Fast Video Stabilization

A Dissertation Submitted

in Partial Fulfilment of the Requirements

for the Degree of

Master of Technology

by

Abhishek Tiwari

Y9227034

Department of Electrical Engineering

Indian Institute of Technology, Kanpur

June, 2014

CERTIFICATE

It is certified that the work contained in the thesis entitled "*Fast Video Stabilization*", by *Abhishek Tiwari*, has been carried out under our supervision and that this work has not been submitted elsewhere for a degree.

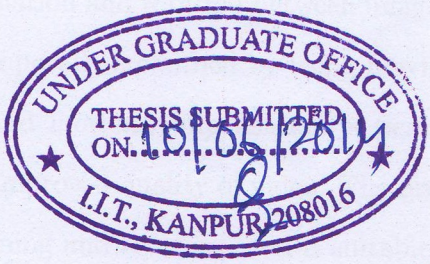
K. S. Venkatesh

Department of Electrical Engineering,

Indian Institute of Technology,

Kanpur-208016.

June, 2014



Abstract

Digital video stabilization is a very important tool to remove jitter and unwanted motion, captured in the video sequences caused by any relative motion between digital camcorder and the subject. Given an unstabilized video sequence, the objective of this thesis is to synthesize a new sequence as seen from a stabilized camera trajectory.

A fast and real-time solution to digital video stabilization is presented in this thesis. The algorithm is based on two-dimensional feature based motion estimation. The method tracks a small set of features and estimates the movement of camera between consecutive frames. An affine camera motion model is used to determine the parameters of translation and rotation between images. Trajectory of computed affine parameters is temporally smoothed by using moving average filter to remove high frequency jitter and motion compensation is performed based on smoothed trajectory resulting in improved quality of video. The proposed algorithm stabilizes horizontal/vertical panning and rotations and is suitable for variety of applications such as handheld camcorder video, camera mounted on car video, cellphone camera video, etc.

The algorithm **adaptively** selects feature points detector threshold as per video frame size to save significant computational costs and appropriate mask width to eliminate unreliable feature points which cannot be used for estimating frame-to-frame motion. Our algorithm features **novel** CPU-GPU parallel computation framework in which both GPU and CPU cores work in parallel and **novel** pipeline implementation of OpenCV routines to bring computational cost down to allow real-time rates for 720p video sequences. In addition to GPU, parallel computation capabilities of multi-core processors was exploited to implement the OpenCV routines pipeline leading to sizeable reduction in computational costs. The minimal computational cost of this fast algorithm gives **73 FPS** for VGA video and **32 FPS** for 720p video sequences. The proposed solution has been tested on a number of video sequences and results have been demonstrated.

Acknowledgments

I take this immense opportunity to express my sincerest gratitude to my supervisor Prof. K. S. Venkatesh for his ceaseless support, guidance and encouragement throughout my Masters Degree program. He always inspired me towards innovation and his creative ideas and simple solutions provided me insights and helped me to develop a simple approach to any difficult problem. I consider myself very fortunate to have been under his supervision and get a chance to learn and grow.

I would like to thank my friends, Shubham for his constant support and encouragement, Vinod and Anand for their valuable suggestions and helpful discussions, Mahendra, Kishalve, Kaushal helping me with programming issues and Pragyanandesh for helping me with GPU. I am also thankful to Shri Narendra Ji for all his assistance in the lab.

I am thankful to my parents and sister who always made me believe in myself supporting me in my pursuits and providing me emotional help at every point. I would like to thank the Department of Electrical Engineering and ACES at IIT Kanpur for providing excellent education, congenial environment and an advanced research facility.

Dedicated to

My Parents

and

My Sister

Contents

Abbreviations

CPU: Central Processing Unit

CUDA: Compute Unified Device Architecture

DIS: Digital Image Stabilization

FPS: Frames Per Second

GPU: Graphics Processing Unit

ITF: Inter-Frame Fidelity

MIS: Mechanical Image Stabilization

NNDR: Nearest Neighbor Distance Ratio

OIS: Optical Image Stabilization

OpenCV: Open Computer Vision

PSNR: Peak Signal-to-Noise Ratio

RANSAC: Random Sample Consensus

RMSE: Root Mean Square Error

SNR: Signal-to-Noise Ratio

List of Figures

Chapter 1

Introduction

This chapter describes the motivation behind this research, a brief about what we did and the process through which we arrived to the algorithmic solution to the underlying problem. Section 1.5 describes organization of thesis.

1.1 Overview

Due to various reasons such as vibrations in a moving vehicle or an unsteady hand, video sequences captured from hand-held camcorders often have unwanted motions. As a result of these unwanted motions, various video stabilization techniques have been developed. A video stabilization algorithm results in a new compensated video sequence where unwanted camera motions have been compensated.

There are many more benefits of removal of jitter in camera video other than increasing the aesthetic appearance of the footage, such as:

- Detection and Tracking of independently moving objects from moving platforms can be greatly improved by removing background motion of the scene. This is important in both robot navigation and scene modelling.

- A steady video can greatly increase the clarity of the picture. For good high quality security video footage, sharp details are necessary.

- Increased compression. Modern digital compressors use a lot of bits to encode

moving features of a video. If the whole image shakes, it is all moving, thereby, wasting an enormous number of bits. As a result stabilized video has a much better compression rate.

- Human fatigue. Unstable footage can be strenuous to watch and having to examine it for hours on end can lead to difficulties. Ensuring operators are alert is a major challenge in real-life security CCTV. Stable footage can greatly help in this situation.

1.2 Video Stabilization Methods

Depending on the approach, three types of stabilizers are used: Optical Image Stabilization (OIS), Mechanical Image Stabilization (MIS) and Digital Image Stabilization (DIS). Next, a brief explanation of these techniques.

1.2.1 Optical Image Stabilization

Optical Image Stabilization systems manipulates the image before it gets to the camera sensor. When the lens moves, the light rays from the subject are bent relative to the optical axis, resulting in an unsteady image because the light rays are deflected. By shifting image stabilization lens group on a plane perpendicular to the optical axis to counter the degree of image vibration, the light rays reaching the image plane can be steadied. Two vibration-detecting sensors for yaw and pitch are used to detect the angle and speed of movement because vibrations might occur in both horizontal and vertical directions. An actuator moves the lens group horizontally and vertically thus counteracting the vibration and maintaining the stable picture.

1.2.2 Mechanical Image Stabilization

Mechanical image stabilization involves stabilizing the entire camera. This type of stabilization can use a motion sensor as a gyroscope or mechanical devices such as

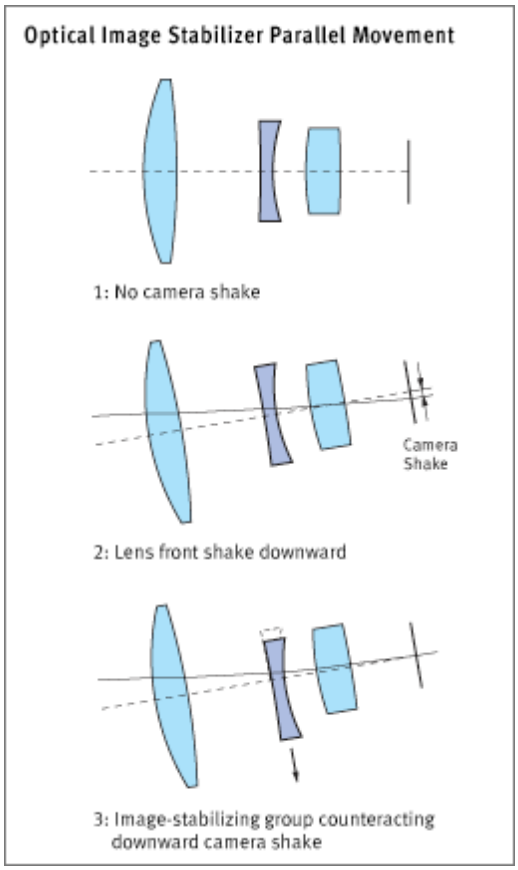


Figure 1.1: Optical image stabilization system

shock absorbers for passively damp any kind of vibrations. Mechanical stabilizer has the advantage of instantly smoothing high frequency vibrations. Commercial products of this type are available such as in Figure 1.2.

1.2.3 Digital Image Stabilization

Digital Image Stabilization (DIS) systems use electronic processing and image manipulation to control image stability. Unlike OIS, the image is manipulated after reaching the sensor and is more cost-effective. DIS systems detects what they think is camera vibration, it slightly moves the image so that it remains in the same place on the sensor. A major risk of this system is the level of noise for some scenes; for instance, a large object moving in the frame may be interpreted as camera vibration and the camera will attempt to stabilize the subject causing a blurring of the image and reduction in picture resolution. Thus robustness of DIS system becomes an important issue.

Figure 1.2: Mechanical image stabilization system

In some DIS, the camera can also use the motion sensors to detect vibrations. Since this method senses movements in the camera and not the image, movement of an object in the image cannot fool it. Unfortunately, not all cameras are equipped with motion sensors then it becomes necessary to use a Computer Vision approach.

1.3 Types of Camera Motion

Due to vibration, the camera can move in different directions. These motions can be characterized as follows:

- Camera moves in Horizontal/Yaw/X-axis direction.
- Camera rotated in Vertical/Pitch/Y-axis direction.
- Camera rotated around z-axis/Roll.

- Focal length is varied.

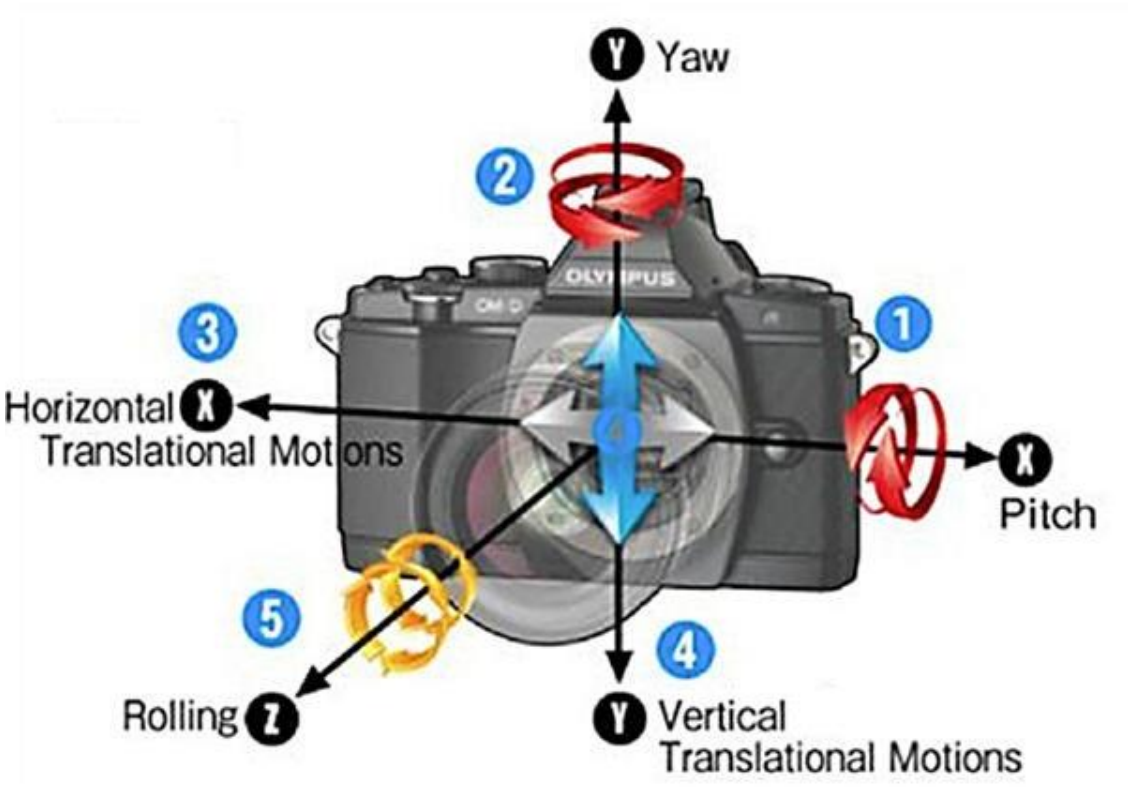


Figure 1.3: Camera motions

Based upon estimation of these motions, video stabilization is carried out. To further categorize, we divided these motions into two broad categories:

- Linear (Translation or 2D Stabilization).
- Rotational (z-axis/zoom/scale or 3D Stabilization).

1.4 Basics of Digital Image Stabilization

Figure 1.4: Digital image stabilization flowchart

Generally there are three main principal modules:

- Global Motion Estimation
- Motion Filtering
- Motion Compensation

Global Motion Estimation block estimates the motion of frame with respect to the previous frame and then sends the parameters to the Motion Filtering block. Motion Filtering refers to the removal of high frequency jitter (unwanted motion) from actual camera motion (desired motion). Finally Motion Compensation block modifies the image and generates the stabilized sequence.

This technique is software based, therefore very flexible. There are no moving hardware parts or system dependency. This is machine independent, low price and suitable for miniature hardware implementation. Precision is very high due to absence of any moving hardware. Real-time and post processing both are possible here. It requires only video sequence and requires no knowledge of camera motion.

1.5 Thesis Organization

The rest of the thesis is organized as follows:

Chapter 2 contains detailed discussion of video stabilization basics and theory. For Motion Estimation methods such as feature tracking, feature registration, block matching, etc. are discussed. Camera motion models and limitations are presented. Robust methods for rejecting outliers and filtering out unwanted motion from desired camera motion are also presented. A brief review of existing video stabilization techniques is presented.

Chapter 3 contains an exhaustive analysis of feature-based image registration methods and algorithms. Matching performance and computational cost for various detector-descriptor combination is evaluated. Performance evaluation of false match rejection techniques is performed.

Chapter 4 contains detailed discussion of the proposed algorithm. Feature tracking based on optical flow is employed for Motion Estimation followed by moving

average low pass filter for camera motion smoothing. Results show the effectiveness of the proposed algorithm.

Chapter 5 concludes the thesis with scope for future work.

Chapter 2

Background Concepts

2.1 Motion Estimation Theory

Estimating the motion is an essential part of performing a satisfying image stabilization. Many camera models have been developed to relate the motion between two frames. This section presents various approaches for Motion Estimation.

2.1.1 Features Tracking

Feature tracking estimates the motion between frames by selecting features from the first frame and finds those features in the second frame, so features that can be accurately tracked must be selected. Optical Flow is used for tracking the selected features.

Feature tracking algorithms generally work well for different camera movements including translation, rotation and different zooming models. Shi and Tomasi [2] *good features to track*, 1994 is most widely used feature selection algorithm for feature tracking.

The feature selection can be of following types:

Region Based Feature Selection

In region based method, image is divided into regions and best feature candidate is selected from each region. Although many features can get selected by this method in close proximity to high texture regions, but it guarantees that all those features will not be selected from a single region in the image.

Separation Based Feature Selection

This method chooses the features with the highest rating from the image guaranteeing that no two selected features are within a specified minimum distance. This method is however more computationally intensive but assures that no clumping of selected features occurs.

Grid Based Feature Selection

This method divides the image into rectangular grids and selects the best feature from each region, guaranteeing uniform distribution of feature points over the whole image.

For each selected feature, local motion vectors (LMVs) between two frames are computed using Optical Flow method. Optical Flow estimates the motion that occurred at the pixel of interest. Two methods are commonly used: Horn and Schunck [3] and Lucas and Kanade [4]. However, these methods can only handle motion of one or two pixels between frames. Thus, a pyramidal version of the Lucas and Kanade method [5] is used to handle large motions.

Motion estimate needed to overlay the current frame on the previous frame in order to minimize the visible motion is computed from the set of feature motion vectors obtained using features tracking. In order to estimate this motion, the feature motion vectors must fit to a frame motion model which will be discussed in detail later in this chapter. Two methods of detecting improperly tracked features will be discussed later in this chapter.

2.1.2 Feature Matching-based Image Registration

Image registration is the process of finding the transformation between two images. One of the image is referred to as the source, reference and the other one is target. Image registration algorithms can be classified into intensity-based and feature-based. Intensity-based methods compare intensity patterns and registers entire images, therefore, requires high computation cost. Feature-based methods establish a correspondence between a number of especially distinct points in images (much lesser than total number of pixels), therefore, computationally cheap.

Pixel-to-pixel correspondence between two frames is established by a geometric transformation to map the target image to the source image, determined using known correspondences between a number of point-pairs in images. Many false correspondence rejection techniques have been developed and robust techniques are used for fitting the transformation, which will be discussed later in this chapter.

Image registration can be divided into four major steps:

1. *Keypoints Detection:* A detector analyses the frame to extract a set of salient keypoints, e.g., interest points that represent the most informative parts of an image. The detectors can be organized into two main classes, namely blob-like feature detectors and corner-like feature detectors. For each class, detectors are presented in chronological order of appearance in the literature.

    (a) Blob Detectors: Blob detectors detect local extrema of the responses of particular filters as keypoints. Most common blob detectors are:

        i. SIFT: The Scale Invariant Feature Transform (SIFT) [6] is widely recognized as the gold-standard approach for scale and rotation invariant interest point detection.

        ii. SURF: The Speeded Up Robust Feature (SURF) [7] detector is based on SIFT and requires much lesser computational effort.

        iii. MSER: The Maximally Stable Extremal Regions (MSER) [8] is used as a method of blob detection in images.

(b) Corner Detectors:

i. GFT: Good Features to Track (GFT) was developed by Shi & Tomasi [2] for feature tracking.

ii. FAST: The Fast Accelerated Segment Test (FAST) [9] detector represents a clear breakthrough in high-speed corner detector.

iii. BRISK: In Binary Robust Invariant Scalable Keypoints (BRISK) [10], FAST detector is executed for each pyramid layer separately.

iv. ORB: Similar to BRISK, Oriented FAST and Rotated BRIEF (ORB) [11] is a refined, scale-invariant version of FAST.

2. *Descriptor Computation:* Image patches around each keypoint are processed further and compactly represented by means of fixed-dimensional descriptors that capture, e.g., their photometric properties.

(a) Non-binary Descriptors

i. SIFT [6]

ii. SURF [7]

(b) Binary Descriptors

i. BRIEF [12]

ii. ORB [11]

iii. BRISK [10]

iv. FREAK [13]

To build a binary descriptor it is only necessary to compare the intensity between pairs of pixels located around the selected keypoint. This allows to obtain a representative description at very low computational cost.

3. *Descriptor Matching:* Brute-Force matcher is employed for feature matching. It takes the descriptor of one feature in the first set and is matched with all

other features in the second set using some distance calculation (different for binary & non-binary descriptors) and the closest one is returned.

For non-binary descriptors, L1-norm or L2-norm matching distance calculation is performed whereas for binary descriptors, Hamming distance is evaluated. Matching binary descriptors is very fast as it requires the computation of Hamming distances, which can be executed very fast through XOR primitives on modern architectures.

4. *False Correspondence Rejection Techniques:* Descriptor matching can lead to false correspondences which must be filtered out to obtain a precise and consistent global motion vector. Techniques commonly employed for false correspondence rejection are:

   (a) Cross-Check Filtering: Only those matches with value (i,j) such that i-th descriptor in set-A has j-th descriptor in set-B as the best match and vice-versa are considered as correct correspondences.

   (b) Nearest Neighbour Distance Ratio (NNDR) filtering: For each feature descriptor in set-A, best and second best match in set-B is computed. If the ratio of distance of first best and second best correspondence is near 1, it implies that the match is false correspondence and not unique descriptor matching. A threshold (0.8 in original Lowe's paper [6]) is fixed for good correspondences.

   (c) Bi-Directional NNDR Filtering: The previous ratio test is performed in both directions (set-A to set-B and set-B to set-A) and only those correspondences which satisfy the threshold in both directions are considered to be good correspondences.

Each computed good correspondence has a motion vector associated to it must fit to a frame motion model which will be discussed in detail later in this chapter.

2.1.3 Block Matching

Block matching algorithms are most commonly used due to their simplicity. The image frame is divided into rectangular blocks and each block is searched in the next image frame in its neighbourhood using some similarity or error metric, e.g., Sum of Absolute Differences (SAD), Sum of Squared Differences (SSD), etc. Various search methods for matching having different computational complexity are discussed:

1. Full Search: It is an exhaustive search method in which the whole image is searched for the best similarity. For example the search can start from one corner of the image and end in other corner, searching for all possible matching blocks based on the similarity metric.

2. Three-step Search: Three-step search is an iterative method in three steps. The three-step search method is illustrated in Figure 2.1. The similarity will be calculated between a specific pixel in the previous frame and all nine pixels in the current frame, marked as the 0 and the eight 1 in Figure 2.1. When the highest similarity value is found, the discrete search distance is reduced and a new similarity value is calculated. Thus a finer search net is used and the accuracy of the search is increased. The procedure is repeated again, this time centered around the boxed 2. The boxed 3 is called the final match and it will represent the most accurate result of the search. Three-step is faster and computationally cheaper compared to full search.

3. Spiral Search: It assumes that motion between frames is small and hence rotational search will prove to be more effective. The search is started from the center (x = 0 and y = 0) of the current block and moving outwards in a spiral manner. Until similarity value reaches a threshold, the search is carried on. The last similarity value in the search spiral will then be considered to be the best match, see Figure 2.2. In best case scenario this search can be faster than three-step and in worst case, this method can be as costly as full search.

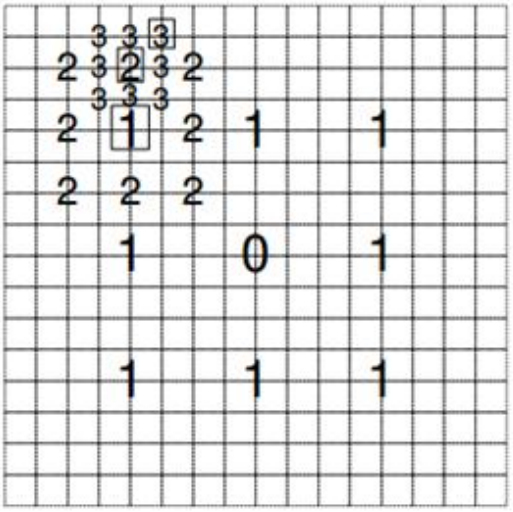


Figure 2.1: Three Step Search

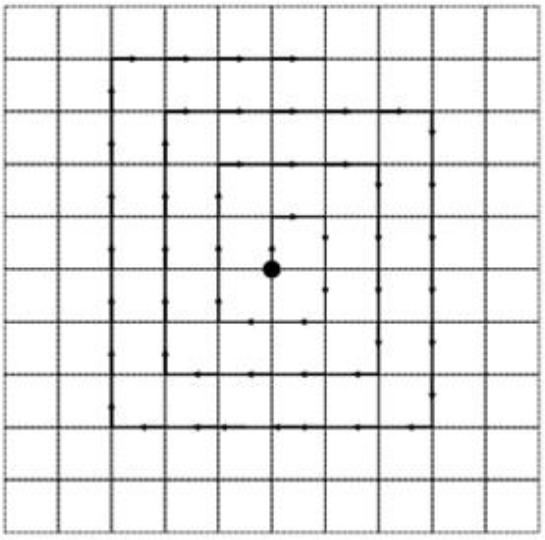


Figure 2.2: Spiral Search

Techniques have been developed to filter out erroneous motion vectors estimated by block matching. To filter out these vectors, the following considerations may be used:

- The error (e.g. SAD) values have to be low (effective match).
- Local motion vectors have similar values in their neighbourhood (motion continuity).
- Local motion vectors referred to homogeneous blocks are not reliable.

Filtered out motion vectors must fit the frame motion model which will be discussed later in this chapter.

2.1.4 Phase Correlation

Phase correlation is a method of image registration, and uses a fast frequency-domain approach to estimate the relative translative offset between two images. The phase correlation method is resilient to noise and occlusions. This method can be extended to determine rotation and scaling difference between two images by first converting the images to log-polar coordinates.

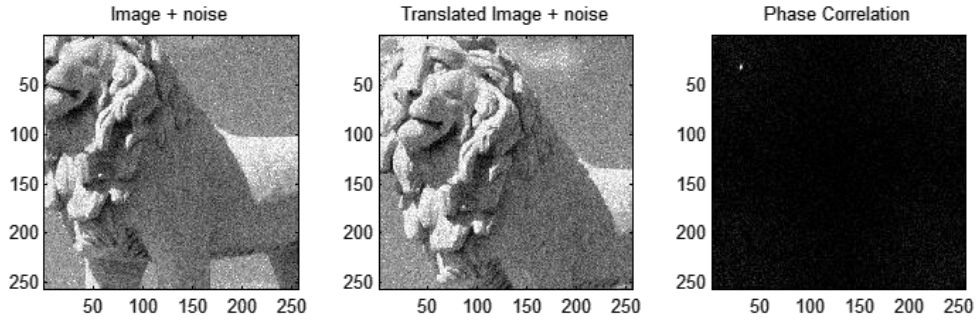


Figure 2.3: Motion Estimation: Phase Correlation (adapted from wikipedia)

Under ideal conditions, phase correlation method output is a single spike (impulse) located at the correct value of translation in x- and y-directions. However, phase correlation fails for periodic images (such as chessboard) where it may yield ambiguous results with several peaks in the resulting output.

2.2 Camera Motion Models

An image is a 2D pattern is the result of the projection of a 3D scene onto a 2D plane. In 2.4, the basic pinhole camera model is used to conveniently simplify a linear mapping from P^3 to P^2. Correspondences under perspective projection between 3D camera coordinate points $\mathbf{X} = \begin{pmatrix} X & Y & Z \end{pmatrix}^T$ and points $\mathbf{x} = \begin{pmatrix} x & y \end{pmatrix}^T$ are formed by the intersection of the 2D image plane with a ray linking camera lens and R.

If furthermore, the scales of the image and world coordinate systems match and

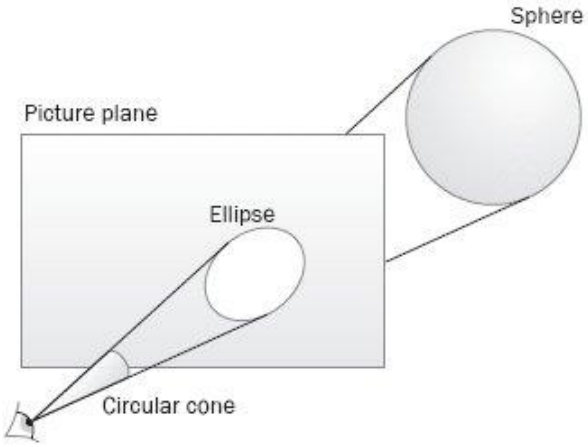


Figure 2.4: Projective Geometry

the focal length is f = 1, the projection of a world point onto image plane can be simplified into following equation:

$$\tilde{x} = \begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} X \\ Y \end{pmatrix}, \tag{2.1}$$

Generally in motion estimation, firstly, we select a parametric model, which represents our a priori knowledge about the camera movement. Then we use data to fit the model. A criterion is selected to measure how well the model represents the experimental data. The problem gets converted into an optimization problem where the best model is the one which fits better the data in the sense defined by a given criterion. An outlier method is also needed to filter noisy samples which will be discussed later in this chapter.

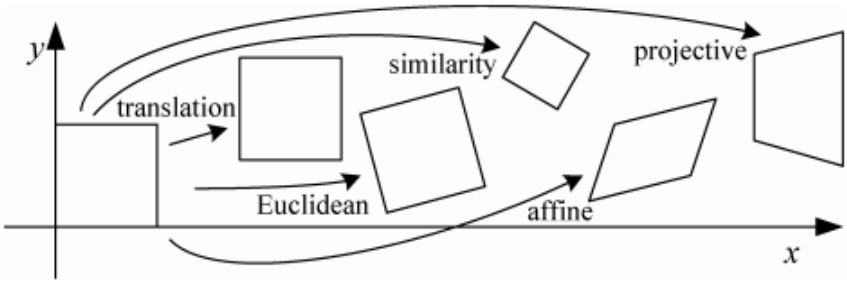


Figure 2.5: Basic set of 2D planar transformations (adapted from [1])

Motion Models establish the mathematical relationships for mapping pixel coordinates from one image to another. A variety of parametric motion models are

possible, from simple 2D transforms, to planar perspective models, 3D camera rotations, lens distortions, and the mapping to nonplanar (e.g., cylindrical) surfaces.

2.2.1 2D (planar) Motions

Having defined our coordinate system, we can now describe how coordinates are transformed. The simplest transformations occur in the 2D plane and are illustrated in Figure 2.5.

2.2.2 Hierarchy of 2D Transformation

Name	Matrix	Number of d.o.f.	Preserves	Icon
Translation	$[\boldsymbol{I}\|\boldsymbol{t}]_{2\times3}$	2	Orientation $+\cdots$	
Rigid (Euclidean)	$[\boldsymbol{R}\|\boldsymbol{t}]_{2\times3}$	3	Lengths $+\cdots$	
Similarity	$[s\boldsymbol{R}\|\boldsymbol{t}]_{2\times3}$	4	Angles $+\cdots$	
Affine	$[\boldsymbol{A}]_{2\times3}$	6	Parallelism $+\cdots$	
Projective	$[\tilde{\boldsymbol{H}}]_{3\times3}$	8	Straight lines	

Figure 2.6: Hierarchy of 2D coordinate transformations (adapted from [1])

1. **Translation:** 2D translations can be written as $\mathbf{x'} = \mathbf{x} + \mathbf{t}$ or

$$\mathbf{x'} = \begin{bmatrix}\mathbf{I} & \mathbf{t}\end{bmatrix}\tilde{x}, \tag{2.2}$$

   where $\mathbf{I}$ is the $(2 \text{ x } 2)$ identity matrix and $\tilde{\mathbf{x}} = (\text{x,y,1})$ is the *homogeneous* or *projective* 2D coordinate.

2. **Rotation + Translation:** This transformation is also known as *2D rigid body motion* or the *2D Euclidean Transformation* (since Euclidean distances are preserved). It can be written as $\mathbf{x'} = \mathbf{Rx} + \mathbf{t}$ or

$$\mathbf{x'} = \begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\tilde{x}, \tag{2.3}$$

where

$$\mathbf{R} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}, \tag{2.4}$$

is an orthonormal rotation matrix with $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ and $\mathbf{R} = 1$.

3. **Scaled Rotation:** Also known as the *similarity transform*, this transform can be expressed as $\mathbf{x'} = s\mathbf{R}\mathbf{x} + \mathbf{t}$, where s is an arbitrary scale factor. It can also be written as

$$\tilde{\mathbf{x}} = [s\mathbf{R} \quad \mathbf{t}]\,\tilde{\mathbf{x}} = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \end{bmatrix}\tilde{\mathbf{x}}, \tag{2.5}$$

where we no longer require that $a^2 + b^2 = 1$. The similarity transform preserves angles between lines.

4. **Affine:** The affine transform can be written as $\mathbf{x'} = \mathbf{A}\mathbf{x}$, where $\mathbf{A}$ is an arbitrary 2 x 3 matrix, i.e.,

$$\mathbf{x'} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix}\tilde{\mathbf{x}}. \tag{2.6}$$

Parallel lines remain parallel under affine transformations.

5. **Projective:** This transform, also known as a *perspective transform* or *homography*, operates on homogeneous coordinates $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}'$,

$$\tilde{\mathbf{x}}' \sim \tilde{\mathbf{H}}\tilde{\mathbf{x}}, \tag{2.7}$$

where $\sim$ denotes equality up to scale and $\tilde{\mathbf{H}}$ is an arbitrary 3 x 3 matrix. Note that $\tilde{\mathbf{H}}$ is homogeneous, i.e., it is only defined up to a scale. The resulting homogeneous coordinate $\tilde{\mathbf{x}}'$ must be normalized in order to obtain an inhomogeneous result $\tilde{\mathbf{x}}$, i.e.,

$$\tilde{\mathbf{x}}' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}} \quad and \quad \tilde{\mathbf{y}}' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}, \tag{2.8}$$

Perspective transformations preserve straight lines.

2.3 Robust Estimation - Dealing with Outliers

We need to refine the initial set of feature correspondences to find a set (called inlier set) which will produce a high accuracy alignment of the pair of images. So far these algorithms are only robust with respect to noise if the source of this noise is in the measurement of the correspondence feature positions. There will be other situations where the input will be corrupted with completely false correspondences, meaning that the two features in the images don't correspond to the same real world feature at all. This section discusses ways to distinguish inlier and outlier correspondences for robust estimation using only inlier matches.

2.3.1 Error Metrics

Once a suitable motion model is chosen to compare and describe the alignment between a pair of images, a method to estimate its parameters is needed. Error metrics for functions F and G (which can be images) with respect to displacement h.

- First Order Norm:

$$L_1(\mathbf{h}) = \sum |F(\mathbf{x}+\mathbf{h}) - G(\mathbf{x})|, \tag{2.9}$$

- Second Order Norm:

$$L_2(\mathbf{h}) = (\sum |F(\mathbf{x}+\mathbf{h}) - G(\mathbf{x})|^2)^{1/2}, \tag{2.10}$$

First order norm is called Sum of Absolute Difference (SAD) and second Sum of Squared Difference (SSD). We can make the above error metric more robust to outliers by using a function that grows less quickly than the quadratic penalty associated with least squares.

Outlier detection methods combined with suitable norms must be implemented to give robustness. A correspondence is considered to be an outlier if it lies outside the error threshold that defines maximum deviation attributable to effects of noise.

2.3.2 RANSAC

RANSAC (Random Sample Consensus) is an iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers, was first presented by Fischler and Boles (1981) [14]. RANSAC is the most commonly used robust estimation method for homographies according to [15].

It starts by selecting (at random) a subset of k correspondences, which is then used to compute a motion estimate $\boldsymbol{p}$. The *residuals* of the full set of correspondences are then computed as

$$\boldsymbol{r_i} = \tilde{\boldsymbol{x}}'_i(\boldsymbol{x_i}; \boldsymbol{p}) - \hat{\boldsymbol{x}}'_i, \tag{2.11}$$

where $\tilde{\mathbf{x}}'_\mathbf{i}$ are the *estimated* (mapped) locations, and $\hat{\mathbf{x}}'_\mathbf{i}$ are the sensed (detected) feature point locations.

The RANSAC technique then counts the number of *inliers* that are within ϵ of their predicted locations, i.e., whose

$$||\mathbf{r_i}|| \leq \epsilon$$

The ϵ value is application dependent, but often is around 1-3 pixels. The other robust method, Least Median of Squares (discussed in next subsection) finds the median value of $||\mathbf{r_i}||$.

The random selection process is repeated S times, and the sample set with the largest number of inliers (or with the smallest median residual) is kept as the final solution. Either the initial parameter guess $\boldsymbol{p}$ or the full set of computed inliers is then passed on to the next data fitting stage.

Advantages of RANSAC:

- RANSAC can estimate with a high degree of accuracy even when significant number of outliers are present in the dataset.

Limitations of RANSAC:

- There is no upper bound on the time it takes to compute the parameters.

- There exists a trade-off between number of iterations and accuracy of computed parameters.
- RANSAC requires setting of problem specific thresholds, e.g., distance threshold t for classifying correspondences as inliers or outliers.

2.3.3 Least Median of Squares Regression

Least median of squares regression (LMedS) works very well if there are less than 50% outliers and has the advantage over RANSAC that it requires no setting of threshold or a priori knowledge of how much error to expect. The major disadvantage of LMedS is that it is unable to cope with more than 50% of the data being outliers. In this case, the median distance would be to an outlier correspondence.

2.4 Desired Motion Estimation

Estimated motion between two frames consists of intended camera motion and unwanted jitter (noise). To stabilize video sequence, the unwanted jitter must be filtered out. It can be observed that unwanted jitter will be high frequency motion (noise) whereas intended camera motion corresponds to low frequency desired motion. Many techniques with increasing complexity have been developed to filter out unwanted jitter from estimated camera motion. Some of them are:

2.4.1 Low Pass Filter - Moving Average Filter

Assuming that intended motion is correlated within n consecutive frames and unwanted jitter may be correlated within up to m consecutive frames where n >> m. The set of indices of neighbouring frames is defined as

$$N_t = s|s \neq t, |s - t| < R, \tag{2.12}$$

where R is the range of the neighbourhood.

The motion model parameters can be simply averaged in the neighbourhood of the target frame to cancel out the random unwanted jitter (noise), thereby, preserving desired motion. The new computed parameters are used to generate the stabilized sequence.

Similarly, a Gaussian kernel centered about the target frame can be used for the weighted averaging purpose.

2.4.2 Motion Vector Integration

Motion Vector Integration (MVI) can be used to filter the cumulative motion curve, i.e., the motion of the current frame with respect to the first one. Cumulative motion vector is obtained as the sum of all previous interframe motion vectors, which can be integrated with a damping constant δ and the Integrated Motion Vector of frame n is generated as

$$IMV(n) = \delta * IMV(n-1) + GMV(n), \tag{2.13}$$

where $GMV(n)$ is the Global Motion Vector between frame n and $n-1$. In this way compensation $C(n)$ to be applied on frame n is obtained as

$$C(n) = IMV(n) - IMV(n-1), \tag{2.14}$$

and intentional motion is therefore smoothed and not compensated. The damping constant δ is chosen between 0 and 1, depending on the application. An *adaptive* damping constant δ can provide better solution and more flexibility. Its value can depend on the sum of last few Global Motion vectors, for if this sum is low the algorithm assumes an intentional static camera motion and chooses a high damping constant to strongly stabilize the sequence, whereas a high sum value indicates large motion and therefore a lower value of δ is chosen, following more closely the intentional motion. The filtering is applied independently on the x-translation, y-translation and angle values of the parametric motion model.

2.5 Literature Review

The methods for video stabilization presented in literature are based on different models. Following are the algorithms reported in papers. The number in [*] is the reference paper number, whose details are in bibliography.

In [16] block motion vectors were computed to estimate motion. The algorithm uses as motion estimator a module that produces block motion vectors and from these evaluates a vector identifying unwanted motions. This vector summarizes the motion of the frame and is called Global Motion Vector (GMV), which is then compensated.

In [17] the presented algorithm uses block matching technique for computation of local motion vectors. Some of the computed vectors are filtered out by making use of ad-hoc rules taking into account local similarity, local variance and matching effectiveness. Also a temporal analysis of the relative error computed at each frame has been achieved. Reliable local motion vectors are then used to compute inter-frame transformation parameters.

In [18] a technique for video stabilization based on particle filtering framework is used. Tracking of the projected affine model of the camera motions is performed using particle filters tracking. The inverse of resulting image transform is used to obtain a stable video sequence. The correspondence between SIFT points is used to obtain a crude estimate of the projected camera motion. Crude estimate is post-processed with particle filters to obtain a smooth estimate. It is shown both theoretically and experimentally that particle filtering can reduce the error variance compared to estimation without filtering.

In [19] motion inpainting was proposed to enforce spatial and temporal consistency of the completion in both static and dynamic image areas. Local motion is propagated from defined areas to missing areas and interpolation of sharper image pixels of neighbouring frames to increase the sharpness of the frame. Global motion estimation is performed by aligning pair-wise adjacent frames assuming a geometric transformation. However, it might fail when speedily-moving objects are in the

scene, and frame rates were not real-time.

[20] presents digital image stabilization algorithm with sub-image phase correlation based global motion estimation and Kalman filtering based motion correction. Global motion is estimated from the local motions of four sub-images, each local motion vector computed using phase correlation based motion estimation. The global motion vector corresponds to peak values of sub-image phase correlation surfaces, instead of impartial median filtering. The peak values of sub-image phase correlation surfaces reveals reliable local motion vectors, as poorly matches sub-images result in considerably lower peaks in the phase correlation surface due to spread. Fast implementation is made possible due to utilization of sub-images for phase correlation. The global motion vectors of image frames are accumulated to obtain global displacement vectors, that are Kalman filtered for stabilization.

In [21] a set of fast and robust electronic video stabilization algorithms are presented in this thesis. The first algorithm is based on a 2D feature-based motion estimation technique. The method tracks a small set of features and estimates the movement of the camera between consecutive frames. An affine motion model is utilized to determine the parameters of translation and rotation between images. The determined affine transformation is then exploited to compensate for the abrupt temporal Also, a Frequency domain approach is developed to estimate translations between two consecutive frames in a video sequence. Finally, a jitter detection technique has been developed to isolate vibration affected sub-sequences from an image sequence discontinuities of input image sequences.

In [22] motion was estimated using feature-based motion estimation employing SIFT features and local motion vectors are computed. The intentional motion was filtered using Adaptive fuzzy filter, Kalman filtering and motion vector integration to preserve panning. Finally, motion compensation is performed to produce stabilized video sequences. Moreover, a subjective visual perception scheme was used to rate the stabilized video quality on a scale of 1 (bad) to 9 (excellent) with respect to original videos.

In [23] a novel global estimation method based on the phase-correlation of central sub-image is presented. Five sub-images, 4 on corners and 1 in centre were individually used to compute global motion vectors. Results on videos taken from front mounted camera on car showed that central sub-image has the best performance.

In [24] Global motion was estimated using SURF as stable feature points to be tracked between the frames. Feature correspondences were filtered using bi-directional nearest neighbour distance ratio (NNDR) matching technique. An objective function to associate error with each feature point, based on median of local motion vectors, was used to discard local motion vectors corresponding to foreground moving objects. A discrete Kalman filter is used to smoothen the estimated motion vectors to obtain stabilized video sequence.

In [25] global motion estimation was performed using FAST keypoints detection and ORB feature descriptors matching. FAST combined with ORB gives real-time performance with very high accuracy. Affine transformation parameter were computed and refined using RANSAC and motion smoothing was performed using Gaussian filter.

In [26] some representative points were selected and matched to obtain motion vectors, which increases computation speed and accuracy of motion vectors. Median filter was used to get global motion vector. Jitter vector was estimated using temporal mean filtering of motion vectors and suitable compensation was applied to generate stabilized video stream.

[27] proposed a method based on the computation of optical flow between consecutive video frames and an affine model is adopted in conjunction to the optical flow field obtained to estimate objects or camera motions using Horn-Schunck algorithm. The estimated motion vectors are then smoothed using Savitzky-Golay filter to smooth and stabilize video sequences.

In [28] a method to directly stabilize a video without explicitly estimating camera motion was proposed. Optimization of extracted robust feature (SIFT) trajectories from the input video was performed to obtain a set of transformations to smooth

out these trajectories. In addition, the optimization also considers quality of the stabilized video and select a video with not only smooth camera motion but also less unfilled area after stabilization.

In [29] a novel video stabilization technique utilizing the human perception for analysing the rotation in terms of vertical displacement of two halves of the frame was proposed. Frame was partitioned into two halves and their corresponding motions were then utilized for respective rotational angle and vertical shift estimation.

In [30] motion block filtering methods were proposed to improve the computational speed and accuracy of global motion estimation. Firstly, Gradient information was analysed for the selection of reliable regions and global motion consistency was exploited to remove blocks corresponding to foreground motion. Now, local motion vectors of the selected robust regions were fitted using iterative least squares method to solve for global motion model parameters.

In [31] tracking of SIFT features trajectories is implemented to estimate inter-frame motions. A modified version of iterative least squares method is adopted to avoid estimation errors keeping track of cumulative error associated with a tracked feature. Intentional camera motion is filtered with adaptive motion vector integration technique to yield stabilized sequence.

In [32] motion estimation was performed using polar transform based circular block matching was used to estimate the global motion parameters, including rotation angle and translation values based on rigid motion model, utilizing hierarchical search strategy to speed up the operation. Estimated global motion parameters were then used to stabilize the input video.

In [33] sparse optical flow vectors were computed between successive frames, followed by estimating the camera motion by fitting the computed optical flow vectors to a simplified affine motion model with a robust trimmed least squares method. A regularization approach involving minimizing a cost function was utilized to temporally smooth the computed camera motion parameters in order to obtain stabilized video sequences.

[34] utilized feature matching between successive frames is used to obtain local motion vectors 3 times variance (3σ) criterion of normal distribution is used to discard local motion vectors. The Sage-Husa adaptive filtering method is used of intentional motion filtering and warping frames to obtain stabilized video frames.

In [35] an approach to separate unwanted vibrations from intentional camera motion based on a probabilistic estimation framework was proposed. Estimated parameters of inter-frame camera motion are treated as noisy observations of the intentional camera motion parameters and recursive Kalman filtering is used to smoothen the state-space model of these inter-frame motion parameters.

In [36] motion between successive frames was estimated using affine transform model with FAST corner detection and matching. Vibration compensation was performed based on spline smoothing of affine transform model parameters.

Chapter 3

Performance Analysis of Feature Matching

This chapter contains an exhaustive analysis of feature-based image registration methods and algorithms. Computational cost for various detectors and descriptor combination is evaluated in terms of processing time (section 3.1 & 3.2). Matching performance of detector-descriptor combination is evaluated in section 3.3. Finally, performance evaluation in terms of reduction of error metric of false match rejection techniques is performed in section 3.4.

3.1 Keypoints Detection

A detector analyses the frame to extract a set of salient keypoints, e.g., interest points that represent the most informative parts of an image. The detectors can be organized into two main classes, namely blob-like feature detectors and corner-like feature detectors. The keypoint detection processing time was evaluated. In general, the processing time depends on two factors:

- The number N of detected keypoints, which can be suitably varied by tuning the detection threshold of each detector
- The size of the input image

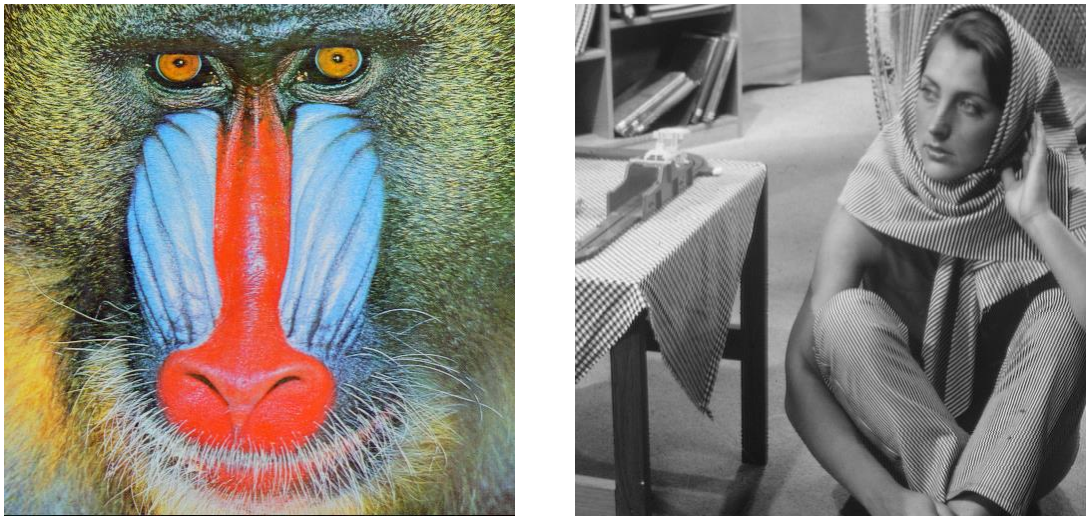


Figure 3.1: Test Images: baboon and barbara

Figure 3.2: Test Images: lenna and peppers

For the evaluation, 4 images (baboon, barbara, lenna and peppers) each of size 512 x 512 were used. For each detector, total time taken for keypoint detection and total number of detected keypoints was recorded.

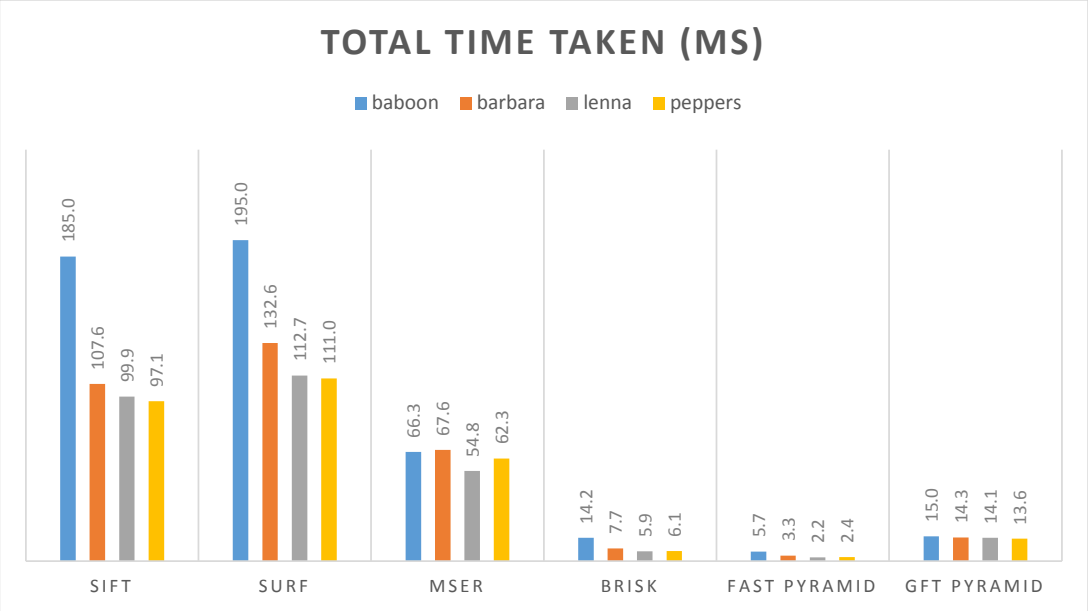


Figure 3.3: Total Time taken by various Keypoint Detectors (in ms)

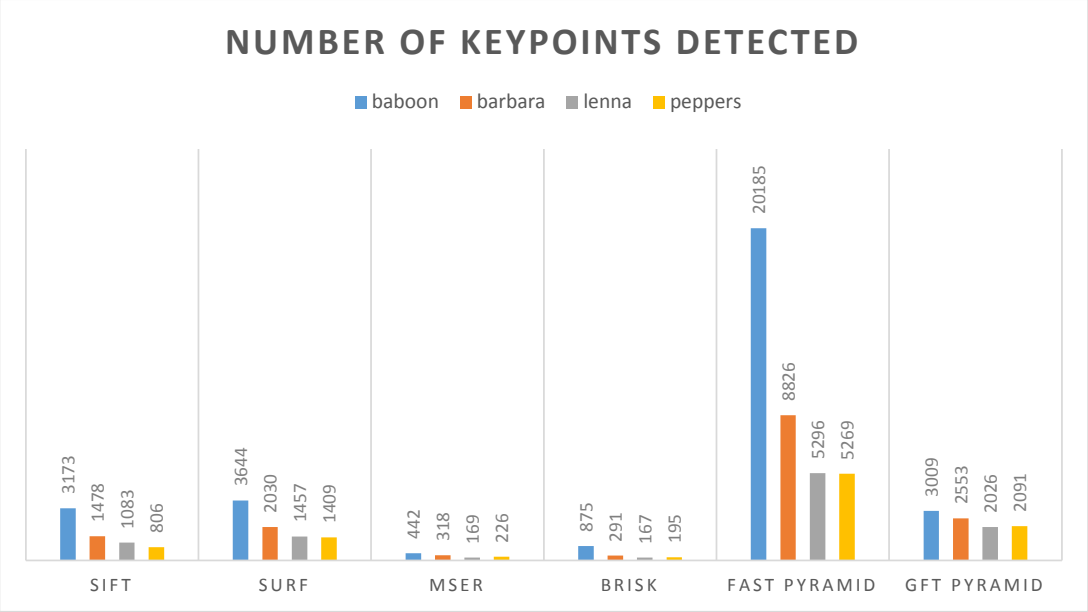


Figure 3.4: Number of Keypoints Detected

3.2 Descriptor Computation

Image patches around each keypoint are processed further and compactly represented by means of fixed-dimensional descriptors that capture, e.g., their photomet-

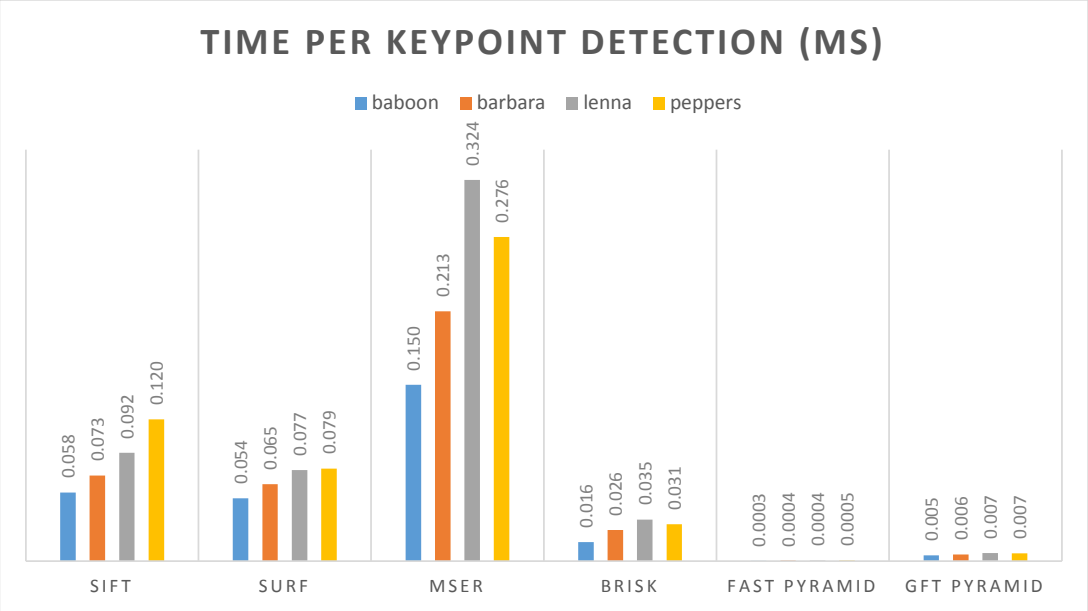


Figure 3.5: Time taken per Keypoint Detection (in ms)

ric properties. The descriptors can belong to either of two types, namely *non-binary descriptors* (SIFT, SURF) and *binary descriptors* (BRIEF, ORB, BRISK, FREAK). Binary descriptors are faster than non-binary descriptors because to build a binary descriptor it is only necessary to compare the intensity between two pixel positions located around the selected keypoint. This allows to obtain a representative description at very low computational cost.

Average processing time for computation of 500 descriptors was evaluated for six descriptor extractors for four test images (baboon, barbara, lenna & peppers). On the average, binary descriptor extractors were **20 times** faster than non-binary descriptor extractors.

Table 3.1: AVERAGE PROCESSING TIME OF VISUAL DESCRIPTOR ALGORITHMS FOR THE COMPUTATION OF 500 DESCRIPTORS.

Descriptor	Proc. Time (ms)
SIFT	61.50
SURF	23.50
BRIEF	2.34
ORB	2.01
BRISK	2.46
FREAK	46.93

3.3 Matching Accuracy

Brute-Force matcher is employed for feature matching. It takes the descriptor of one feature in the first set and is matched with all other features in the second set using some distance calculation (different for binary & non-binary descriptors) and the closest one is returned.

3.3.1 Evaluation Dataset

Comparative evaluation of the matching rates and mean error for matching was performed for various detector-descriptor combination. The various detector-descriptor combinations were evaluated on four image datasets with known ground-truth homography, subject to following transformation:

1. Rotation
2. Zoom
3. Rotation + Zoom
4. Viewpoint Change

Figure 3.6: Test Images: Rotation

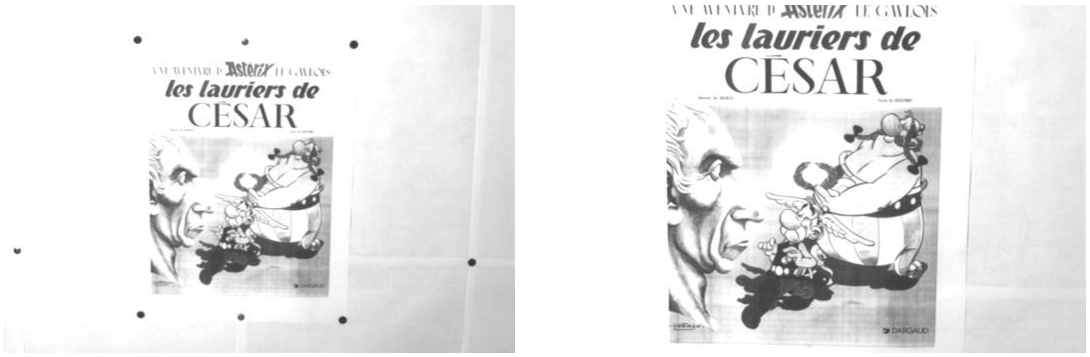


Figure 3.7: Test Images: Zoom

Figure 3.8: Test Images: Rotation + Zoom

Figure 3.9: Test Images: Viewpoint Change

3.3.2 Detectors-Descriptors Matching

Keypoint detectors which were not designed to be scale-invariant were pyramid adapted using OpenCV built-in functionality. The keypoint detectors used are:

1. Blob detectors
   (a) SIFT
   (b) SURF
   (c) MSER
2. Corner detectors
   (a) FAST
   (b) GFT
   (c) BRISK
   (d) ORB

Two types of descriptor extractors were used, namely, non-binary and binary. They are:

1. Non-binary
   (a) SIFT

(b) SURF

2. Binary

   (a) ORB

   (b) BRISK

   (c) BRIEF

Algorithm for matching is described as:

1. Determine keypoints in both images using a keypoint detector.

2. Compute feature descriptors using descriptor extractors for the keypoints from step 1.

3. Brute-Force feature descriptor matcher with suitable distance metric was chosen to get the list of correspondences between two images.

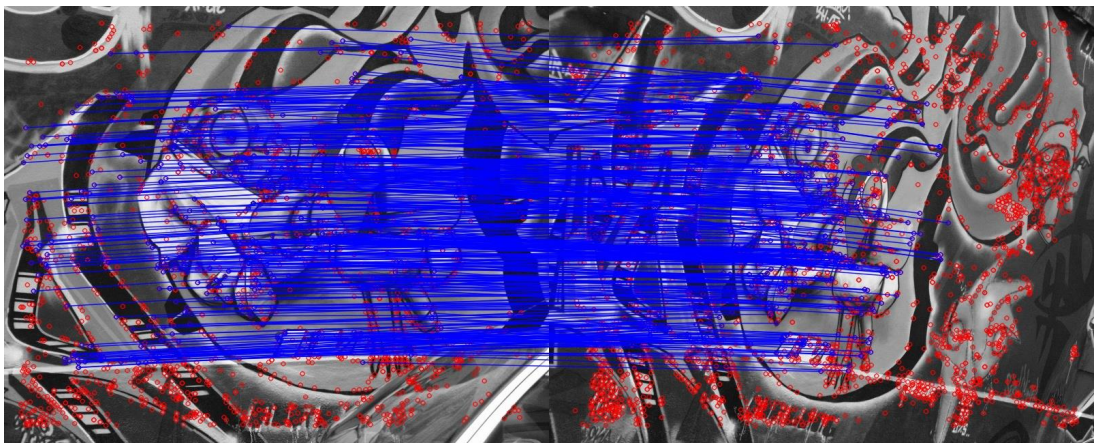


Figure 3.10: Example: Matched Keypoints

3.3.3 Matching Rate

Matching rates was evaluated for each detector-descriptor combination for each dataset. Computation of matching rate involves following steps:

1. Let the keypoints detected in 1st image be *points1* and 2nd image be *points2*.

2. We have the correspondences and the ground-truth homography relating two images. Ground-truth homography was used to forward transform *points1* into *points1t*.

3. Those matches were considered correct whose corresponding point in 2nd image was within a *threshold distance* of the ground-truth projected points.

$$||\boldsymbol{points2} - \boldsymbol{H_{gt}}.\boldsymbol{points1}|| < \boldsymbol{threshold}, \tag{3.1}$$

4. Matching rate was computed as ratio of *correct matches* to the *total number of matches*.

We chose a matching threshold of 2.0 pixels for classifying matches as inliers. ORB descriptor is most versatile. On an average, it performed the best with any detector for rotation, rotation+zoom and viewpoint change sequences. There was no clear winner for zoom sequence, although SIFT-SIFT (detector-descriptor) had the best matching rate.

In the plots 3.11 and 3.12, matching rate is shown on Y-axis and keypoint detectors on X-axis. Each colored line corresponds to a descriptor extractor as shown in the legend.

3.4 Evaluation of False Match Rejection Techniques

Descriptor matching can lead to false correspondences which must be filtered out to obtain a precise and consistent global motion vector. Techniques commonly employed for false correspondence rejection are:

1. *Cross-Check filtering:* Only those matches with value (i,j) such that i-th descriptor in set-A has j-th descriptor in set-B as the best match and vice-versa are considered as correct correspondences.

2. *Nearest Neighbour Distance Ratio (NNDR) filtering:* For each feature descriptor in set-A, best and second best match in set-B is computed. If the ratio of

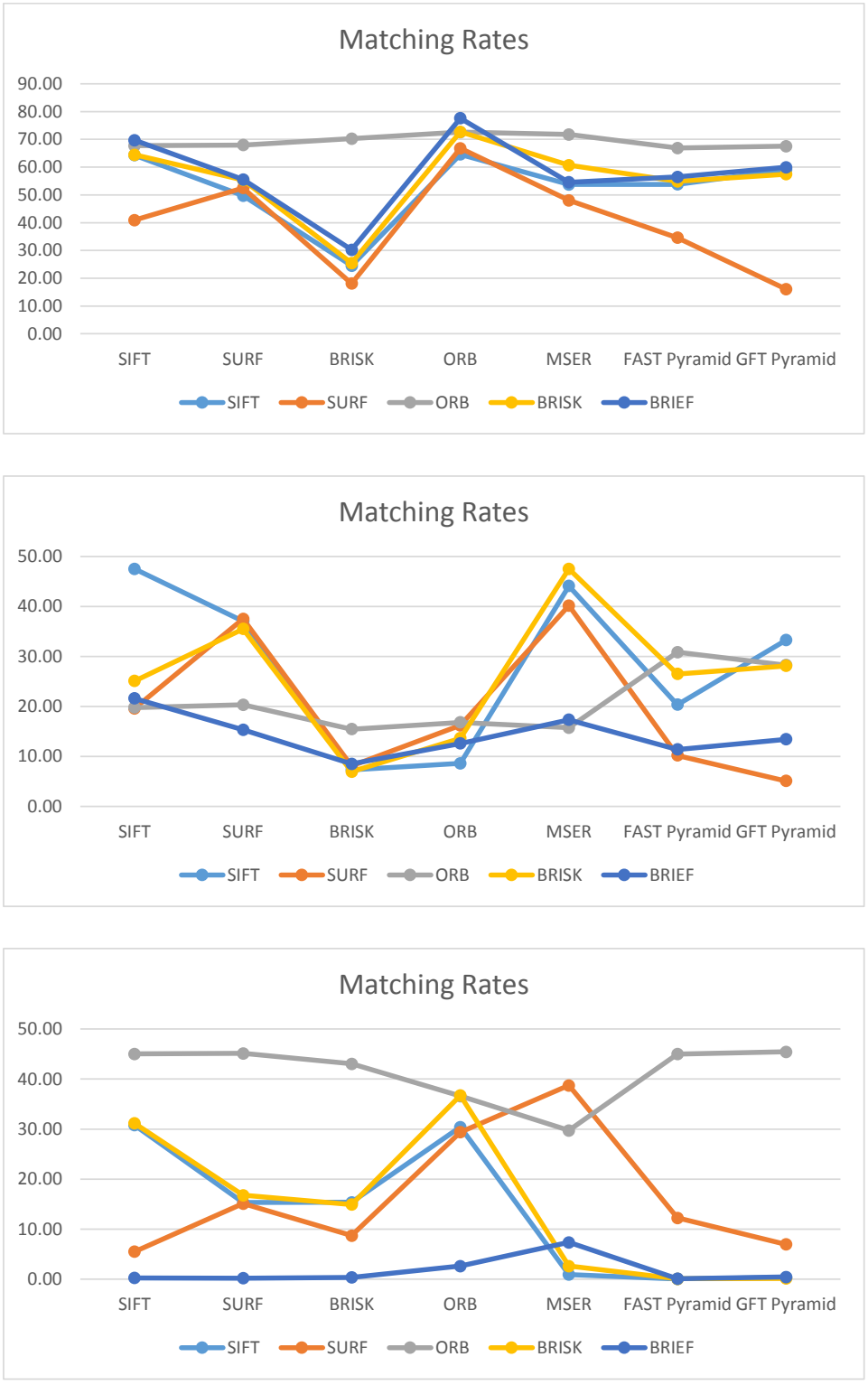


Figure 3.11: Matching Rates - from top to bottom: (1) Rotation Test (2) Zoom Test (3) Rotation + Zoom Test

distance of first best and second best correspondence is near 1, it implies that the match is false correspondence and not unique descriptor matching.

3. *Bi-Directional NNDR filtering:* The previous ratio test is performed in both

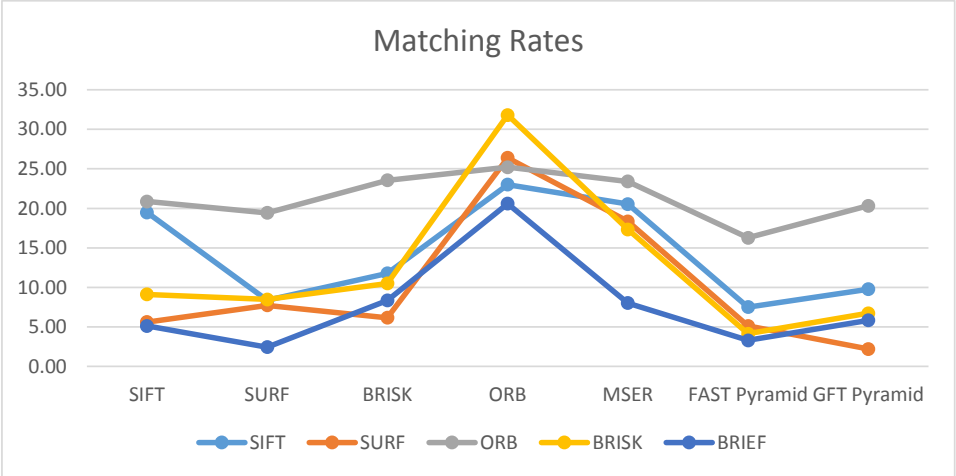


Figure 3.12: Matching Rates - (4) Viewpoint Change Test

directions (set-A to set-B and set-B to set-A) and only those correspondences which satisfy the threshold in both directions are considered to be good correspondences.

A distance ratio of **0.8** (as is in original Lowe's paper [6]) was used for NNDR and Bi-directional NNDR filtering. A comparative evaluation was performed on the reduction in average matching error with respect to ground-truth homography. *Average Matching Error* was evaluated for each detector-descriptor combination for each dataset. Computation of average matching error involves following steps:

1. Let the keypoints detected in 1st image be *points1* and 2nd image be *points2*.

2. We have the correspondences and the ground-truth homography relating two images. Ground-truth homography was used to forward transform *points1* into *points1t*.

3. Average matching error was evaluated for all the matched points as

$$\boldsymbol{Avg.MatchingError} = \sum ||\boldsymbol{points2} - \boldsymbol{H_{gt}.points1}||, \qquad (3.2)$$

The effectiveness of false matches rejection techniques can be demonstrated by plotting average matching error vs. techniques used for a given detector-descriptor combination. Average matching error was computed for all 4 image datasets (same

datasets used for evaluating matching rate as in section 3.3) for all permutations of 7 detectors & 5 descriptor extractors.

In Figure 3.14, simple matching results in a large number of mismatches (false correspondences) as visible from inter-cutting matched line segments. Number of mismatches is reduced somewhat using cross-check filter but not to an acceptable limit. There are less than 10 mismatches using NNDR filtering while bi-directional NNDR filtering reduces this number to zero since there are no intersecting matches line segments.

In Figure 3.15, it is visible that average matching error for all matches decreases as we move from simple matching to cross-check filtering to NNDR filtering to bi-directional NNDR filtering. This implies that these techniques (NNDR and bi-directional NNDR filtering) are highly effective in rejecting ambiguous or false matches.

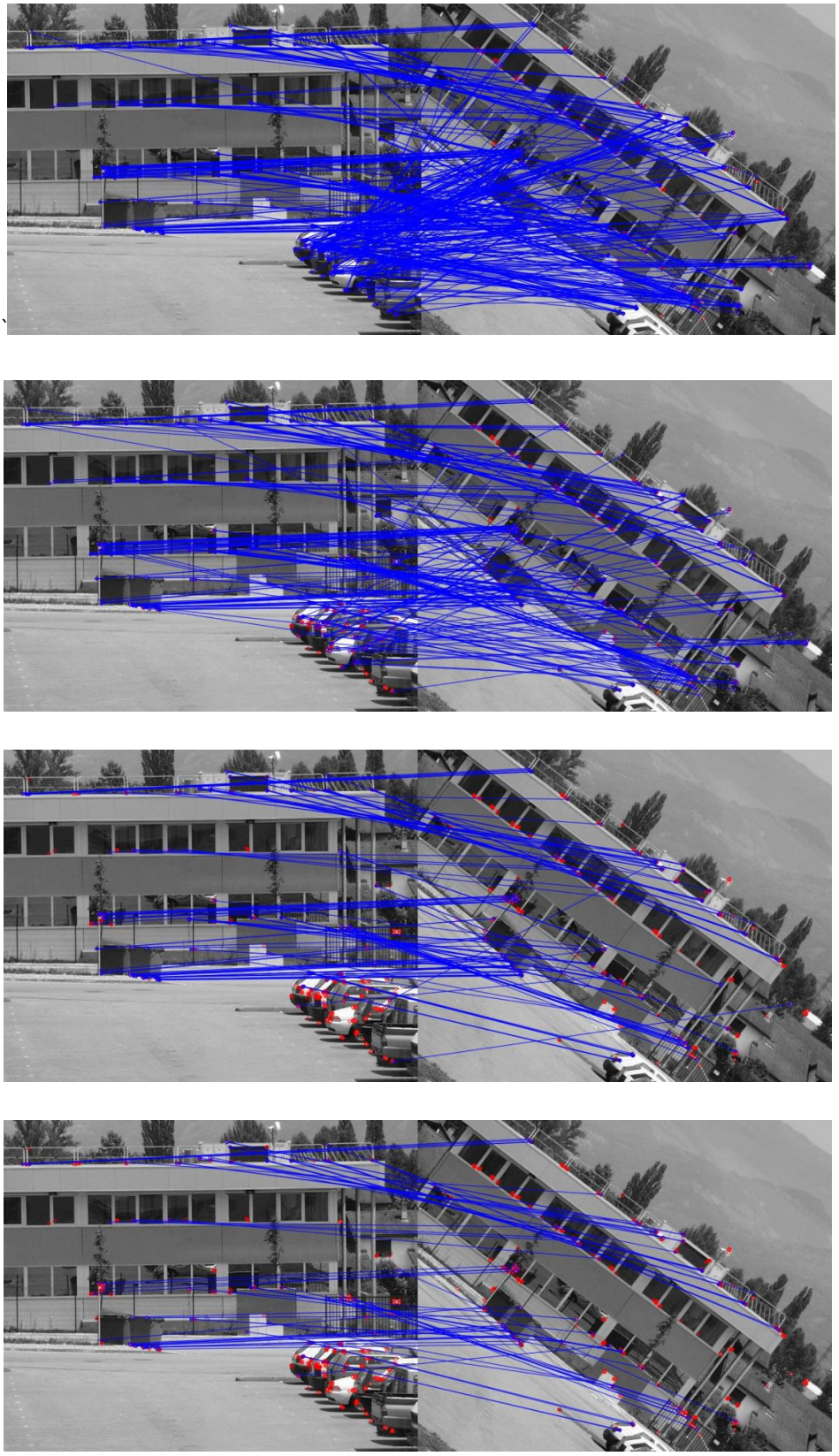


Figure 3.13: False Matches Rejection - from top to bottom: (1) Simple Matching (2) Cross-Check filtering (3) NNDR filtering (4) Bi-Directional NNDR filtering

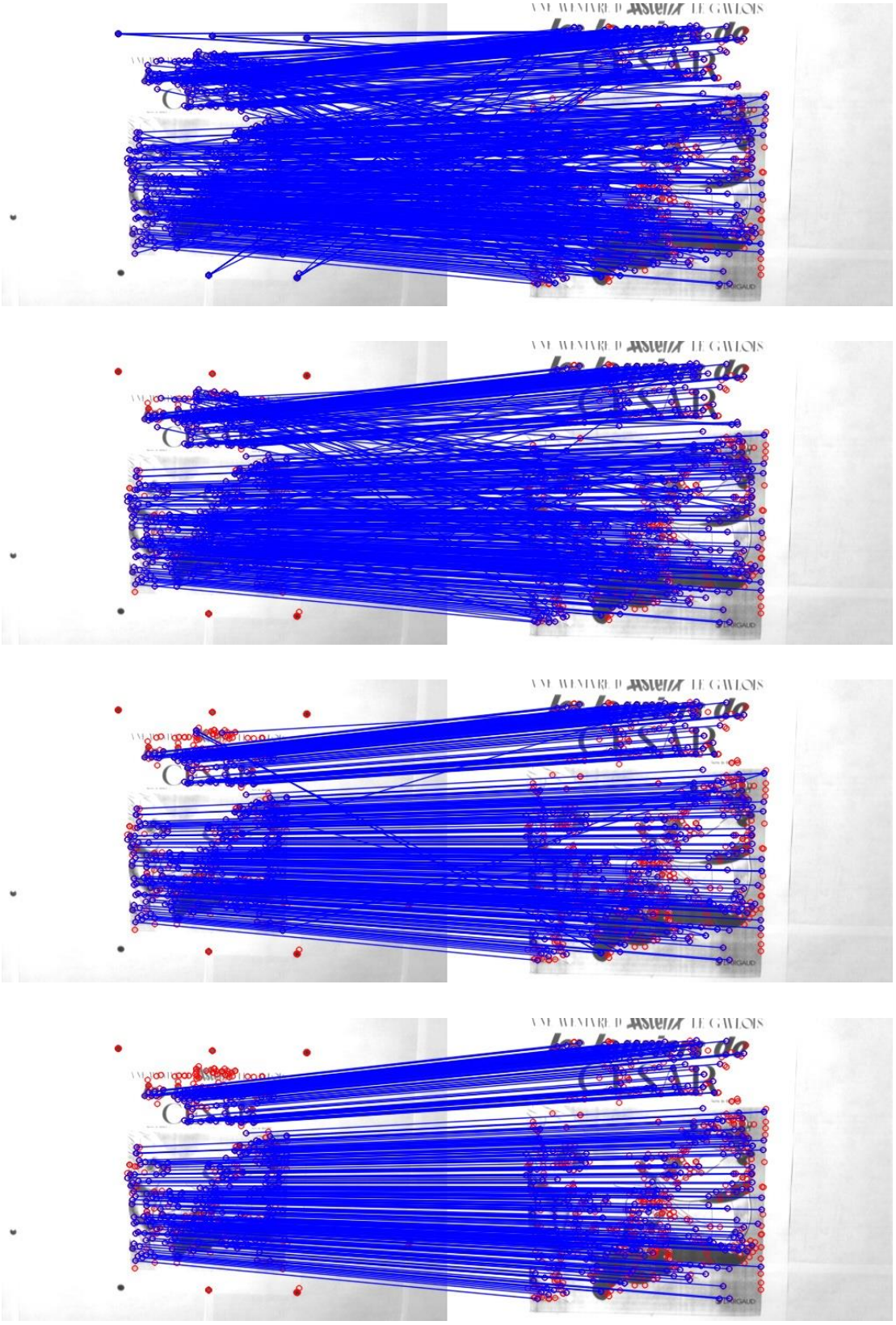
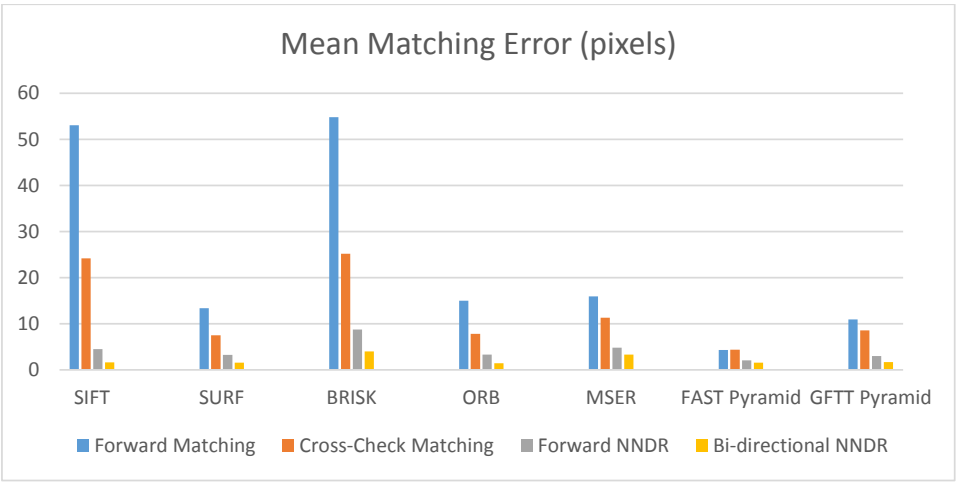
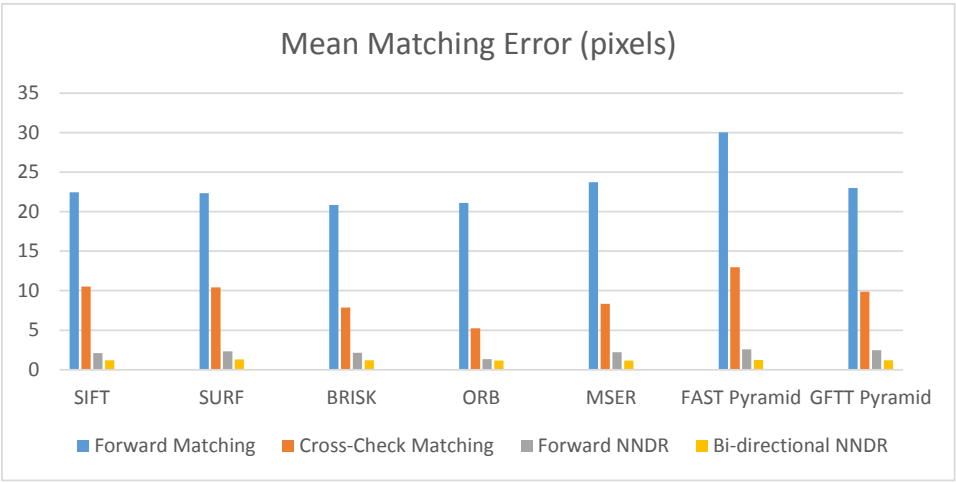


Figure 3.14: False Matches Rejection - from top to bottom: (1) Simple Matching (2) Cross-Check filtering (3) NNDR filtering (4) Bi-Directional NNDR filtering





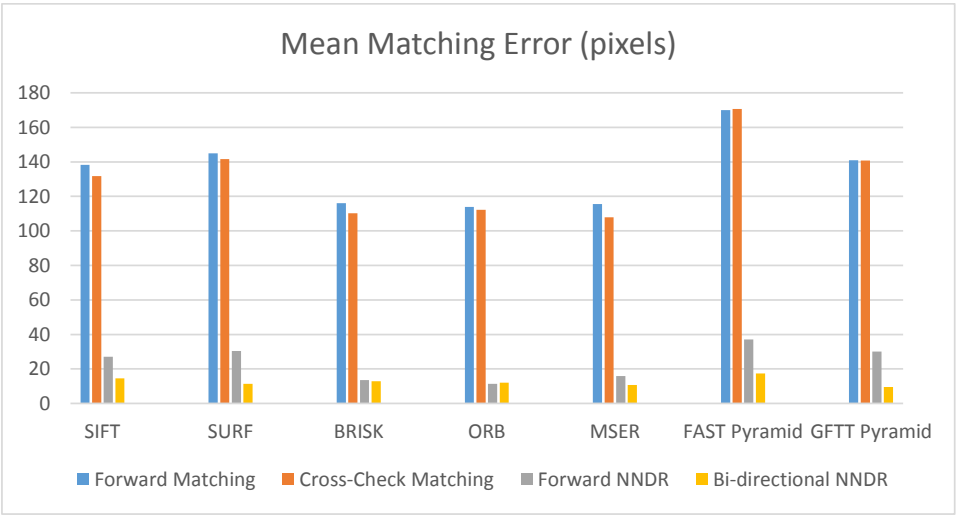


Figure 3.15: False Matches Rejection: Average Matching Error (pixels) vs Rejection Technique

Chapter 4

Proposed Solution and Results

4.1 Real-Time Constraint

In order to obtain correspondences between two frames, we had to choose between feature matching and feature tracking techniques. We had target of real-time processing rates for *720p* (1280 x 720) video sequences, i.e., a processing time of less than or equal to 33 ms per frame.

In order to compare performance of these two techniques, we chose the fastest detector (FAST keypoints detector), fastest descriptor extractor (ORB descriptor extractor) and bi-directional NNDR matching for rejecting false correspondences for *Feature Matching* and fastest detector (FAST keypoints detector) followed by pyramidal KLT tracker based on optical flow for *Feature Tracking*.

A comparative evaluation in terms of processing time of feature matching vs feature tracking technique was performed as shown in the following table:

Table 4.1: Processing Time Comparison (without GPU): Feature Matching vs. Feature Tracking (ms)

Technique	**Feature Matching**				**Feature Tracking**		
Frame Size	FAST	ORB	Bi-NNDR	TOTAL	FAST	Optical Flow	TOTAL
VGA	1.13	1.96	2.12	**5.21**	1.15	4.82	**5.97**
720p	1.27	1.97	2.12	**5.36**	1.40	6.48	**7.88**

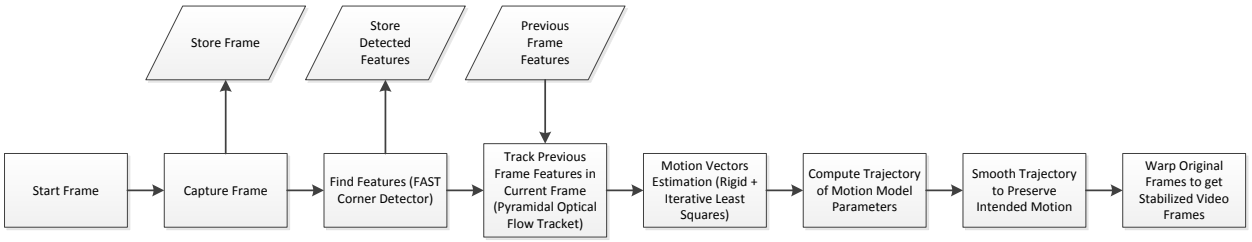
Table 4.2: Processing Time Comparison (with GPU): Feature Matching vs. Feature Tracking (ms)

Technique	**Feature Matching**				**Feature Tracking**		
Frame Size	FAST	ORB	Bi-NNDR	TOTAL	FAST	Optical Flow	TOTAL
VGA	1.13	1.96	2.12	**5.21**	1.15	1.40	**2.55**
720p	1.27	1.97	2.12	**5.36**	1.40	1.63	**3.03**

Since we were using GPU for speedup, feature tracking was slightly faster than feature matching, which we used to develop our algorithm.

4.2 Base Stabilization Algorithm

Figure 4.1: Base Stabilization Algorithm



Algorithm 1 PSUEDO CODE

for $i = 1$ to $TotalNumberofFrames - 1$ **do**
    Compute Features (frame[i])
    Track Features (frame[i+1])
    Estimate Transform ($H_{i,i+1}$)
    Compute and Store Trajectory (x_i, y_i, a_i)
end for
for $i = 1$ to $Size(trajectory)$ **do**
    Smooth Trajectory (Moving Average Filter)
end for
for $i = 1$ to $TotalNumberofFrames - 1$ **do**
    Compute Smoothed Transforms ($H^{new}_{i,i+1}$)
    Warp Original Frames
    Display Stabilized Frames
end for

4.2.1 Motion Estimation

Feature Tracking estimates the motion between frames by selecting features from the first frame and tracking those features in the second frame, so features that can be accurately tracked must be selected. A rigid transformation relates the locations of accurately tracked features, which is decomposed into x-component, y-component and angle-component relating the two image frames.

FAST [9] corner detector was used to select features. It is the fastest (see Figure 3.5) feature detector available in the literature, therefore highly suited for real-time applications. Threshold for FAST detector was varied with frame size to limit number of detected features and therefore reducing computational cost. Feature points near frame boundaries were masked out as they are most likely to disappear in the next frame due to motion. From these selected features, 200 strongest features were tracked in the current frame using Pyramidal Optical Flow Technique.

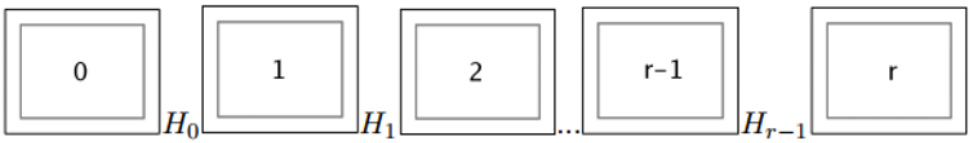


Figure 4.2: Transformations between Frames

Features which were accurately tracked were used to estimate the transformation between the two frames assuming a Rigid Transformation Model consisting of x-translation, y-translation and angle components. The x-, y- and angle-component trajectories are estimated by summing up the consecutive frame components.

OpenCV routines used:

1. *VideoCapture*
2. *cvtColor()*
3. *FastFeatureDetector()*
4. *calcOpticalFlowPyrLK()*
5. *estimateRigidTransform()*

4.2.2 Motion Filtering

To keep computational complexity low, a non-causal non-weighted moving average filter was used to smooth the x-, y- and angle-component trajectories in order to preserve intended motion. The radius of the filter determines the magnitude of stabilization, larger the radius, greater the stabilization and therefore algorithm less sensitive to intentional camera panning or translations. After experimenting with a number of test videos, a radius of $R = 20$ frames was selected as suitable for smoothing trajectories.

$$x_{new}[i] = \sum_{k=i-R}^{i+R} x[k], \tag{4.1}$$

$$y_{new}[i] = \sum_{k=i-R}^{i+R} y[k], \tag{4.2}$$

$$a_{new}[i] = \sum_{k=i-R}^{i+R} a[k], \tag{4.3}$$

where, R is the radius of moving average filter.

4.2.3 Motion Compensation

New frame-to-frame transforms were computed using smoothed trajectories of x-, y- and angle-components. Computed frame-to-frame transforms were used to warp original video frames to produce stabilized video frames, with preserved camera intentional motion and removed unwanted jitter.

OpenCV routines used:

1. *VideoCapture*
2. *warpAffine*
3. *resize()*
4. *VideoWriter*

4.3 Timing Code

Test System

Our algorithm development system consisted of:

- CPU: Intel Core i7-2600 @ 3.40 GHz
- GPU: Nvidia Tesla K20
- OS: Ubuntu 12.04 LTS
- OpenCV Version: 2.4.6
- CUDA version: 6.0

C++ code was profiled section-wise to gain insights for code optimization to achieve real-time performance. The evaluation was performed for two video frame sizes, *VGA*(640 x 480) and *720p*(1280 x 720). The timings for processing a single frame for the algorithm sections are as follows:

Motion Estimation

Table 4.3: Average Processing Time (ms) of various OpenCV Routines

OpenCV Routine	VGA	720p
VideoCapture	1.32	5.86
cvtColor()	0.46	1.50
FastFeatureDetector()	1.15	1.40
calcOpticalFlowPyrLK()	4.82	6.48
estimateRigidTransform()	0.014	0.023
TOTAL	**7.764**	**15.263**

Motion Filtering

Motion filtering section was least computationally intensive and took **0.015 ms** (*VGA*) and **0.124 ms** (*720p*).

Motion Compensation

Table 4.4: Average Processing Time (ms) of various OpenCV Routines

OpenCV Routine	VGA	720p
VideoCapture	0.45	1.08
warpAffine()	2.70	8.16
resize()	1.92	6.01
VideoWriter	2.98	9.45
TOTAL	**8.05**	**24.7**

Execution Time

Profiling code shows that most time consuming code sections are Motion Estimation and Motion Compensation, as shown in the table:

Table 4.5: Section-wise Processing Time (ms) per frame

Section	VGA	720p
Motion Estimation	7.764	15.263
Motion Filtering	0.015	0.124
Motion Compensation	8.05	24.7
TOTAl	**15.829**	**40.087**

4.4 GPU Optimization

OpenCV has a GPU (graphics processing unit) module to speed-up computationally intensive functions by utilizing immense parallel processing capabilities of GPU.

4.4.1 Nvidia CUDA

To be able to accomplish our goal of stabilizing a *720p* video that can process each frame in 33 ms, we needed to parallelize the computations. CUDA is a platform for parallel computing and programming model developed by NVIDIA and is often used for graphics and computer vision programming as it is able to run thousands of lightweight threads at the same time.

Most time consuming functions of our code were evaluated on both CPU and GPU to determine the speedup factor, as shown in the table:

Table 4.6: Processing Time Comparison: CPU vs. GPU (ms)

Frame Size	**VGA**			**720p**		
OpenCV Routines	CPU	GPU	Savings	CPU	GPU	Savings
cvtColor(BGR to GRAY)	0.46	2.58	-2.12	1.50	5.30	-3.80
FastFeatureDetector(threshold=20)	1.15	2.78	-1.63	6.29	6.63	-0.34
calcOpticalFlowPyrLK(200 points)	4.82	1.40	3.42	6.48	1.63	4.85
warpAffine()	2.70	2.94	-0.24	8.16	6.31	1.85

As per the results tabulated, GPU provides significant computation gain only in the case of *calcOpticalFlowPyrLK()* i.e. optical flow tracking of feature points. Therefore, optimizing with GPU reduces computational cost of stabilizing *VGA* video to **12.409 ms** (down from 15.829 ms) and *720p* video to **35.237 ms** (down from 40.087 ms).

OpenCV GPU function for *VideoCapture* and *VideoWriter* are under development and hence could not be tested for cost savings.

Despite taking advantage of GPU, our code was unable to give real-time performance as per frame computation time was **35.237 ms** (28 FPS) for *720p* video. So, there was still scope of improvement, which will be discussed in next section.

4.5 CPU Optimization

4.5.1 Pipelining

Pipelining is a technique used in design of algorithms to increase their throughput (the number of instructions that can be executed in unit time). Rather than processing each instruction sequentially (one at a time, finishing one instruction before starting next), each instruction step can be executed concurrently (by different processor cores), and indeed in parallel (at the same time).

Pipelining increases instruction throughput by performing multiple operations at the same time (in parallel), but does not reduce instruction latency (the time taken to complete a single instruction from start to finish) as it must go through all the steps. We employed a pipeline in our motion compensation section to reduce the time taken to process a single frame.

Table 4.7: Pipeline Execution: how a frame moves through the pipeline

Input Frame Number	VideoCapture	warpAffine()	resize()	VideoWriter
1	1			
2	2	1		
3	3	2	1	
4	4	3	2	1
5	5	4	3	2
6	6	5	4	3
7	7	6	5	4

4.5.2 OpenMP

C++ codes are sequential, therefore are not able to exploit full computational power of modern multi-core processors. *OpenMP* (Open Multi-Processing) is an Applica-

tion Programming Interface (API) that consists of a set of compiler directives, library routines and environment variables that allow C++ programs to take full advantage of multi-core processors to speed-up computation by running code on different cores of processor in parallel.

OpenMP library was used to run sequential functions in Motion Compensation section in parallel in order to reduce computation time per frame. Block diagram of sequential code of Motion Compensation section uses following OpenCV routines:

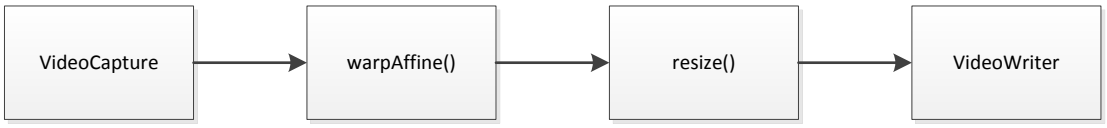


Figure 4.3: Motion Compensation Execution

Let us assume that above OpenCV routines take time a, b, c & d ms starting from left per processed frame. By default the routines run sequentially and therefore sequential execution time ($T_{sequential}$):

$$T_{sequential} = (a + b + c + d)ms, \tag{4.4}$$

The idea is to run each routine in parallel on single core of processor, therefore resulting in parallel execution time ($T_{parallel}$):

$$T_{parallel} = MAX(a, b, c, d)ms, \tag{4.5}$$

But due to OpenCV library's limitations, routine *warpAffine()* cannot be executed in parallel on a single core, therefore time taken after optimization ($T_{optimal}$):

$$T_{optimal} = b + MAX(a, c, d)ms, \tag{4.6}$$

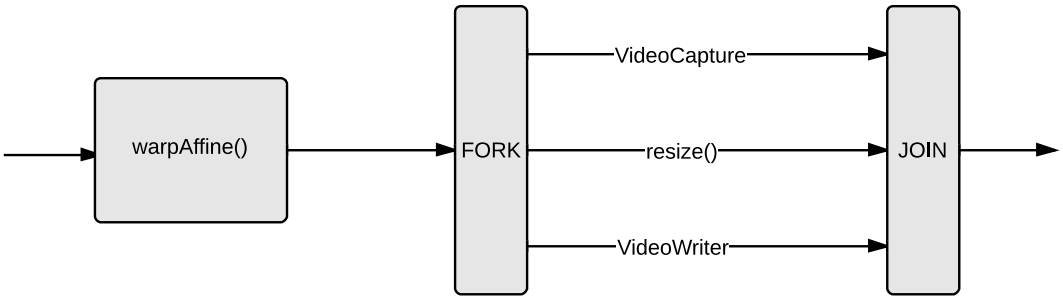


Figure 4.4: Motion Compensation Execution: Parallel on CPU

Despite OpenCV library's limitations, making routines run in parallel forming a pipeline resulted in enough computational cost savings to make our algorithm real-time for *720p* video as shown in table:

Table 4.8: Processing Time Comparison: Sequential vs. Parallel CPU (ms)

Frame Size	**VGA**		**720p**	
Methodology	Sequential	Parallel	Sequential	Parallel
VideoCapture	0.45	1.70	1.08	2.40
warpAffine()	2.70	3.25	8.16	8.34
resize()	1.92	2.95	6.01	7.97
VideoWriter	2.98	4.26	9.45	11.87
TOTAL	**8.05**	**9.26**	**24.70**	**20.21**

Making code parallel (i.e. SEQUENTIAL(warpAffine(), PARALLEL(VideoCapture, resize(), VideoWriter)) resulted in savings of **4.49 ms** for *720p* video but increased the execution time of *VGA* video by **1.21 ms**. Therefore final computational cost breakdown of our algorithm was:

Table 4.9: Section-wise Processing Time (ms) per frame

Section	**VGA**		**720p**	
Processing	CPU	GPU	CPU	GPU
Motion Estimation	7.764	4.344	15.263	10.413
Motion Filtering	0.015	0.015	0.124	0.124
Motion Compensation	9.26	9.26	20.21	20.21
TOTAL	**17.039**	**13.619**	**35.597**	**30.747**
Frame Rate (FPS)	58.69	73.43	28.09	32.52

Therefore, our algorithm gave frame rate of **73 FPS** for *VGA* video and **32 FPS** for *720p* video sequences.

4.6 Stabilization Results

A dataset of 20 unstable videos was used for evaluating the performance of the proposed algorithm. Comparing *Fidelity* of original and stabilized videos has been widely used to evaluate the efficiency of stabilization algorithm. *Fidelity* is a measure of how well stabilization is compensating the motion of the camera, i.e., how precisely the motion model fits the actual camera motion.

Background Motion Compensation

Ideally, video stabilization should compensate background layer motion completely while preserving foreground object motions. Difference of adjacent frames in original and stabilized video sequences is shown in the following images (Figures 4.5 & 4.6) to give an estimate of residual motion between frames.

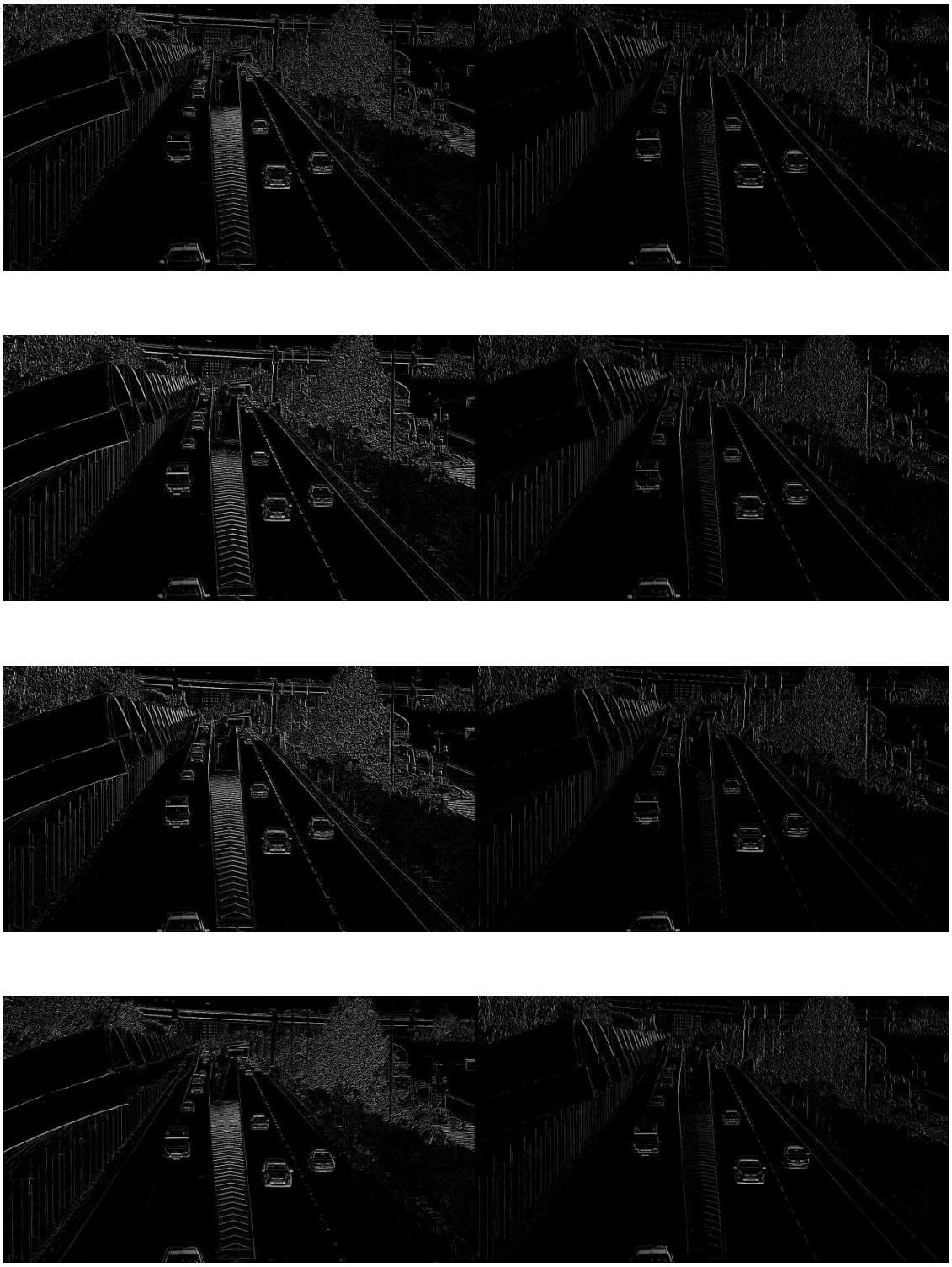


Figure 4.5: Sequence 1: Left half represents original sequence and Right half stabilized sequence

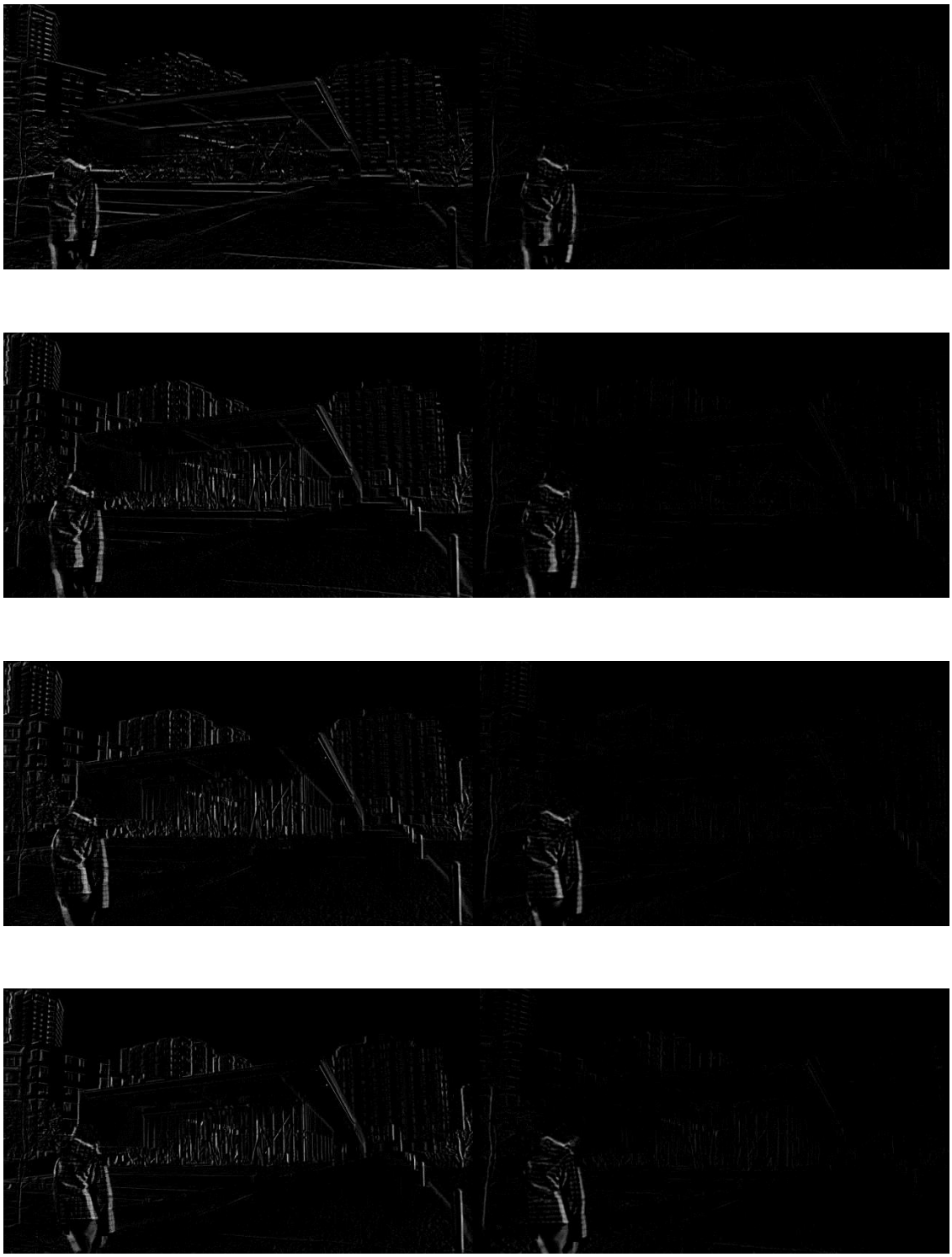


Figure 4.6: Sequence 7: Left half represents original sequence and Right half stabilized sequence

Fidelity

Ideally, when all the motion is compensated for, there should be no residual motion after stabilization, i.e., for every pixel, difference between the two stabilized images should be zero. However due to image noise and distortions arising from limitations of motion model and interpolation during image warping, lead to non-zero differences.

The PSNR between frames is used to measure the fidelity of a system. The PSNR between I_1 and I_0 is defined as:

$$PSNR(I_1, I_0) = 10log\frac{255^2}{MSE(I_1, I_0)}, \tag{4.7}$$

where the mean squared error (MSE) is a measure of average departure per pixel from the desired stabilization result. The higher the PSNR between two stabilized frames, the better the fidelity of the system.

Therefore given a particular stabilization system, **inter-frame transformation fidelity (ITF)** in dB measure is computed for original and stabilized frames, and the gain in ITF determines the superiority of the stabilization system.

Deshaker

Deshaker is a freely available most widely used commercial video stabilizer which can eliminate camera shakiness and makes panning, rotation and zooming smoother. Our methods performance was compared with Deshaker in terms of ITF measure and Time taken to stabilize a video. Mean values of ITF (in dB) for original sequence, sequence stabilized by Deshaker and sequence stabilized by our algorithm are as follows:

Table 4.10: Mean ITF values (in dB)

Video Sequence	Original	Deshaker	Proposed Algorithm
1	13.16	14.48	16.93
2	14.07	14.93	16.96
3	15.07	16.33	18.46
4	16.37	17.30	18.21
5	14.62	16.62	17.55
6	15.83	17.88	18.98
7	20.64	23.99	28.77
8	23.48	26.09	26.46
9	24.43	26.85	28.69
10	19.36	26.56	39.87
11	18.17	22.08	25.22
12	21.68	21.33	23.98
13	19.70	20.58	23.86
14	21.37	20.24	22.68
15	17.78	18.24	19.21
16	24.16	23.82	23.91
17	25.74	26.47	30.97
18	21.59	24.11	25.67
19	20.10	25.25	29.50
20	18.19	19.10	20.45

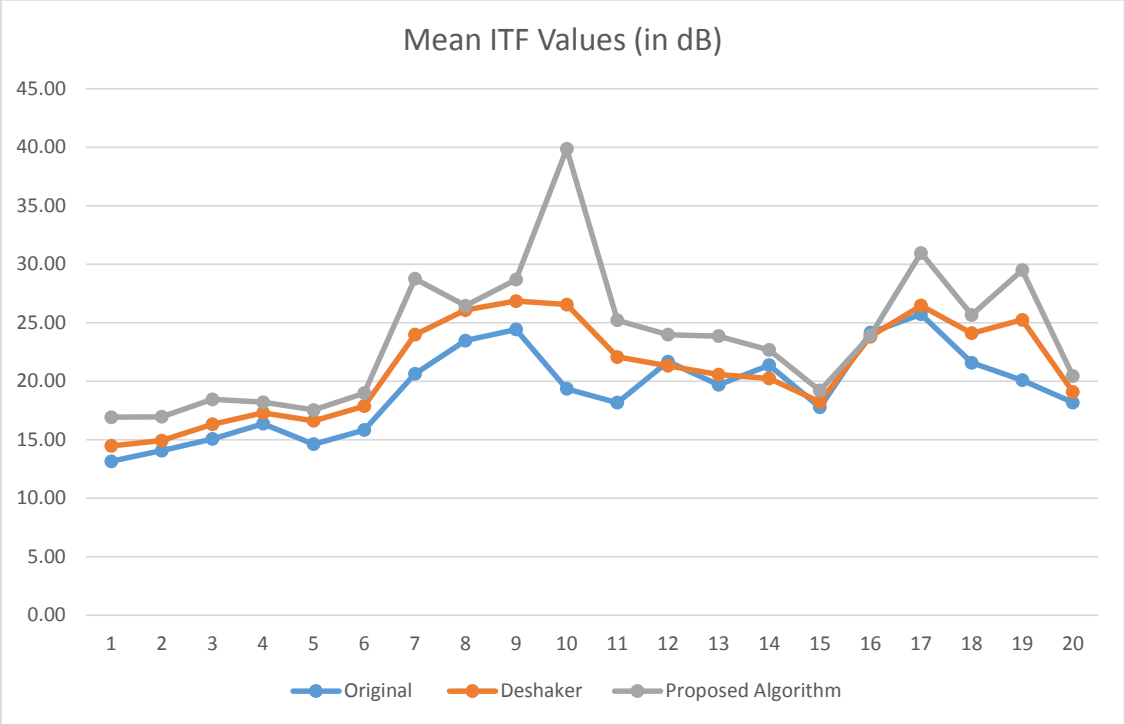


Figure 4.7: Mean ITF Values (in dB)

Processing Time

Table 4.11: Frames Per Second: Deshaker vs. Proposed Algorithm (FPS)

Method	**Deshaker**		**Proposed**	
Frame Size	32 bit	64 bit	CPU	GPU
VGA	43.81	53.16	58.69	73.43
720p	13.79	17.75	28.09	32.52

Mean ITF values and processing time per frame comparison with Deshaker shows the superiority of our stabilization algorithm. Few plots have been included to demonstrate the effect of stabilization on x-, y- and angle-component trajectories and ITF value per frame for some of the video sequences. These plots demonstrate the effectiveness of our algorithm in rejecting unwanted jitter and improving the video sequence quality as visible from the gain in inter-frame fidelity values.

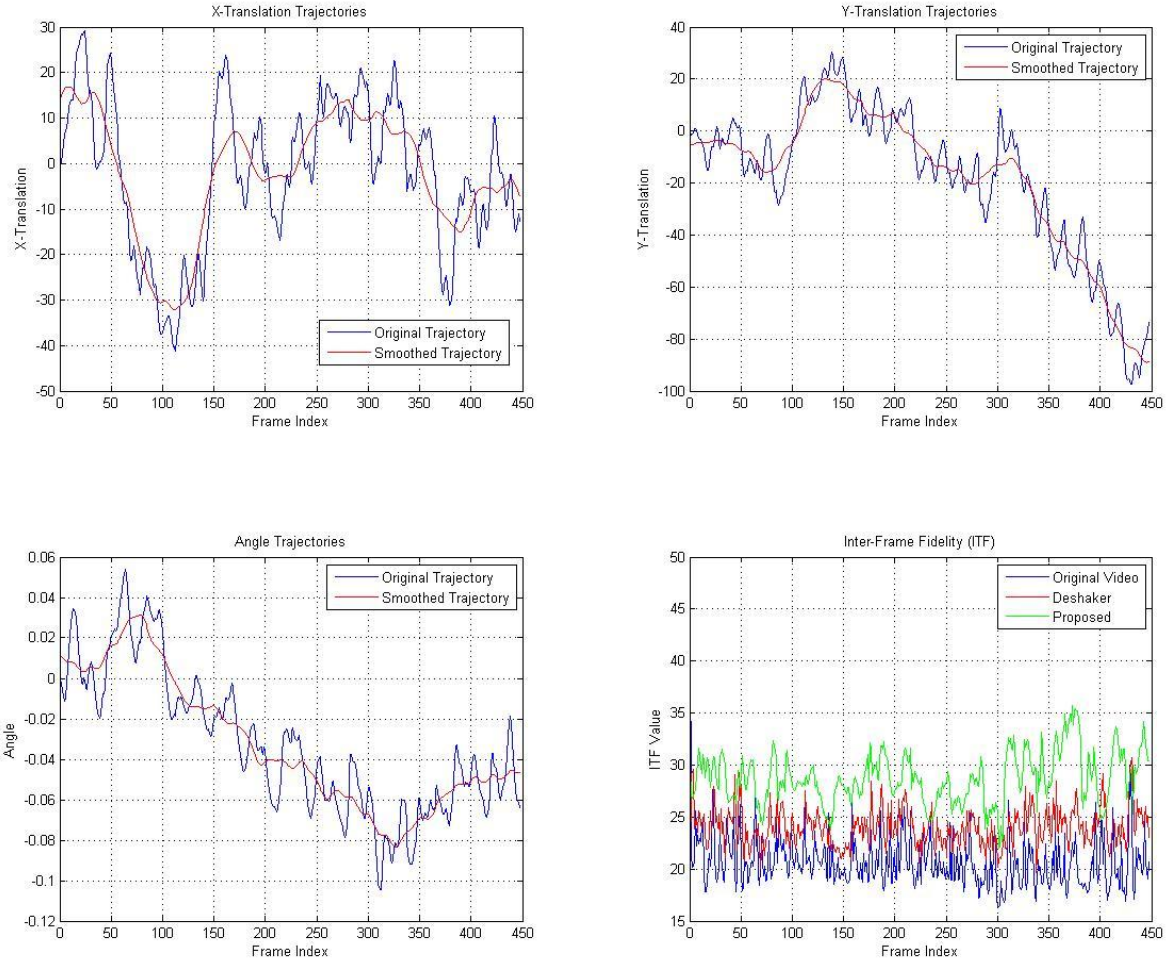


Figure 4.8: Sequence 7

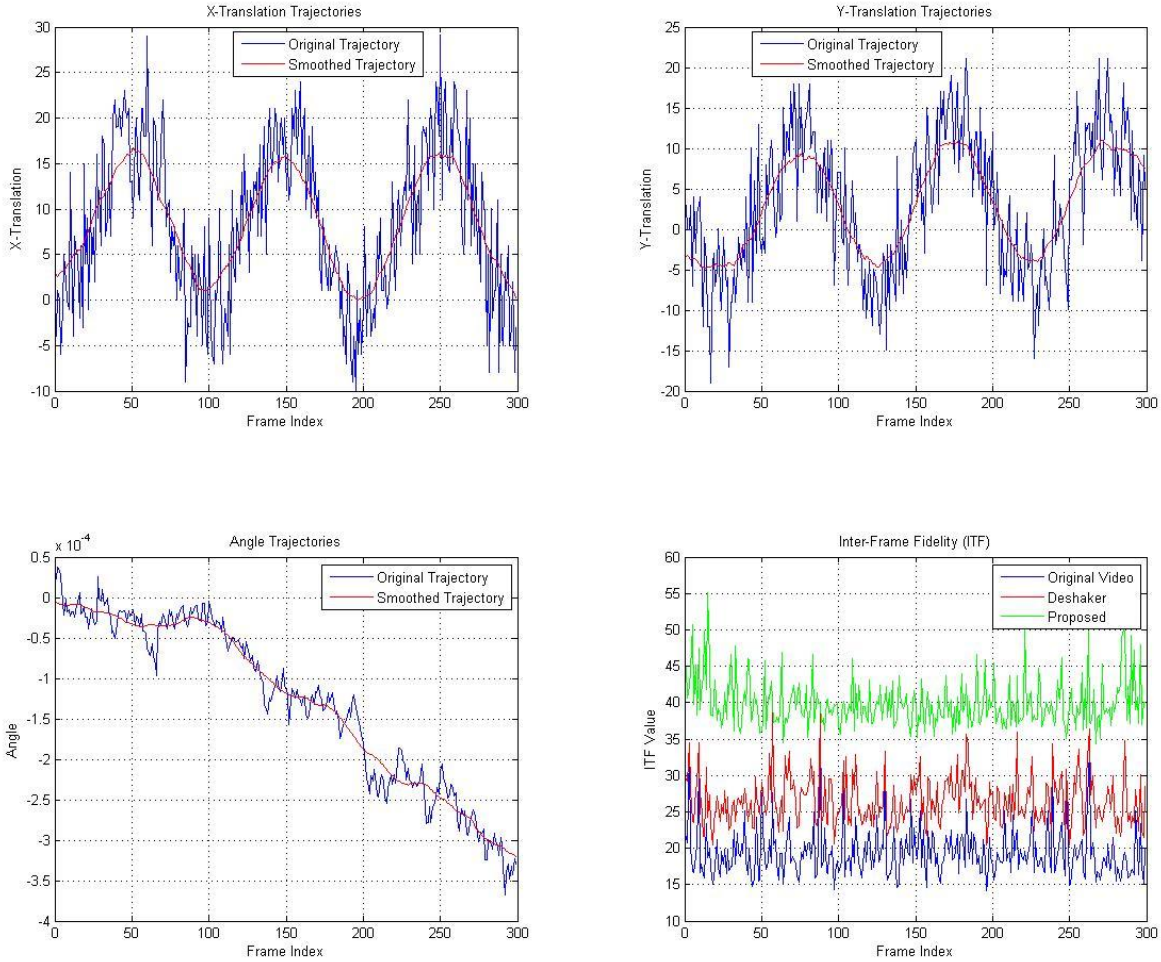


Figure 4.9: Sequence 10

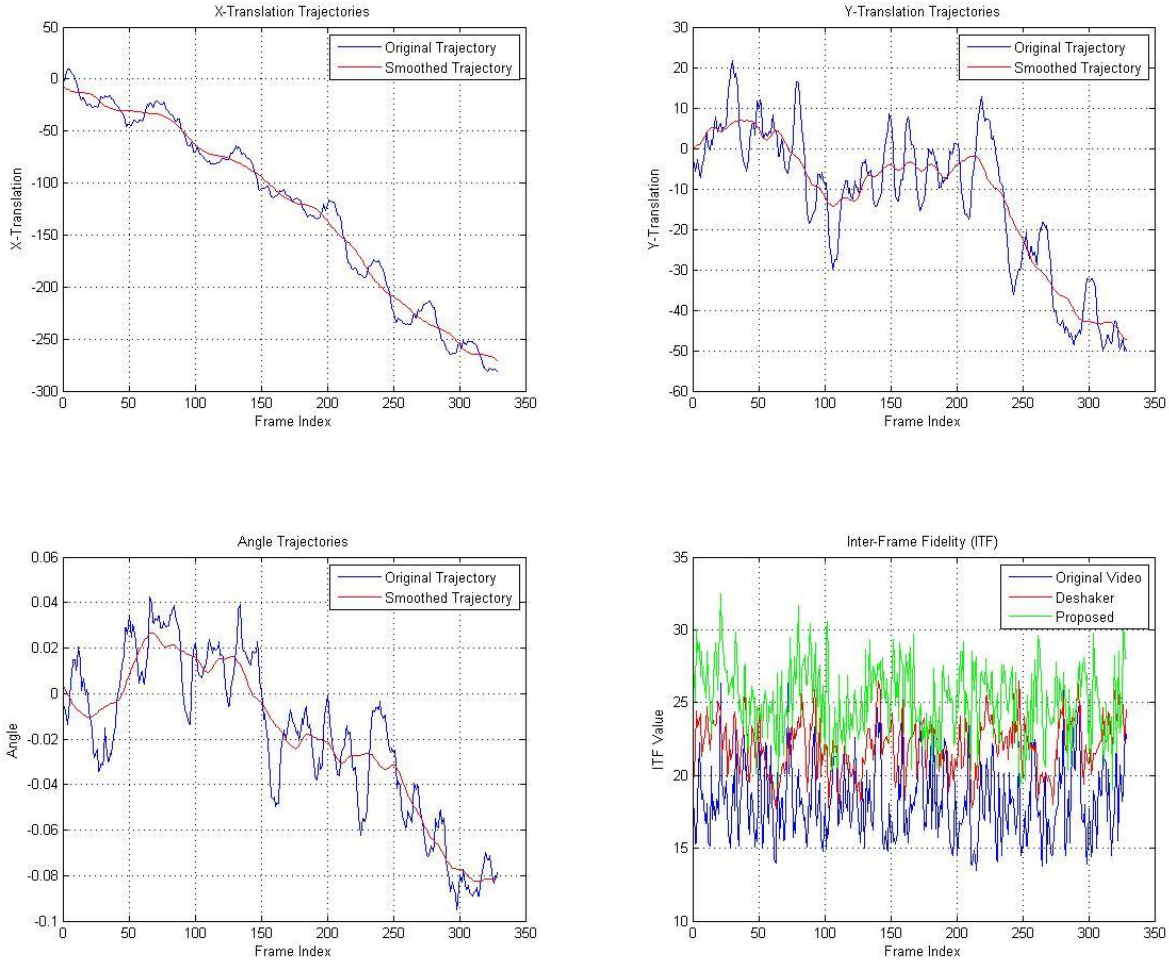


Figure 4.10: Sequence 11

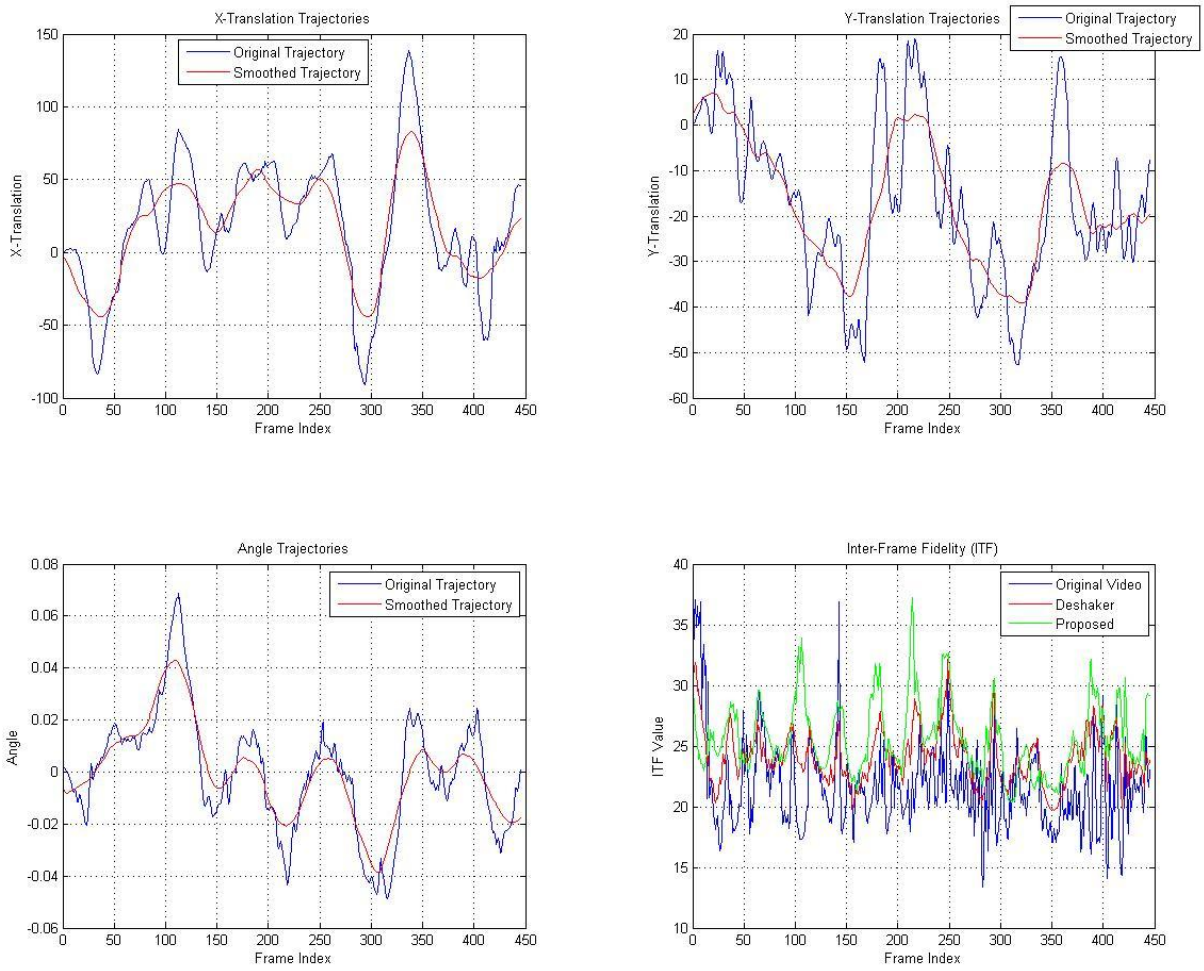


Figure 4.11: Sequence 18

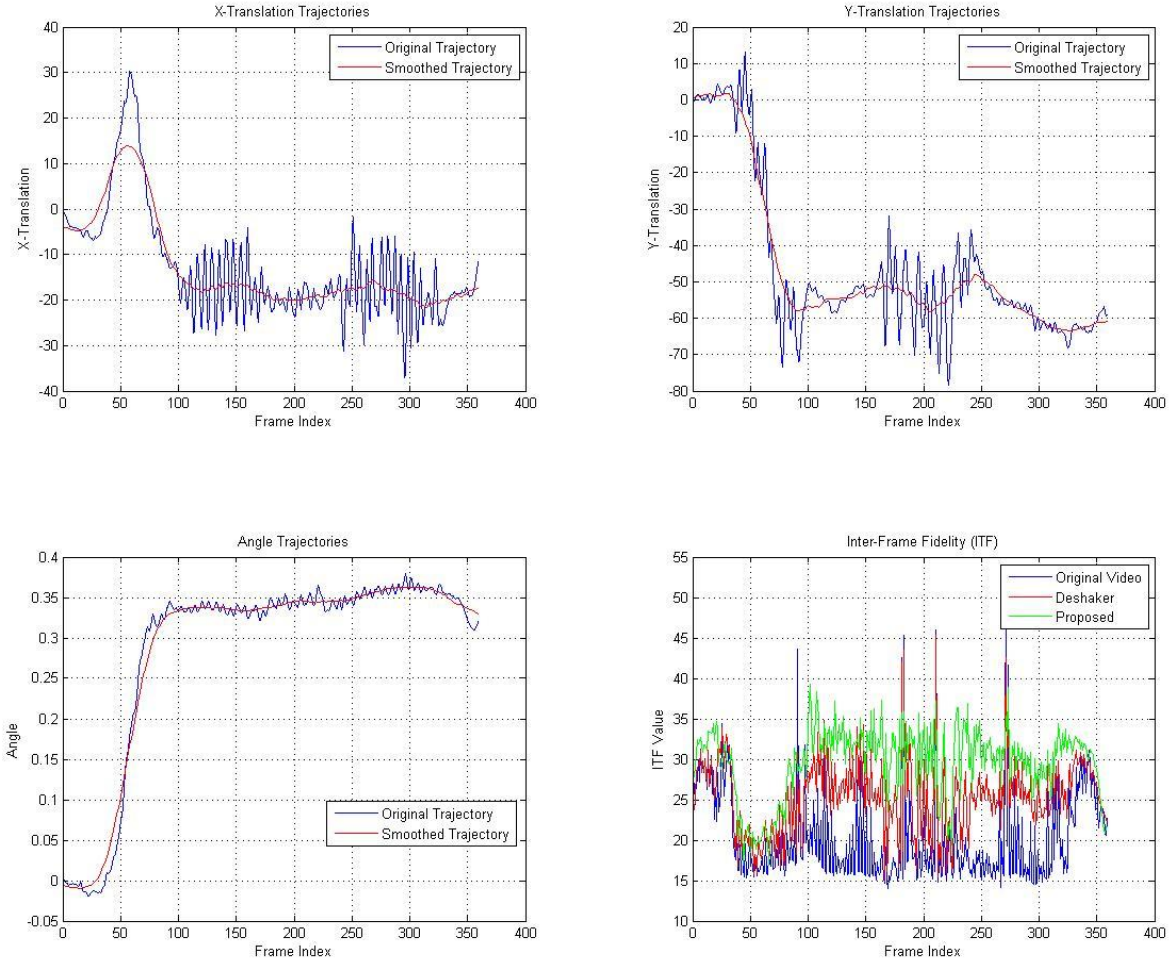


Figure 4.12: Sequence 19

Limitations of Algorithm

Our algorithm uses feature tracking for estimating inter-frame motion. Therefore, if a foreground moving body or object occupies a major portion of the scene, the algorithm may fail to stabilize against background and stabilize with respect to the foreground body motion.

Chapter 5

Conclusion and Future Work

5.1 Summary

We presented a fast and real-time solution to digital video stabilization in this thesis. The proposed algorithm is based on two-dimensional feature based motion estimation.

The method tracks a small set of feature points (detected using FAST corner detector) using pyramidal optical flow technique. An affine camera motion model (with x-, y- and angle-components) is used to model the movement of camera between consecutive frames. A low pass filter (non-weighted non-causal moving average filter) is applied to computed x-, y- and angle-component trajectories to preserve intentional camera motion and discard unwanted jitter. Finally, motion compensation is performed based on smoothed trajectories of x-, y- and angle-components generating stabilized video streams.

The proposed algorithm stabilizes horizontal/vertical panning and rotations and is suitable for a variety of applications such as cellphone camera video, camera mounted on car video, hand-held camcorder videos, etc. The algorithm employed many tweaks and improvements to bring down computational costs to real-time frame rates, such as:

- The algorithm adaptively selects feature points detector threshold as per video

frame size to save significant computational costs.

- The algorithm selects appropriate mask width to eliminate feature points which cannot be used for estimating frame-to-frame motion due to their high probability of disappearance in next frame as a result of camera shake.

- Our algorithm presented a novel CPU-GPU parallel computation framework. GPU was used to speed-up highly computational optical flow computations.

- In addition to GPU, parallel computation capabilities of multi-core processors was exploited to implement OpenCV routines pipeline leading to sizeable reduction in computational costs.

The minimal computational cost of this fast algorithm gives 73 FPS for VGA (640 x 480) and 32 FPS for 720p (1280 x 720) video sequences.

In addition to above, our work also provides insight into feature matching and image registration techniques. Performance of various feature detector and descriptor combinations were evaluated to determine how accurate is each feature detector-descriptor combination for estimating inter-frame motions subject to camera rotation, zoom, rotation+zoom and viewpoint changes. Efficiency of various false match rejection techniques such as nearest neighbor distance ratio, etc, were also analyzed quantitatively. Thus, our work provides a foundation for feature matching based digital video stabilization techniques, which can be used in future.

5.2 Future Work

The future possible extensions of this work are listed as:

- We used just one feature detector, i.e., FAST corner detector to reduce computations. In order to increase accuracy and robustness of frame-to-frame motion estimation, more than one feature detector of complementary types can be used such as a corner detector combined with blob detector.

- For feature points detection, our algorithm converted RGB image into grayscale image and feature points were tracked in grayscale image itself. This resulted in throwing away of useful color information. Color-based feature detection and description is still in nascent stage in literature which can be explored in future. Instead of grayscale conversion, some other channel such as Hue or combination of input channels can be explored, conversion into which will enhance the feature points for tracking purposes.

- The camera motion model can be made adaptive as per camera motion with increasing complexity (Rigid, Affine, Homography) in order to better estimate and model the camera motion.

- The radius of moving average filter (low pass filter) for motion motion filtering can be made adaptive. If the camera motion is larger than a threshold, it is detected as panning and filter radius is decreased to follow camera pannning or translation. If motion magnitude is small, it can be assumed as unwanted camera shake and larger radius filter can be applied to effectively dampen the camera shake.

Appendix A

Image Registration Concepts

A.1 Introduction

Geometric differences between two images that have the same or overlapping contents can be described using transformation functions. Given the coordinates of a point in one image, a transformation function will determine the coordinates of the same point in the other image. One of the image is called as the *reference image* and the other one as the *target image*. The transformation functions for image warping and image registration are determined using the coordinates of a number of corresponding points (called *control points*) in the images, either selected manually or determined by an automatic process.

In image registration, both reference and target images are given, and the target image is required to be deformed to match the reference image. A mapping is required to transform the target image to overlay the reference image.

A.2 Image Warping

Given the coordinates of N corresponding control points in two images of a scene:

$$\{(x_i, y_i), (X_i, Y_i) : i = 1, ..., N\}, \tag{A.1}$$

we want to determine a forward transformation function $\mathbf{f}$(x,y) with components

$f_x(x,y)$ and $f_y(x,y)$ that satisfy

$$X_i = f_x(x_i, y_i), \tag{A.2}$$

$$Y_i = f_y(x_i, y_i), i = 1, ..., N, \tag{A.3}$$

or in terms of backward transformation function

$$x_i = g_X(X_i, Y_i), \tag{A.4}$$

$$y_i = g_Y(X_i, Y_i), i = 1, ..., N, \tag{A.5}$$

Once $\mathbf{f}(x,y)$ is determined, given the coordinates of a point *(x,y)* in one image, the coordinates of the corresponding point in the other image can be determined. From now on in this chapter, we will refer to the image with coordinates *(x,y)* as *reference* and the image with coordinates *(X, Y)* as the *target*.

A.3 Types of Geometric Transformations

The images to be registered can often have non-linear geometric differences either due to non-linear nature of image acquisition systems and, sometimes due to the deformable nature of the scene. When the non-linear geometric difference between the images is negligible, a linear transformation can be used to register them. But in some cases, the geometric difference can be large and complex thus require a composite of local transformations to register them.

Based on linearity, transformations can be classified as:

- Linear Transformations
- Non-linear Transformations

Based on the geometric differences between the images, transformations can be of types:

- Global Transformations

- Local Transformations

Local transformations can be geometric transformations (linear or non-linear) applied individually to local parts of an image.

A.4 Global Transformations

In global transformation, a single function relates all the points in the whole image.

A.4.1 Affine Transformation

It preserves parallelism.

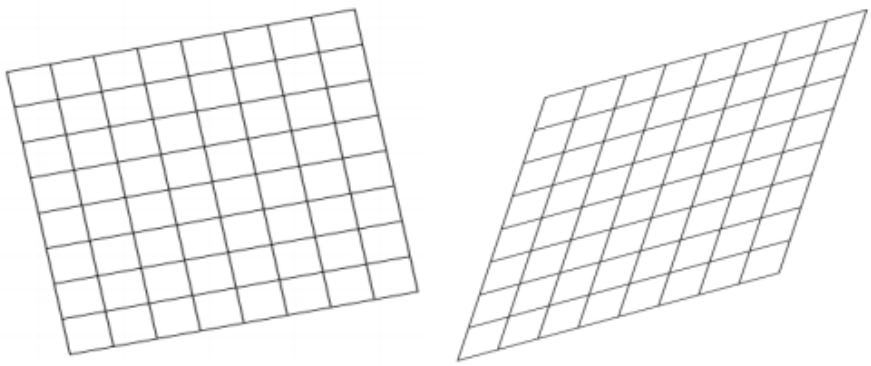


Figure A.1: Affine Transformation

$$X = a_1 x + a_2 y + a_3, \tag{A.6}$$

$$Y = a_4 x + a_5 y + a_6, \tag{A.7}$$

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{A.8}$$

Minimum 3 corresponding points are required to determine affine parameters.

A.4.2 Projective Transformation

Also known as *homography*, it preserves straightness.

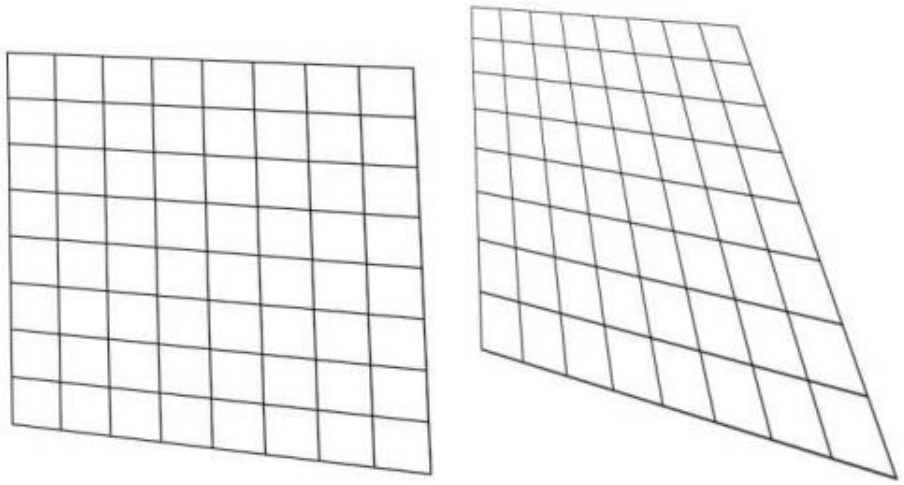


Figure A.2: Projective Transformation

$$X = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}, \tag{A.9}$$

$$Y = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}, \tag{A.10}$$

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{A.11}$$

Minimum 4 corresponding points are required to determine the projective transformation parameters.

A.4.3 Polynomial Transformation

The general form of polynomial transformation is:

$$X = \sum_k \sum_l a_{kl} x^k y^l, \tag{A.12}$$

$$Y = \sum_k \sum_l b_{kl} x^k y^l, \tag{A.13}$$

Example: 2nd-order polynomial transformation

$$X = a_{00} + a_{10} x + a_{01} y + a_{11} xy + a_{20} x^2 + a_{02} y^2, \tag{A.14}$$

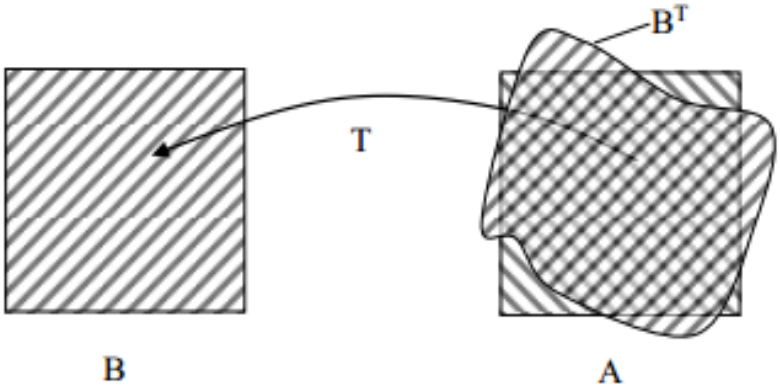


Figure A.3: Polynomial Transformation

$$Y = b_{00} + b_{10}x + b_{01}y + b_{11}xy + b_{20}x^2 + b_{02}y^2, \tag{A.15}$$

In matrix form, we have

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} a_{20} & a_{02} & a_{11} & a_{10} & a_{01} & a_{00} \\ b_{20} & b_{02} & b_{11} & b_{10} & b_{01} & b_{00} \end{bmatrix} \begin{bmatrix} x^2 \\ y^2 \\ xy \\ x \\ y \\ 1 \end{bmatrix}, \tag{A.16}$$

Minimum 6 corresponding points are required to determine 2nd-order polynomial transformation parameters.

Following figure shows results of global mapping using a 2nd order, 3rd order and 4th order polynomial function. As seen from the Figure A.4, 4th order polynomial function maps the transformation between reference and target images with best accuracy.

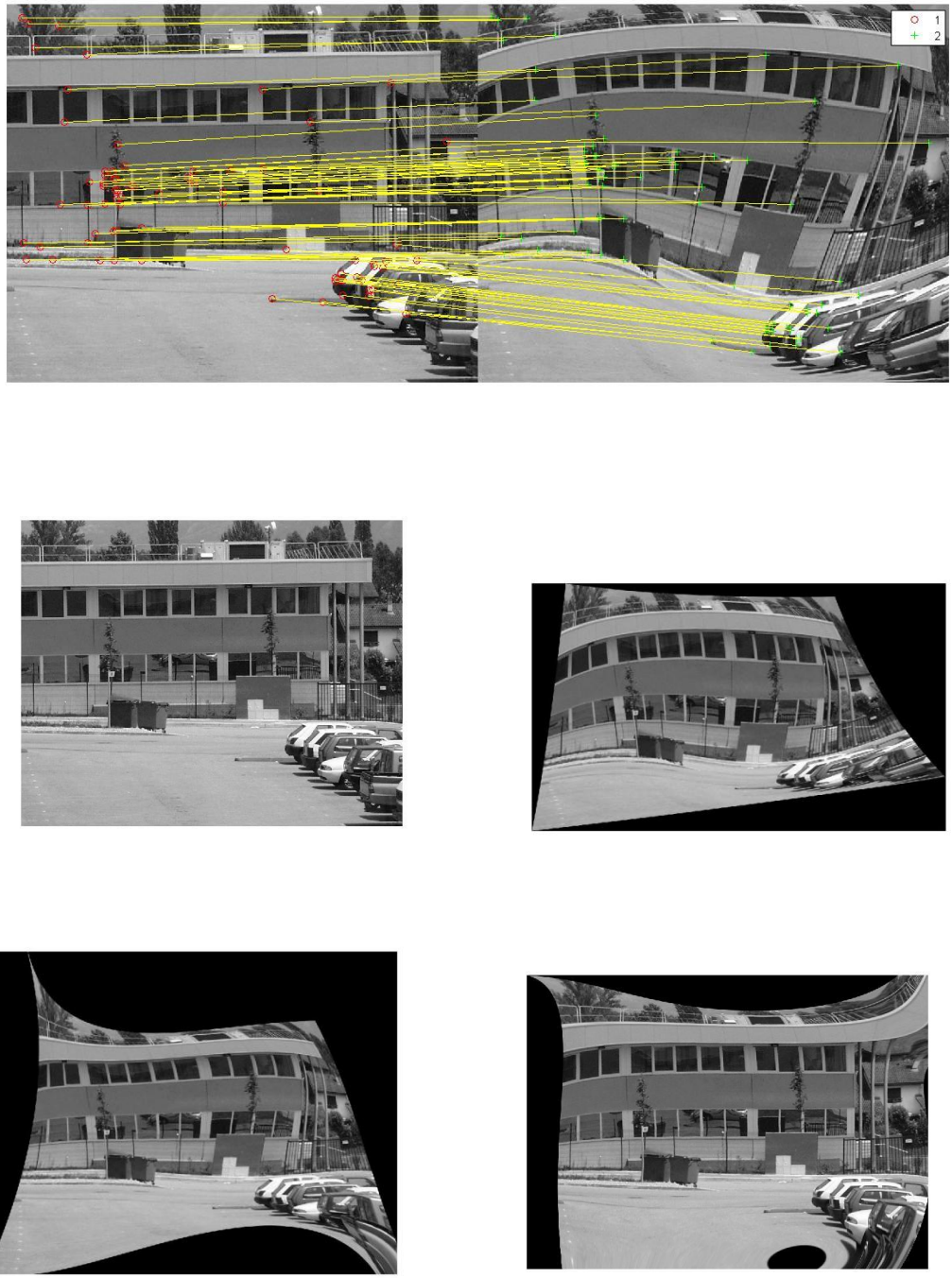


Figure A.4: Polynomial Transformation: (a) Matched Feature Points. (b) Original Image. Warped using (c) 2nd order. (d) 3rd order. (e) 4th order Polynomial function

A.5 Local Transformations

It is not convenient and accurate to describe local distortions differing at different locations using global transformation, which imposes a single mapping function on

the whole image. Local transformation applies a different mapping function to different parts of the image.

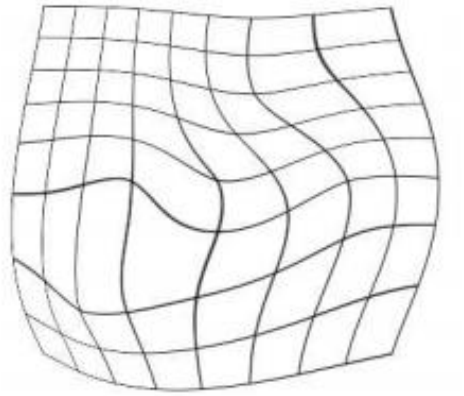


Figure A.5: Local Transformation

A.5.1 Piecewise Methods

Corresponding triangles in the target image are obtained by triangulating control points in the reference image and knowing the correspondences between control points in reference and target images. Piecewise methods map areas within corresponding triangles to each other.

A polynomial or linear transformation function can be used to map a triangle in the target image to the corresponding triangle in the reference image. The transformation will be continuous but not smooth. When the regions are small or local geometric difference between the images is small, piecewise linear may be sufficient. However, if local deformations are large, tangents at the two sides of a triangle edge can be quite different, resulting in an inaccurate registration.

Triangulation method affects registration result. As a general rule, avoid elongated triangles and triangles without acute angles should be given preference. Delaunay Triangulation A.6 algorithm can be used which maximizes the minimum angle in triangles.

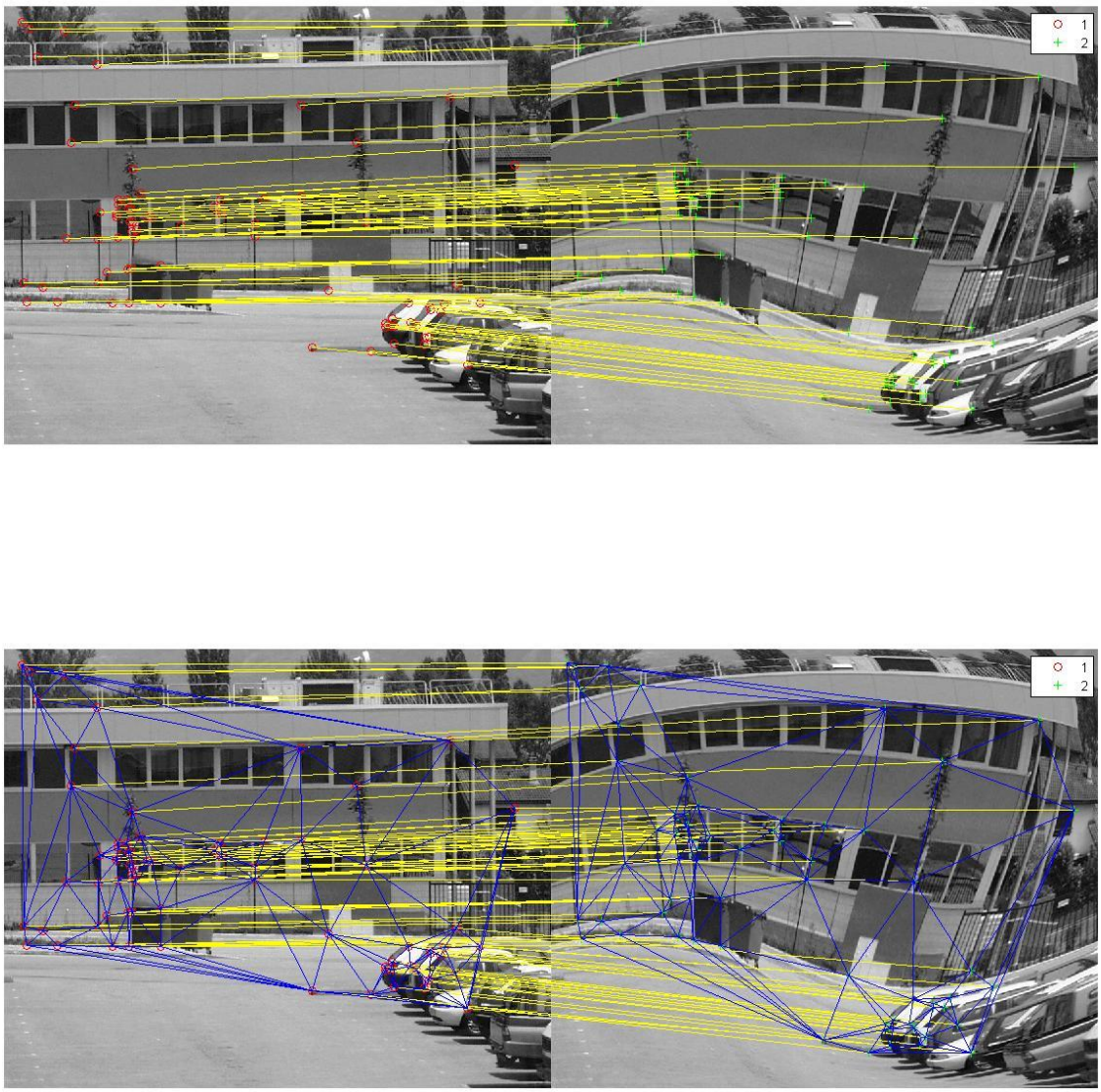


Figure A.6: Piecewise Methods: (a) Matched Feature Points. (b) Delaunay Triangulation

Piecewise linear and piecewise cubic transformation functions have been used to register images with non-linear geometric differences. However, limitations associated with piecewise methods are:

1. Good accuracy is achieved only within the convex hull of the control points of obtained triangles. Large errors can result due to extrapolation outside the convex hull of the control points.

2. Mapping each triangular region independently often leads to folding problems, i.e., more than one triangle in reference image may correspond to same region in the target image as visible in Figure A.7.

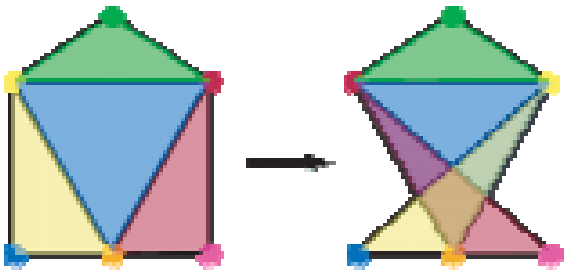


Figure A.7: Piecewise Methods: Folding Problem

A.5.2 Local Weighted Mean

Assume two polynomials, each with n parameters map an area in the reference image to an area in the target image. Now suppose that the area in the target image contains the ith control point and n - *1* of its nearest control points. Then the coefficients of the polynomials can be determined by substituting the coordinates of n corresponding control points into equations of the polynomials and solving two systems of n linear equations.

Using weighted mean method, each component of the transformation function can be obtained from the weighted mean of local polynomials. In order to make the transformation function local, the weight assigned to the ith polynomial is selected such that it becomes zero at a sufficiently large distance.

For non-uniform distribution of control points, extrapolation beyond the convex hull of control points can result in holes in the warped image. Therefore, when using local weighted mean method effort should be made to select control points over whole image as uniformly as possible.

Image registration results employing local weighted mean method are shown in Figure A.8 for varying number of control points.

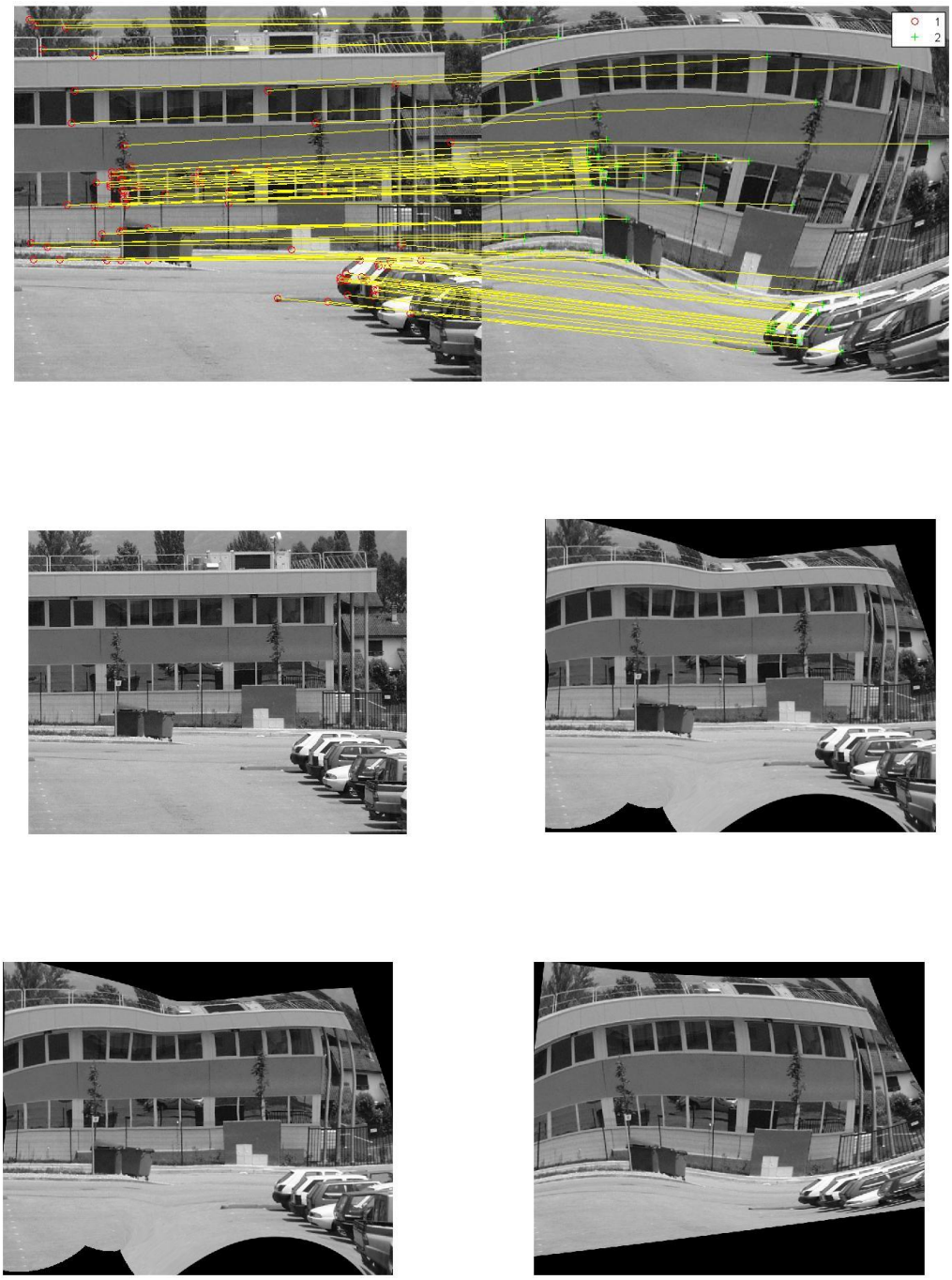


Figure A.8: Local Weighted Mean Method: (a) Matched Feature Points. (b) Original Image. Warped using (c) 25 control points. (d) 50 control points. (e) 75 control points

Bibliography

[1] R. Szeliski, "Image alignment and stitching: A tutorial," *Foundations and Trends® in Computer Graphics and Vision*, vol. 2, no. 1, pp. 1–104, 2006.

[2] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994, pp. 593–600.

[3] B. K. Horn and B. G. Schunck, "Determining optical flow," in *1981 Technical Symposium East*. International Society for Optics and Photonics, 1981, pp. 319–331.

[4] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision." in *IJCAI*, vol. 81, 1981, pp. 674–679.

[5] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 2, p. 3, 2001.

[6] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[7] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 404–417.

[8] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and vision computing*, vol. 22, no. 10, pp. 761–767, 2004.

[9] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 430–443.

[10] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2548–2555.

[11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: an efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2564–2571.

[12] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision–ECCV 2010*. Springer, 2010, pp. 778–792.

[13] A. Alahi, R. Ortiz, and P. Vandergheynst, "Freak: Fast retina keypoint," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 510–517.

[14] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[15] J. jae Lee and G. Kim, "Robust estimation of camera homography using fuzzy ransac," in *Computational Science and Its Applications–ICCSA 2007*. Springer, 2007, pp. 992–1002.

[16] F. Vella, A. Castorina, M. Mancuso, and G. Messina, "Digital image stabilization by adaptive block motion vectors filtering," *IEEE Transactions on Consumer Electronics*, vol. 48, no. 3, pp. 796–801, 2002.

[17] S. Battiato, G. Puglisi, and A. Bruna, "A robust video stabilization system by adaptive motion vectors filtering," in *Multimedia and Expo, 2008 IEEE International Conference on*. IEEE, 2008, pp. 373–376.

[18] J. Yang, D. Schonfeld, and M. Mohamed, "Robust video stabilization based on particle filter tracking of projected camera motion," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 7, pp. 945–954, 2009.

[19] 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*, vol. 28, no. 7, pp. 1150–1163, 2006.

[20] S. Erturk, "Digital image stabilization with sub-image phase correlation based global motion estimation," *Consumer Electronics, IEEE Transactions on*, vol. 49, no. 4, pp. 1320–1325, 2003.

[21] M. A. Alharbi, "Fast video stabilization algorithms," DTIC Document, Tech. Rep., 2006.

[22] J. A. Kumar and D. B. Naidu, "Digital video stabilization using sift feature matching and adaptive fuzzy filter," *Electrical Engineering*, 2013.

[23] Y. Zhang, M. Xie, and D. Tang, "A central sub-image based global motion estimation method for in-car video stabilization," in *Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on*. IEEE, 2010, pp. 204–207.

[24] B. Pinto and P. Anurenjan, "Video stabilization using speeded up robust features," in *Communications and Signal Processing (ICCSP), 2011 International Conference on*. IEEE, 2011, pp. 527–531.

[25] J. Xu, H.-w. Chang, S. Yang, and M. Wang, "Fast feature-based video stabilization without accumulative global motion estimation," *Consumer Electronics, IEEE Transactions on*, vol. 58, no. 3, pp. 993–999, 2012.

[26] T. Jin, H. Xiaowei, Y. Zhonghu, and Z. Hongying, "An approach of electronic image stabilization based on the representative point matching," in *Genetic and Evolutionary Computing, 2009. WGEC'09. 3rd International Conference on.* IEEE, 2009, pp. 347–350.

[27] W. Xu, X. Lai, D. Xu, and N. A. Tsoligkas, "An integrated new scheme for digital video stabilization," *Advances in Multimedia*, vol. 2013, 2013.

[28] K.-Y. Lee, Y.-Y. Chuang, B.-Y. Chen, and M. Ouhyoung, "Video stabilization using robust feature trajectories," in *Computer Vision, 2009 IEEE 12th International Conference on.* IEEE, 2009, pp. 1397–1404.

[29] D. Shukla, R. K. Jha, and K. Aizawa, "A novel approach for combined rotational and translational motion estimation using frame projection warping," in *Visual Communications and Image Processing (VCIP), 2013.* IEEE, 2013, pp. 1–6.

[30] Z. XI and S. CHU, "Two-step global motion estimation algorithm based on the robustness region," *Journal of Computational Information Systems*, vol. 9, no. 3, pp. 1019–1026, 2013.

[31] S. Battiato, G. Gallo, G. Puglisi, and S. Scellato, "Sift features tracking for video stabilization," in *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on.* IEEE, 2007, pp. 825–830.

[32] H. Zhang, C. Deng, J. Li, F. Yuan, and R. Jia, "Fast digital image stabilization algorithm based on polar transform and circular block matching," in *Signal Processing, 2008. ICSP 2008. 9th International Conference on.* IEEE, 2008, pp. 1124–1127.

[33] H.-C. Chang, S.-H. Lai, and K.-R. Lu, "A robust real-time video stabilization algorithm," *Journal of Visual Communication and Image Representation*, vol. 17, no. 3, pp. 659–673, 2006.

[34] Z. Juanjuan and G. Baolong, "Electronic image stabilization system based on global feature tracking," *Systems Engineering and Electronics, Journal of*, vol. 19, no. 2, pp. 228–233, 2008.

[35] A. Litvin, J. Konrad, and W. C. Karl, "Probabilistic video stabilization using kalman filtering and mosaicing," in *Electronic Imaging 2003.* International Society for Optics and Photonics, 2003, pp. 663–674.

[36] Y. Wang, Z. Hou, K. Leman, and R. Chang, "Real-time video stabilization for unmanned aerial vehicles." in *MVA*, 2011, pp. 336–339.