline algorithms (Baseline-1 and Baseline-2) and the two new alignment algorithms based on first frame (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 successive frames of stabilized videos.

The stabilization error score is a robust version of the fidelity measure based on mean squared error used in [20]. 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. Since the size of the moving object is less than half the frame size, it can be safely assumed that the differences below the median would not include the pixel differences

arising due to the moving object. Although slightly complex, this error score checks how well the background is registered while ignoring pixel differences that 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 PASC [21] and compared based on these evaluation criteria. Figure 5.1 presents four curves summarizing how the mean number of video breaks relates to the stabilization error with different algorithm configurations. The x axis is the number of breaks. The stabilization error score averaged over all the videos for each algorithm is plotted along y axis.

By different configurations, we mean different thresholds used in creating a break in the video. In particular, the threshold for controlling excessive scaling is altered to generate different configurations. The circled measurement points indicate default algorithm configurations. As mentioned in Section 3.1.3, to trigger large unexpected scale change, we check whether the determinant of the accumulated motion model $H_{1\cdots i}$ falls outside the range x to $\frac{1}{x}$ (where x is the lower threshold and lies below 1, and $\frac{1}{x}$ is the upper threshold), so that errors do not accumulate in the accumulated motion model. By default x is 0.95. When the threshold x is made smaller, the range between the thresholds become broader and when the threshold is made larger then the range becomes smaller. 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. This is seen in Figure 5.1 with the configurations marked on the left portion of each curve having larger range between thresholds and the configurations on the right side of the curve having a smaller range between the thresholds. Overall, Baseline-2 algorithm shows worst performance for wider threshold range and becomes slightly better

than Baseline 1 at the smallest range of threshold. Both variants of RSRT perform better than the baselines relative to the number of breaks and mean stabilization error at all four thresholds and thus the curves remain below the Baseline curves. The dense grid variant of RSRT outperforms the SIFT-based version of RSRT.

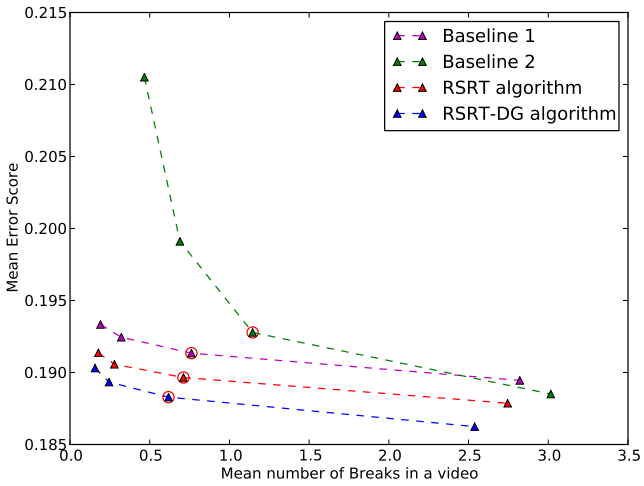


FIGURE 5.1. Plot showing how the stabilization error relate with number of breaks for different configurations for each algorithm

The error bar plot in Figure 5.2 confirms our observations from the Figure 5.1 for all 4 algorithms at default configuration. The horizontal axis is the number of videos with zero breaks. The four different algorithms lie along vertical axis of the bar plot. Since RSRT-DG showed minimum mean breaks and the Baseline 2 had maximum mean breaks, thus the bar plot shows maximum number of videos (1009) with zero breaks generated by RSRT-DG and minimum number of videos (640) with zero breaks generated by Baseline 2 algorithm.

The breaks that appear in RSRT-DG are mostly due to panning. The statistical measures of the average stabilization score and mean number of breaks per video shown in Tables 5.1 and 5.2 respectively establishes that RSRT-DG is the algorithm with the smallest accumulated error and fewest breaks.

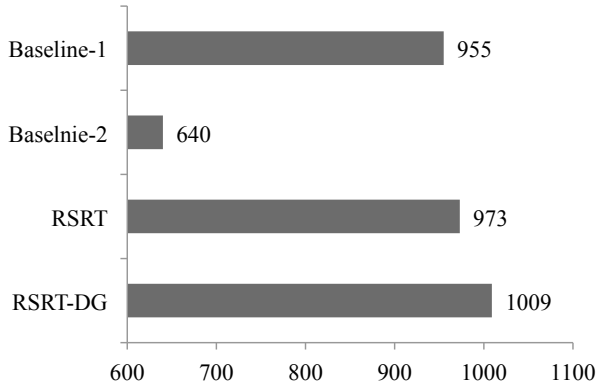


FIGURE 5.2. Bar plot showing number of stabilized videos with zero breaks for each algorithm at the default configuration. Large values are better.

TABLE 5.1. Table showing the summary of Errors for each algorithm over all 1401 videos at the default configuration for each

	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

TABLE 5.2. Table showing the summary of Break 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

5.3. RESULTS USING BEST FRAME VIDEO ALIGNMENT

In addition to the two evaluation measures introduced in Section 5.2, a new measure, the **missed pixel score** is defined for evaluating the best frame video alignment. The missed pixel score is defined as the total number of pixels over all the frames that are lying outside the reference frame when the frames of the video are mapped to the reference. The

missed pixel score corresponds to the black imagery of stabilized frames for lack of having data, described in Chapter 4. The aim of selecting the best reference frame for stabilizing the video in Chapter 4 is to minimize the missed pixel score so that maximum amount of background is visible over the whole stabilized video. All three evaluation measures 1) stabilization error score 2) breaks in videos and 3) missed pixel score are recorded for 1401 videos.

5.3.1. Best frame video alignment vs first frame video alignment. As described in Section 4.2, the best frame alignment technique has two variants 1) the mean technique 2) median technique. In best frame video alignment, both mean and median techniques are used to pick the reference frame and then optimized by local search described in Section 4.4. The results of best frame alignment are recorded for both optimized mean and median variants. The best frame video alignment is run on all 1401 videos for both the mean and median variants and the results are compared with the first frame video alignment based on the three evaluation measures.

Table 5.3. Table comparing the summary of Errors for Best Frame video alignment and First Frame video alignment over all 1401 videos

Name of the Method	Median	Mean
Best Frame video alignment (Mean Technique to pick the reference frame)	0.181177	0.18656
Best Frame video alignment (Median Technique to pick the reference frame)	0.181258	0.18654
First Frame RSRT	0.18190	0.18970
First Frame RSRT-DG	0.18180	0.18830

The Table 5.3 summarizes the stabilization error score for best frame video alignment and first frame video alignment. The stabilization error score is computed for every video and then the statistical measures of central tendency- mean and median are computed for each alignment technique. In best frame alignment, the determined best frame has a higher overlap with the frames as compared to the first frame alignment. Higher is the overlap,

better is the registration of frames. Thus the best frame alignment produces overall lower stabilization error score than first frame alignment, which is apparent from the Table 5.3.

TABLE 5.4. Table comparing the Mean of Breaks for Best Frame video alignment and First Frame video alignment over all 1401 videos at default configuration

Name of the Method	Average number of breaks
Best Frame video alignment	0.031
First Frame RSRT	0.7116
First Frame RSRT-DG	0.6181

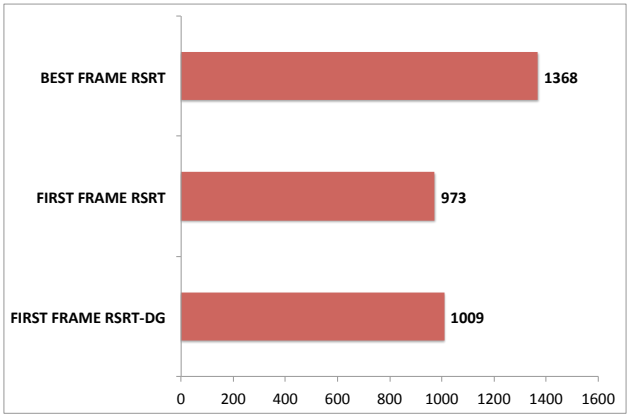


FIGURE 5.3. Bar plot showing number of stabilized videos with zero breaks for for First Frame video alignment and Best Frame video alignment at the default configuration. Large values are better.

Next we evaluate the best frame alignment and first frame alignment based on number of breaks and outline the results in Table 5.4 and Figure 5.3. Table 5.4 and Figure 5.3 are two different but related views of the same evaluation. Table 5.4 presents the average number of breaks for each technique. Figure 5.3 shows the number of videos with zero breaks for each technique. Section 4.5 introduce a new technique for best frame alignment to determine the criteria for detecting pans by determining the largest distance between two camera positions. The first frame technique, on the other hand break a video if more than 50% of the pixels in the current frame lie outside the reference frame. The scale criteria for breaking videos remain same for both alignment techniques. The Table 5.4 shows the best

frame video alignment is more effective than first frame video alignment in bringing down the average number of breaks per video. Also, if we consider the number of videos with no breaks in Figure 5.3, best frame alignment technique is the winner. Since the best frame video alignment has lower average breaks, it records maximum number of videos with no breaks. This is evident since the breaks are determined from scanning the whole video while determining the best frame.

Finally, first frame video alignment and best frame video alignment are compared based on the missed pixel score in Figure 5.4. The missed pixel score for both variants of best frame alignment shows significant improvement as compared to that of the first frame alignment. This justifies our selection of a reference frame based on a local search that maximizes the amount of background visible over the whole stabilized video.

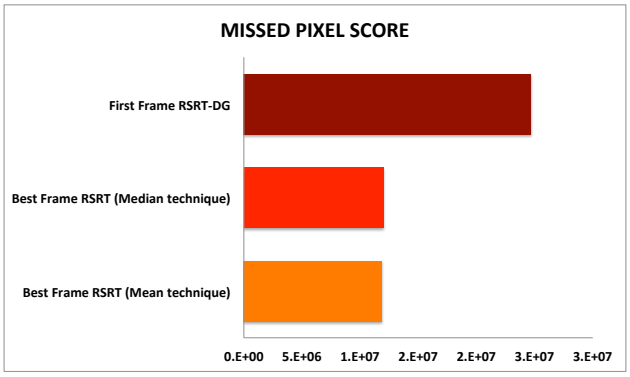


FIGURE 5.4. Bar plot showing missed pixel score for First Frame video alignment and Best Frame video alignment at the default configuration. Smaller values are better.

5.3.2. LOCAL SEARCH: REFERENCE MOTION MODEL AND MISSED PIXEL SCORES BEFORE AND AFTER. In this Section, the missed pixel measure is used to evaluate the performance of two variants of best frame video alignment for picking the reference: 1) Mean technique and 2) Median technique. A reference frame that provides better registration tends to have better overlap with the other frames of the video and thus has a lower missed

pixel score. We perform an experiment on all 1401 videos to investigate the change in missed pixel score during the local search for both the variants (detailed in Section 4.4) and plot the results in Figure 5.5.

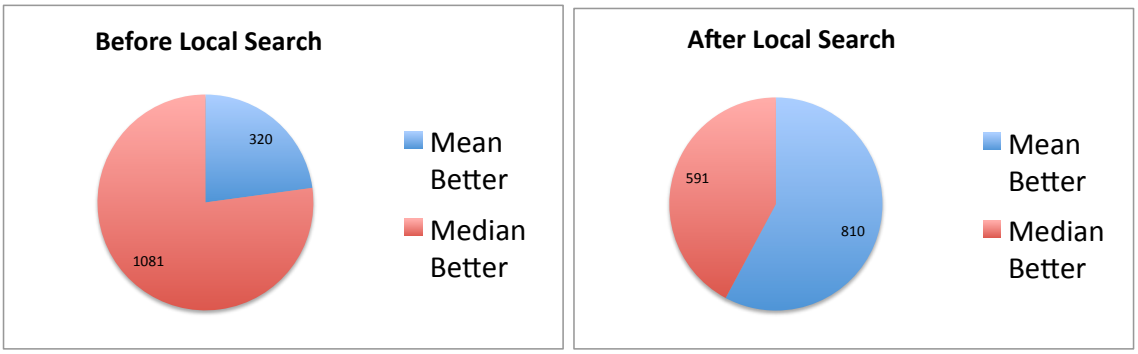


FIGURE 5.5. Illustration showing the Mean Reference Frame performing better than Median frame before the Local Search Optimization. The locally optimized Mean Reference frame is a better reference frame than locally optimized Median Reference frame.

Figure 5.5 shows how the performance of mean and median variants of best frame video alignment varies with the local search optimization technique with respect to missed pixel score. In the first experiment, the missed pixel score is calculated for both the mean and median variants of the best frame video alignment for each video. Then for each video, a decision is made, whether the mean variant is better than the median variant based on the missed pixel score. This result is recorded for all 1401 videos and depicted in the pie chart on left of Figure 5.5. Blue color represent the videos for which mean is better than median technique and red represents the videos for which the median technique is better. As the figure shows, for majority number of videos, the median variant scores lower missed pixel score and is thus better than the mean variant.

In the next experiment, the local search optimization is performed over the mean and median reference frames of best frame alignment and the missed pixel score for both these variants are recorded and compared for each video. The number of videos, for which the

optimized mean reference is better than the optimized median reference, is computed and represented in the pie plot on the right side of Figure 5.5. The pie plot shows that for most videos optimized mean reference performs better than the optimized median reference.

From robust statistics we know that "median is more robust measure of central tendency than mean" [27]. Thus in Figure 5.5 as expected, we find that the median reference frame records a lower missed pixel score than the mean reference for more than 1000 videos out of total 1401 videos before local search optimization. However, after the local search optimization, we observe that the optimized mean reference frame registers a lower missed pixel score than the optimized median reference frame for around 800 videos. The reason behind this behavior of the missed pixel score may be that it is a non-convex function. Even if we assume the missed pixel score to be a locally convex function, still the mean and median technique does not reach the same optima because the search space for the local search is two dimensional translational components whereas an affine model has a total 6 parameters (rotational, scale and shear). However, a six dimensional search space is too expensive and not feasible.

TABLE 5.5. Average difference in missed pixel score between mean and median technique before and after the local search optimization

Before or after the local search	Average difference in missed pixel score between mean and median technique
Before	8.07e05
After	-1.52e05

The Figure 5.5 has already shown that for majority of the videos the median initialization is better than the mean initialization technique before the local search and after local search mean technique mean is better. However, we also want to quantify the amount by which each variant is better than the other, before and after the local search. Thus we run an experiment both before and after the local search, where for every video we record the

difference between missed pixel scores of the mean and median techniques and average them over all 1401 videos. Table 5.5 summarizes the result of this experiment. Table 5.5 shows that the average difference between the mean and median technique before the optimization is almost 8 times more than the average difference between the median and mean methods after the local optimization.

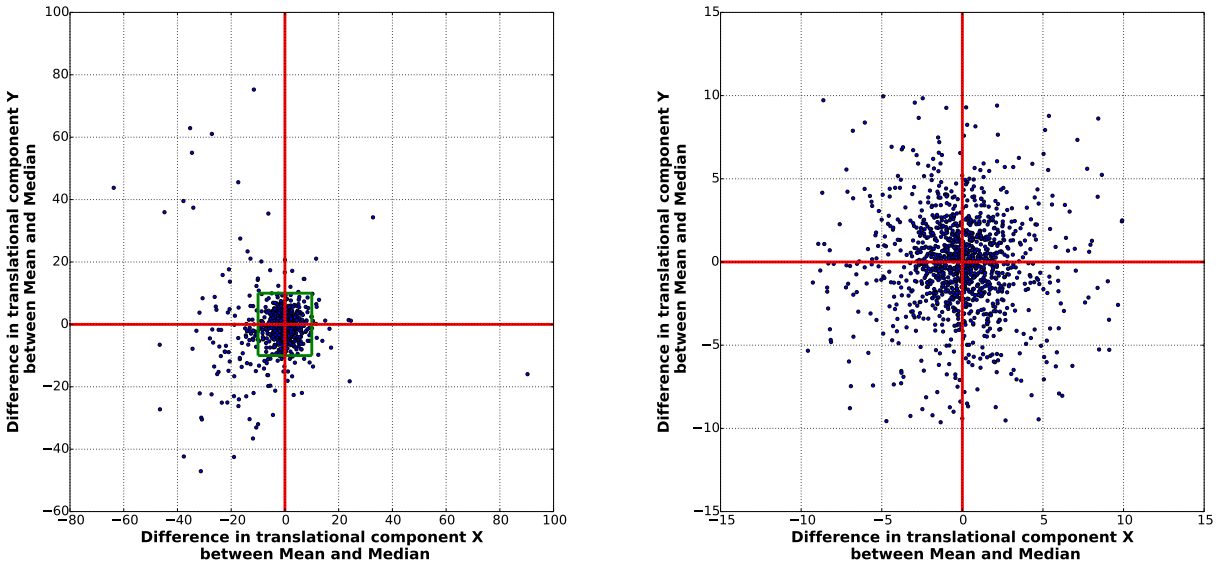
TABLE 5.6. Table showing the average shift in the translational components by the local search to reach the optimum for both mean and median techniques of best frame video alignment

Name of two variants	Average shift in (x,y) translational component to reach minimum
Mean technique	(25.59,4.84)
Median technique	(1.47,1.49)

TABLE 5.7. Table showing the decrease in missed pixel score by the local search to reach the optimum for both mean and median techniques of best frame video alignment

Name of two variants	Average decrease in missed pixel score to reach minimum
Mean technique	9.93e05
Median technique	3.36e04

We are also interested in how close are the reference frames, produced by two techniques of best frame video alignment, to their respective optima. In Table 5.6, while applying local search optimization, the average shift in the x and y translational components of the reference motion model to reach the local minimum, for the median technique is very small as compared to that of mean technique. Similarly, the average decrease in the missed pixel score to reach the local minimum, for the median technique is also less as compared to that of mean technique as shown in Table 5.7. This shows that the reference frame or the motion model produced by the median technique is closer to the optimum than the reference frame generated by the mean technique.



(a) Scatter plot (b) Zoomed scatter plot

FIGURE 5.6. Illustration showing whether the solutions of local search are similar in terms of geometry for both the mean and median techniques. The scatter plot on right (b) zooms in the portion of the scatter plot on the left (a), enclosed in the green rectangle.

Another consideration is whether the solutions of local search are similar in terms of geometry for both the mean and median techniques- that is do they pick the same basic background. This can be roughly determined by the difference between the translational components of the locally optimized reference models of mean and median techniques, for choosing the background frame. The reference model is nothing but a 2-D affine motion model, thus it has a x translational component and a y translational component. We run an experiment on all 1401 videos, where for every video, we compute the differences between the x and y translational components of two locally optimized reference motion models derived

by using mean and median techniques. To elaborate, for a video, if

$$H^{ref}_{\text{optimized median}} = \begin{bmatrix} a & b & x_{median} \\ c & d & y_{median} \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

and

$$H^{ref}_{\text{optimized mean}} = \begin{bmatrix} a' & b' & x_{mean} \\ c' & d' & y_{mean} \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

then the difference in x translational component would be $x_{mean} - x_{median}$ and the difference in y translational component would be $y_{mean} - y_{median}$. We plot every video at the coordinate ($x_{mean} - x_{median}$,$y_{mean} - y_{median}$) as shown in Figure 5.6a. We observe that almost 91% of videos lie within the green rectangle formed by lines $x = -10$, $x = 10$, $y = 10$ and $y = -10$ which is enlarged and shown in Figure 5.6b. The videos which lie far outside the rectangle were investigated and we concluded that all of those outlying videos had sudden large change in view which is associated with the camera panning right or left. This difference was reasonable since the mean and the median reference motion models in case of such motion would certainly vary and reach two very different local minima.

CHAPTER 6

Conclusion

6.1. 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 background points 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.

The RSRT algorithm presented in this thesis is novel in two ways: 1) It presents a 2 stage 3 frame mechanism to select background points, and 2) it introduces new feature points in the tracking. We evaluate the RSRT algorithm with two previous techniques on the PASC hand-held videos. We see that RSRT registers the frames to its reference better than the previous techniques. We also introduce a best frame video alignment technique that process the whole video using the RSRT algorithm and computes a locally optimized reference frame that minimizes the number of pixels lying outside the reference. The best frame RSRT does a even better job in registering frames than the first frame RSRT.

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 videos where much of the camera motion is unintentional and in videos where there is some camera panning.

6.2. Future Works

Our initial goal was to perform human detection on PASC videos. The stabilization algorithm provided in this thesis is useful in removing jitters from hand-held videos so that

the foreground objects can be extracted using some background subtraction model. Then a filter could be used to estimate the moving body location which can be used as a focus of attention for body detection by some DPM model. This particular pipeline is faster and more accurate as compared to the DPM model applied to a whole image:

- Faster because the search of the body is performed in a more focused window of attention instead of the whole image.
- More accurate because it decreases the false positives, that may arise in hand-held low quality videos due to motion blur.

Some examples of head detection are shown in Figures 6.1 and 6.2.

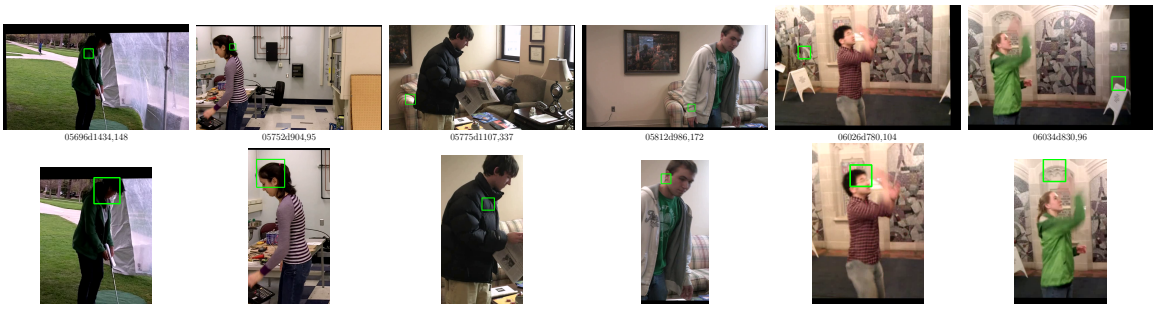


FIGURE 6.1. First Row shows the Articulated Human detection on whole frame. Second Row shows the Articulated Human detection constrained on body region. Labels are as (Video name, Frame number)

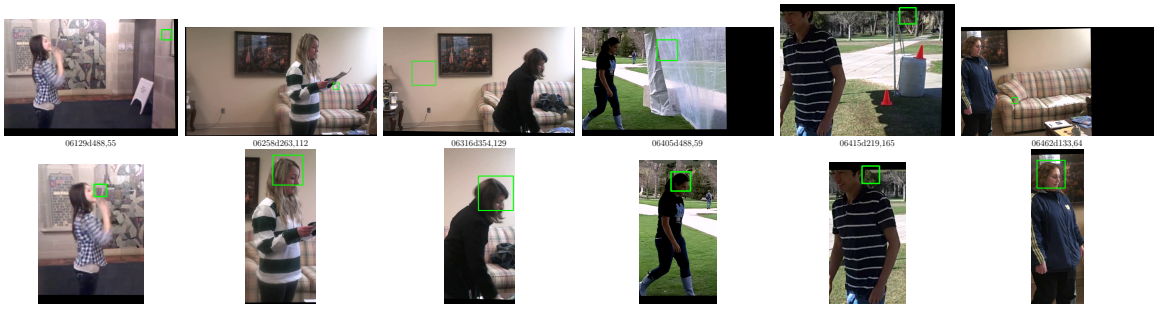


FIGURE 6.2. Some more examples First Row shows the Articulated Human detection on whole frame. Second Row shows the Articulated Human detection constrained on body region, Labels are as (Video name, Frame number)

Bibliography

[1] O. Barnich and M. Van Droogenbroeck. Vibe: A universal background subtraction algorithm for video sequences. *Image Processing, IEEE Transactions on*, 20(6):1709–1724, 2011.

[2] J. R. Beveridge, P. J. Phillips, D. Bolme, B. A. Draper, G. H. Givens, Y. M. Lui, M. N. Teli, H. Zhang, W. T. Scruggs, K. W. Bowyer, et al. The challenge of face recognition from digital point-and-shoot cameras. In *International Conference on Biometrics: Theory, Applications and Systems*. IEEE, 2013.

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

[4] 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.

[5] 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.

[6] 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.

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

[8] R. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, pages 132–133, 1972.

[9] 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.

[10] 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.

[11] 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.

[12] H. Jung, J. Ju, and J. Kim. Rigid motion segmentation using randomized voting. *Computer Vision and Pattern Recognition 2014*.

[13] M. lail Shamos. *COMPUTATIONAL GEOMETRY*. PhD thesis, Citeseer, 1978.

[14] 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.

[15] F. Liu, M. Gleicher, H. Jin, and A. Agarwala. Content-preserving warps for 3d video stabilization. In *ACM Transactions on Graphics (TOG)*, volume 28, page 44. ACM, 2009.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] R. Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2006.

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

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

[26] G. T. Toussaint. Solving geometric problems with the rotating calipers. *Proceedings of IEEE MELECON'83*, 1983.

[27] Wikipedia. Robust statistics, 1999. http://en.wikipedia.org/wiki/Robust_statistics.

[28] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *Computer Vision–ECCV 2006*, pages 94–106. Springer, 2006.