**CHAPTER 3**

**VIDEO STABILIZATION VIA POINT FEATURE**

**MATCHING**

1. **Introduction**

In this chapter, Video Stabilization via Point Feature Matching will be discussed. There are three sections for this system. Each section has a lot of steps to process. Firstly, input videos are taken by camera and send to processing unit. Secondly, processing unit maintains the video to get stabilized frame sequence and produced output to the display unit. Finally, display unit shows video with stabilized frame sequence. The processing steps can be configured by following Figure 3.1.

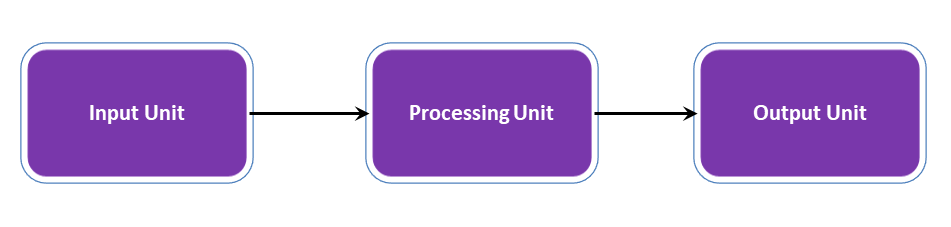


Figure 3.1: General Video Stabilization Steps

1. **Input Unit**

There are mainly two types of video inputs; analog and digital. Analog videos are widely used in many areas for years. The basic analog video standards are PAL (Phase Alternation Line-Europe) and NTSC (National Television Standards Committee-USA) which have been used since the foundation of the color TV. Digital video standards are newer when compared to analog standards. The well known digital standards are DVI (Digital Video Interface) and Cameralink.

The type of video input is related with the capability of the hardware. Most of videos are recorded by camera and stored as file systems in memory drive. Unstabilized video are occurred when recording camera are shake by movement of outer effects. That shaking video can be stabilized by using this method. In this thesis

study, unstabilized recorded videos are used for stabilization method, but future applications may use camera to take video sequence for real time video stabilization.

* 1. **Processing Unit**

This thesis adopts RANSAC paradigm to stabilize a shaky video sequence. The input video frames are modified with the purpose of maintaining a stable image. The implemented framework presented in Figure 3.1 will be discussed in this chapter.

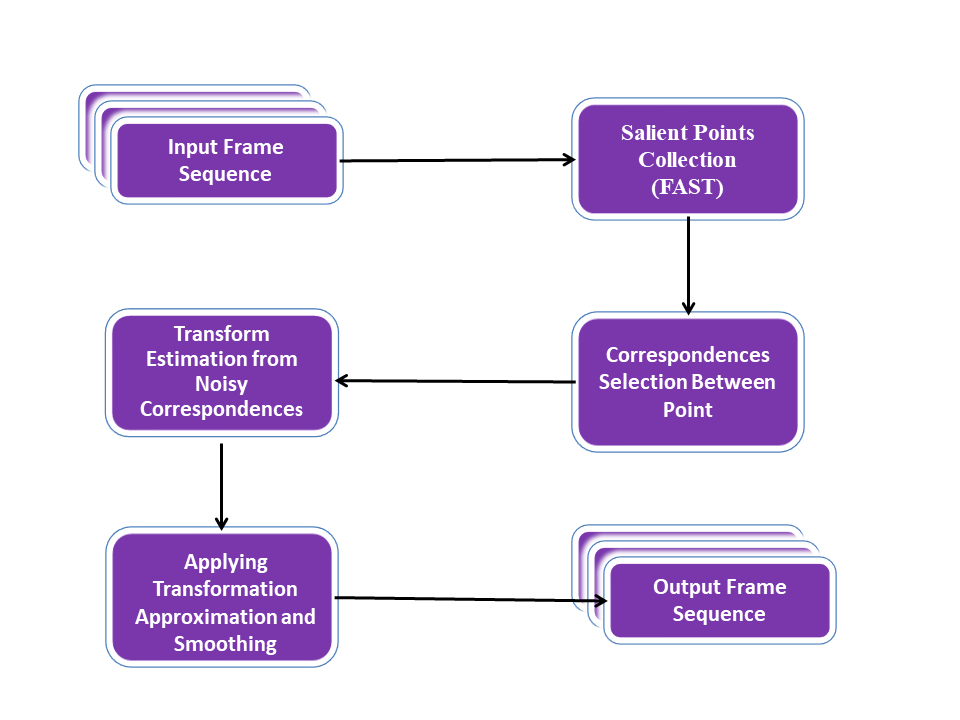


Figure 3.1: Video Stabilization Procedure

1. **Reading Video Frames**

The first step of video stabilization algorithm is to read the first two consecutive frames (Frame A and Frame B) of the video as grayscale images.

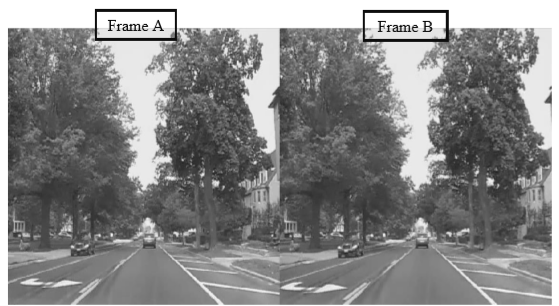


Figure 3.2: The First Two Frames of the Video

1. **Salient Points Collection**

In general, points containing the dominant information of an image are referred to as salient points. The first step of any robust estimation technique is detecting the salient points. Corner points and edges of an image are the best candidates for salient points. Some corner detection algorithm are sa follow:

* Moravec corner detection algorithm
* Harris corner detection algorithm
* Noble corner detection algorithm
* SUSAN corner detection algorithm
* FAST corner detection algorithm

In this study, FAST corner detection algorithm is used.

**3.3.2.1. Features from accelerated segment test (FAST)**

Features from accelerated segment test (FAST) is a corner detection method, which could be used to extract feature points and later used to track and map objects in many computer vision tasks. The FAST corner detector was originally developed by Edward Rosten and Tom Drummond, and was published in 2006.[[1]](https://en.wikipedia.org/wiki/Features_from_accelerated_segment_test#cite_note-1) The most promising advantage of the FAST corner detector is its computational efficiency. Referring to its name, it is indeed faster than many other well-known feature extraction methods, such as Difference of Gaussians (DoG) used by the SIFT,  SUSAN and Harris detectors. Moreover, when machine learning techniques are applied, superior performance in terms of computation time and resources can be realised. The FAST corner detector is very suitable for real-time video processing application because of this high-speed performance.

**3.2.2.2. Feature Detection using FAST**

1. Select a pixel *p* in the image which is to be identified as an interest point or not. Let its intensity be *Ip*.
2. Select appropriate threshold value *t*.
3. Consider a circle of 16 pixels around the pixel under test. (See the image below)

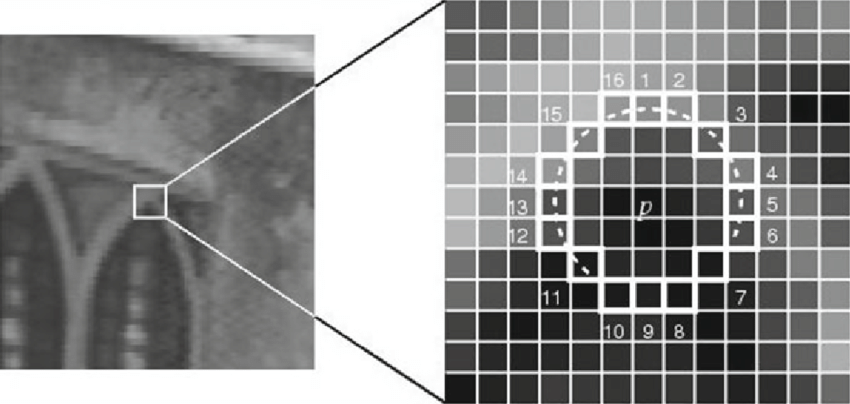


Figure 3.3: Feature Detection using FAST

1. Now the pixel *p* is a corner if there exists a set of *n* contiguous pixels in the circle (of 16 pixels) which are all brighter than *Ip + t*, or all darker than *Ip − t*. (Shown as white dash lines in the above image). *n* was chosen to be 12.
2. A high-speed test was proposed to exclude a large number of non-corners. This test examines only the four pixels at 1, 9, 5 and 13 (First 1 and 9 are tested if they are too brighter or darker. If so, then checks 5 and 13). If *p* is a corner, then at least three of these must all be brighter than *Ip + t* or darker than *Ip − t*. If neither of these is the case, then *p* cannot be a corner. The full segment test criterion can then be applied to the passed candidates by examining all pixels in the circle. This detector in itself exhibits high performance, but there are several weaknesses:

* It does not reject as many candidates for n < 12.
* The choice of pixels is not optimal because its efficiency depends on ordering of the questions and distribution of corner appearances.
* Results of high-speed tests are thrown away.
* Multiple features are detected adjacent to one another.

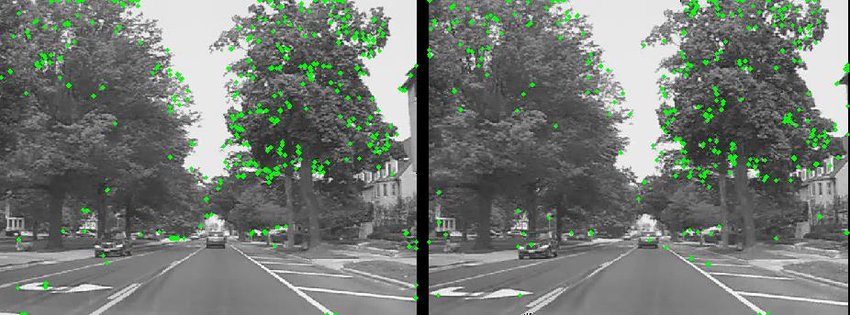


Figure 3.4: Feature Points in Frame A and B

1. **Correspondences Selection between Points**

In order to stabilize a video sequence we mainly need to find a transformation which reduces the amount of distortion between frames. In this step the likely correspondences between the derived points of interest are selected. In order to find the correspondences between feature points we extract a 9×9 block centered on each point. Sum of Squared Differences (SSD) is then adopted as the matching cost between respective points.

For two images *f*(𝑥,𝑦) and 𝑔(𝑥,𝑦) SSD can be defined as following.

SSD(𝑑1,𝑑2) = ∑ ∑ (𝑓(𝑥 +𝑖,𝑦+𝑗)−𝑔(𝑥 +𝑖 −𝑑1 ,𝑦+𝑗 −𝑑2))2 (5.2)

Where the summation extends over a region of size (2𝑛1 +1)×(2𝑛2 +1) and as we have chosen a 9×9 block so 𝑛1= 𝑛2= 4.

There exists one point in Frame B which corresponds to the points in Frame A. When finding all possible matching costs the algorithm searches to find the lowest one which means the best cost.

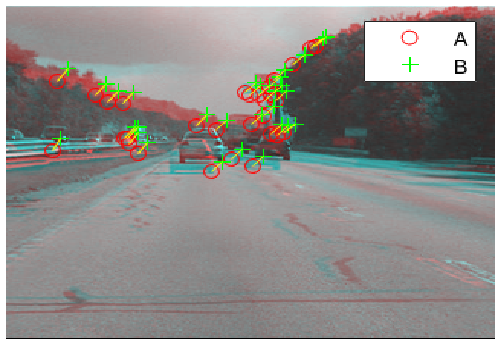


Figure 3.5: Correspondences Between Frame A and B

1. **Transform Estimation from Noisy Correspondences**

Numerous correspondences achieved in the previous step are not acceptable. Using the Random Sample Consensus (RANSAC) algorithm, a robust estimate of transformation between Frame A and Frame B can be derived. Receiving the point correspondences from the previous step, the video stabilization algorithm searches to find effective inlier correspondences and afterward it derives the affine transformation mapping the inliers in Frame A to Frame B. This transformation is only capable to alter the image plane.

As mentioned in Chapter 2 the affine transform is a matrix of the following form.

(3.3)

Where 𝑡𝑥 and 𝑡𝑦 are translation parameters and 𝑎1, 𝑎2, *a*3 and 𝑎4 describe sheering effect, rotation and scale. The affine transform targets to overlay the correspondence points on each other by warping the image.

This geometric transformation is estimated several times and for each result a cost is calculated based on the Sum of Absolute Differences (SAD) between frame A and B. The best transform which minimizes the cost is selected. This procedure increases the robustness.

SAD is the most commonly used algorithm which measures the distortion between two images by evaluating the similarity between image blocks. Equation 5.4 defines the SAD between elements in two image blocks.

SAD = (3.4)

Where 𝑟𝑖, represent elements in the first frame and 𝑐𝑖, are the elements in the second one.

In the color composite Figure 3.6 the re-projected Frame B is laminated on Frame A. As illustrated in the figure the inlier correspondences strongly match. The centers of the images are aligned where the red-cyan color composite is almost black and white.



Figure 3.6: Correct Correspondences Based on RANSAC Paradigm

1. **Random sample consensus (RANSAC)**

Random sample consensus (RANSAC) is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers, when outliers are to be accorded no influence on the values of the estimates. Therefore, it also can be interpreted as an outlier detection method. It is a non-deterministic algorithm in the sense that it produces a reasonable result only with a certain probability, with this probability increasing as more iterations are allowed. The algorithm was first published by Fischler and Bolles at SRI International in 1981. They used RANSAC to solve the Location Determination Problem (LDP), where the goal is to determine the points in the space that project onto an image into a set of landmarks with known locations.

A basic assumption is that the data consists of "inliers", i.e., data whose distribution can be explained by some set of model parameters, though may be subject to noise, and "outliers" which are data that do not fit the model. The outliers can come, for example, from extreme values of the noise or from erroneous measurements or incorrect hypotheses about the interpretation of data. RANSAC also assumes that, given a (usually small) set of inliers, there exists a procedure which can estimate the parameters of a model that optimally explains or fits this data.

**3.3.5.1. Example of RANSAC**

A simple example is fitting a line in two dimensions to a set of observations. Assuming that this set contains both inliers, i.e., points which approximately can be fitted to a line, and outliers, points which cannot be fitted to this line, a simple least squares method for line fitting will generally produce a line with a bad fit to the data including inliers and outliers. The reason is that it is optimally fitted to all points, including the outliers. RANSAC, on the other hand, attempts to exclude the outliers and find a linear model that only uses the inliers in its calculation. This is done by fitting linear models to several random samplings of the data and returning the model that has the best fit to a subset of the data. Since the inliers tend to be more linearly related than a random mixture of inliers and outliers, a random subset that consists entirely of inliers will have the best model fit. In practice, there is no guarantee that a subset of inliers will be randomly sampled, and the probability of the algorithm succeeding depends on the proportion of inliers in the data as well as the choice of several algorithm parameters.

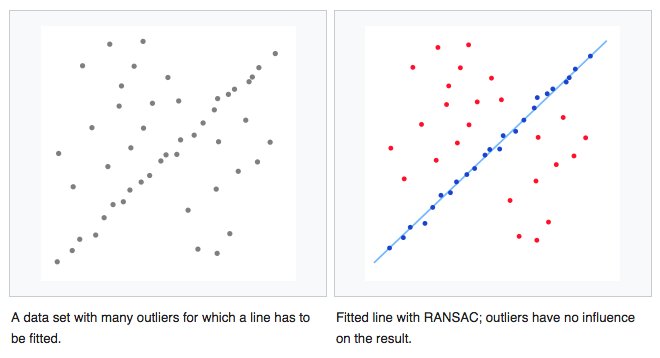


Figure 3.7: The data set observation (RANSAC)

1. **Transform Approximation and Smoothing**

We could use all the six parameters of the affine transform but for numerical simplicity and stability, we choose to re-fit the matrix as a simple scale-rotation-translation transform (s-R-t Transform).

This transform has only four free parameters which are one scale factor, one angle, and two translations. Construction of s-R-t transform is performed as follows:

1. Extract scale and rotation part of sub- matrix from affine transform of matrix.
2. Calculate angle from mean of two possible arctangents.
3. Compute scale from mean of two stable mean calculations
4. Translation will remain same.

This new transform matrix is of the form:

(3.5)

Where *s* is the scale factor, 𝑡𝑥 and 𝑡𝑦 are the two translation parameters and the parameter (*ang)* is the angle describing the rotation. This matrix contains two translation factors, one angle and one scale. In order to show that the error of replacing the transform H with the equivalent transform given as equation 3.5 is minimal, we re-projected the two processed Frame B on each other as a red-cyan composite which is depicted in Figure 3.8. The pixel-wise difference between images can be neglected and the image appears nearly black and white.



Figure 3.8: Color Composite of Affine and s-R-t Transform Outputs

1. **Running the Full Video**

The last step of video stabilization algorithm is to run the above procedure in a loop for all frames in a video sequence.

The transform *H* is calculated between consecutive frames in each step and the result is smoothed by fitting *H* as an s-R-t transform. Then the result in each loop is combined with the 𝐻𝑐𝑢𝑚𝑢𝑙𝑎𝑡𝑖𝑣𝑒 describing the entire camera motions. This transform is estimated in every loop.

1. **Output Unit**

Processing unit stabilized the video and produced output to the display unit. The output unit is a unit that is constructed to display the stabilized video. So, this unit can also be called display unit. In video stabilization, the output is the stabilized video sequence, in stereo matching it is the disparity map and in object tracking, it is the sign on the object. Like video inputs, there are two video output types; analog and digital. All video input standards are also valid for video outputs.

Analog video outputs can be displayed on CRT (Cathode Ray Tubes) monitors and digital outputs are displayed on digital monitors like LCD (Liquid Crystal Display). In this thesis study, the output of the stabilization algorithm will be displayed on LCD via DVI standard. The DVI signals are generated by FPGA.

1. **PYNQ Overlays**

The Xilinx® Zynq® All Programmable device is an SOC based on a dual-core ARM® Cortex®-A9 processor (referred to as the Processing System or PS), integrated with FPGA fabric (referred to as Programmable Logic or PL). The PS subsystem includes a number of dedicated peripherals (memory controllers, USB, Uart, IIC, SPI etc) and can be extended with additional hardware IP in a PL Overlay.

Overlays, or hardware libraries, are programmable/configurable FPGA designs that extend the user application from the Processing System of the Zynq into the Programmable Logic. Overlays can be used to accelerate a software application, or to customize the hardware platform for a particular application.

For example, image processing is a typical application where the FPGAs can provide acceleration. A software programmer can use an overlay in a similar way to a software library to run some of the image processing functions (e.g. edge detect, thresholding etc.) on the FPGA fabric. Overlays can be loaded to the FPGA dynamically, as required, just like a software library. In this example, separate image processing functions could be implemented in different overlays and loaded from Python on demand.

PYNQ provides a Python interface to allow overlays in the PL to be controlled from Python running in the PS. FPGA design is a specialized task which requires hardware engineering knowledge and expertise. PYNQ overlays are created by hardware designers, and wrapped with this PYNQ Python API. Software developers can then use the Python interface to program and control specialized hardware overlays without needing to design an overlay themselves. This is analogous to software libraries created by expert developers which are then used by many other software developers working at the application level. The following figure 3.9 shows the internal structure of PYNQ.

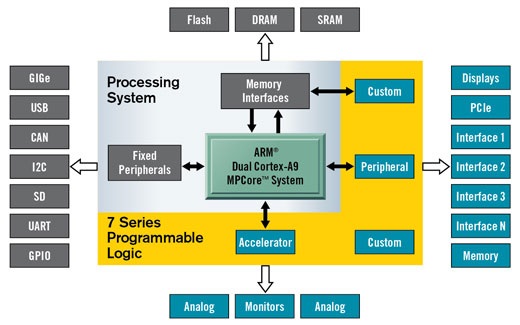


Figure 3.9: Internal Structure of PYNQ