VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnana Sangama", Belagavi- 590 018

PROJECT REPORT

ON

"Dynamic Video Stitching via Shakiness Removal"

Submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Under the Guidance of

Mrs. Arpitha C.N **B.E., M.Tech.,**
Assistant Professor, Department of CS & E
AIT, Chikkamagaluru

Submitted by

Gokul K.B. (4AI14CS031) **Muskaan(4AI14CS050)**

Amulya (4AI14CS007) **Ashik M.S.(4AI14CS015)**

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY
(Affiliated to V.T.U., Accredited by NBA)
CHIKKAMAGALURU-577102, KARNATAKA
2017- 2018

ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belagavi)
CHIKKAMAGALURU, KARNATAKA, INDIA -577 102

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the **Project Work – 10CS85 entitled** ***" Dynamic Video Stitching via Shakiness Removal"*** is a bonafide work carried out by **Gokul K.B(4AI14CS031), Muskaan(4AI14CS050), Amulya (4AI14CS007), Ashik M.S (4AI14CS015)** student of VIII semester B.E, in partial fulfillment for the award of Degree of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belagavi, during the year **2017 - 2018**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report. The project report has been approved, as it satisfies the academic requirements in respect of project prescribed for the said Degree.

Signature of the Guide
Ms. Arpitha C.N. **B.E., M.Tech.,**
Assistant Professor
Dept. of CS & E

Signature of the Coordinator-1
Mr. Taranath N.L **M.Tech., (Ph.D)**
Assistant Professor
Dept. of CS & E

Signature of the Coordinator-2
Mr. Darshan L .M **B.E., M.Tech.,**
Assistant Professor
Dept. of CS & E

Signature of the HOD
Dr. Pushpa Ravikumar **B.E., M.Tech., Ph.D, LMISTE**
Professor & Head
Dept. of CS & E

Signature of the Principal
Dr. C.T Jayadeva **B.E., M.Tech., Ph.D**
Principal,
A.I.T, Chikkamagaluru

External Examiners	**Signature**
1. ________________	________________
2. ________________	________________

ABSTRACT

Stitching videos captured by hand-held mobile cameras can essentially enhance entertainment experience of ordinary users. However, such videos usually contain heavy shakiness and large parallax, which are challenging to stitch. We propose a novel approach of video stitching and stabilization for videos captured by mobile devices. The main component of our method is a unified video stitching and stabilization optimization that computes stitching and stabilization simultaneously rather than does each one individually. In this way, we can obtain the best stitching and stabilization results relative to each other without any bias to one of them.

We propose an optimized method to identify background of input videos, and also common background of them. This allows to apply our optimization on background regions only, which is the key to handle large parallax problem. Since stitching relies on feature matches between input videos, and there inevitably exist false matches, we thus propose a method to distinguish between right and false matches, and encapsulate the false match elimination scheme and our optimization into a loop, to prevent the optimization from being affected by bad feature matches. We test the proposed approach on videos that are causally captured by smartphones when walking along busy streets, and use stitching and stability scores to evaluate the produced panoramic videos quantitatively.

ACKNOWLEDGEMENT

We express our humble pranams to his holiness **Divine soul Bhairavaikya Padmabhushana Paramapoojya Jagadguru Sri Sri Sri Dr. Balagangadaranatha Mahaswamigi and Paramapoojya Jagadguru Sri Sri Sri Nirmalanandanatha mahaswamiji, Pontiff, Adichunachanagiri Mahasamasthana Mutt and Sri Sri Gunanatha Swamiji, A.M.S.M, Shringere Branch** who have showered their blessings on us for framing our carrier successfully.

We are deeply indebted to our honorable Director **Dr. C.K.SUBBARAYA** for creating the right kind of care.

We are deeply indebted to our honorable principal **Dr. C.T.Jayadeva** for his constant support and encouragement.

We express our deepest gratitude to **Dr. PUSHPA RAVIKUMAR,** Professor and Head of the Department of Computer Science & Engineering A.I.T, for her valuable guidance, suggestions and constant encouragement without which success of our project work would have been difficult.

We are grateful to our coordinators, **Mr. Taranath N L** and **Mr**. **Dharshan L M** for their excellent guidance, constant encouragement, support and constructive suggestions.

We are thankful to our guide **Mrs**. **Arpitha C N,** Asst. Professor, Department of Computer Science & Engineering, for his inspiration and lively correspondence right from the beginning of our project work till its completion.

We would like to thank our beloved parents for their support, encouragement and blessings.

And last but not the least, we would be very pleased to express our heartfelt thanks to the all teaching and non-teaching staff of CS&E Department and our friends who have rendered their help, motivation and support.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE No.

LIST OF FIGURES

LIST OF SNAPSHOTS

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Introduction

Image processing is generally a study of computer algorithms which are used for analysis and manipulation of the digitized images specifically to enhance the image quality. It is the manipulation and analysis of digitized images specifically in order to improve its quality. Video stitching is a technique that stitches multiple videos acquired from moving cameras. By allowing the videos to have a wider field of view (FOV) and higher a resolution, users can be immersed in the video. Video stitching has been applied in various fields, including surveillance systems, teleconferencing, immersive virtual reality, and sports broadcasting.

Taking casual videos by personal devices such as smart phones has become a popular way to record highlights of people's personal lives. However, despite the increasing sensor resolution, the field of view (FOV) of the cameras are still limited by the built-in consumer-grade lens, making the captured videos less immersive. Large FOV videos can be obtained with professional photography equipments which however are expensive and far from being popular. Therefore, video stitching, the technique of merging multiple input videos together into a wider-angle panoramic video, is a good compensation for personal devices, and has obtained a lot of research in the image and video processing community.

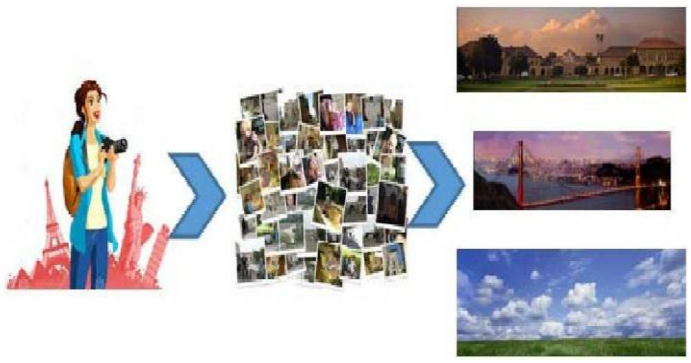


Figure 1.1: Panoramic view

Video stitching is a technique in which several videos of overlapping domain of view are blended together to result in a panoramic video of high resolution. Video stitching surveys depict that video stitching is till now a perplexing issue for panoramic videos. Video stitching is the current research area in the fields of computer vision, computer graphics and Photo graphics.

1.2 Motivation

The development and requirements of the multimedia products is increasingly raising fast in the recent time resulting in insufficient bandwidth of the network and storage of memory device. Digital videos tend to be very large in size and they require larger storage space. As the videos are very large, they need larger bandwidth and time for downloading or uploading through internet. It is also difficult to get a wide-angle view of the same scene. Hence Dynamic video Stitching provides a wide-angle view of the same two input videos.

The method used in this project emphasis on edges as it contributes towards the visual quality of the two input videos and hence the stitching is done without causing major degradations to the quality of the video. Therefore, this incorporates wide angle panoramic view.

1.3 Problem Statement

To design and implement dynamic video stitching via shakiness removal. Initial step is to capture video or an image by smart phone. By using optimal unified video stitching and stabilization framework, shaky paths are removed and small grids are stabilized to produce a panoramic video.

1.4 Scope of the project

The scope of the implemented system are:

- It can be applied in computer generated virtual reality scenarios which creates environment that is similar to the real world or in physical reality.
- It is suitable for multi view video stitching also called panoramic video stitching, which is an emerging and common research area in computer vision and computer graphics.
- It can be used as an approach in Surveillance system for stitching videos automatically using multi cameras.

- It can be used in Human Computer Interaction system for co-articulation to improve the animation performance.

1.5 Objectives

The objectives of the video stitching system are:

- To analyse unified video stitching and stabilization of videos.
- To obtain good stitching result, stabilization is made by choosing 2D based bundled camera paths framework.
- To identify stitching score and pixel measurement.
- To stabilize grids and frames of videos.
- To enhance stabilization of background regions, to handle large parallax.
- To get a better visual quality.
- To obtain high resolution photo mosaics in digital maps and satellite videos.

1.6 Review of the Literature

A literature survey is a record that has been already published on a topic by the scholars and researchers. It is briefing and explanation to the current knowledge on the topic presented in academic books and articles in journals.

[1] M. Zheng, X. Chen, and L. Guo, "Stitching video from webcams", in International Symposium on Visual Computing. Springer,2008, pp.420–429.

Zheng, x chen et.al presents a technique to create wide field-of-view from common webcams. The system consists of two stages: the initialization stage and the real-time stage. In the first stage, we detect robust features in the initial frame of each webcam and find the corresponding points between them. Then the matched point pairs are employed to compute the perspective matrix which describes the geometric relationship of the adjacent views. After the initialization stage, we register the frame sequences of different webcams on the same plane using the perspective matrix and synthesize the overlapped region using a nonlinear blending method in real time. In this way, the narrow fields of each webcam are displayed together as one wide scene.

It demonstrates the effectiveness on a prototype that consists of two ordinary webcams and show that this is an interesting and inexpensive way to experience the wide-angle observation. Authors, proposed a novel method to create panoramic video from webcams. Different from previous video mosaics which move one camera to record a continuous image sequence and then create a static panoramic image, they capture two pass videos and stitch each pair of frames from the different videos in real-time. In other words, their panorama is a wide video displayed in real-time instead of a static panoramic picture.

[2] F. Perazzi, A. Sorkine-Hornung, H. Zimmer, P. Kaufmann, O. Wang,S. Watson, and M. Gross, "Panoramic video from unstructured camera arrays," in Computer Graphics Forum, vol. 34, no. 2. Wiley Online Library, 2015, pp. 57–68.

F. Perazzi describes an algorithm for generating panoramic video from unstructured camera arrays. Artifact-free panorama stitching is impeded by parallax between input views. Common strategies such as multi-level blend- ing or minimum energy seams produce seamless results on quasi-static input. However, on video input these approaches introduce noticeable visual artifacts due to lack of global temporal and spatial coherence.

Weighted extrapolation of warps in non-overlap regions ensures temporal stability, while at the same time avoiding visual discontinuities around transitions between views. Remaining global deformation introduced by the warps is spread over the entire panorama domain using constrained relaxation, while staying as close as possible to the original input views.

In combination, these contributions form the first system for spatiotemporally stable panoramic video stitching from unstructured camera array input. They presented an algorithm and processing pipeline for creating panoramic video from camera arrays, which focuses on the particular challenges arising for high resolution video input: spatiotemporally coherent output and globally minimized distortion despite considerable scene motion and parallax. We demonstrated that, using our algorithm, it is possible to create panoramic videos even from larger arrays with challenging unstructured camera configurations and practical issues such as lack of perfect temporal synchronization or rolling shutter.

The aim was to provide a step towards the direction of ubiquitous panoramic video capture, similar to how panoramic image capture has become part of every consumer camera.

[3] H. Guo, S. Liu, T. He, S. Zhu, B. Zeng, and M. Gabbouj, "Joint video stitching and stabilization from moving cameras," IEEE Transactions on Image Processing, vol. 25, no. 11, p. 5491, 2016.

H. Guo, S. Liu, T. He, S. Zhu, B. Zeng, and M. Gabbouj, extended the image stitching to video stitching for videos that are captured for the same scene simultaneously by multiple moving cameras. In practice, videos captured under this circumstance often appear shaky. Directly applying image stitching methods for shaking videos often suffers from strong spatial and temporal artifacts. To solve this problem, they propose a unified framework in which video stitching and stabilization are performed jointly.

Specifically, our system takes several overlapping videos as inputs. The estimation of both inter motions (between different videos) and intra motions (between neighbouring frames within a video). Then, they solve an optimal virtual 2D camera path from all original paths. An enlarged field of view along the virtual path is finally obtained by a space-temporal optimization that takes both inter and intra motions into consideration. Two important components of this optimization are that:

1) A grid-based tracking method is designed for an improved robustness, which produces features that are distributed evenly within and across multiple views

2) A mesh-based motion model is adopted for the handling of the scene parallax. Some experimental results are provided to demonstrate the effectiveness of our approach on various consumer-level videos and a Plugin, named "Video Stitcher" is developed at Adobe After Effects CC2015 to show the processed videos.

By dividing each video frame into cells so as to facilitate the bundled-path approach, our method is capable of handling scenes with parallax. Extensive simulations and some supplementary on various videos are provided to show the effectiveness of our method.

[4] S. Liu, L. Yuan, P. Tan, and J. Sun, "Bundled camera paths for video stabilization," ACM Transactions on Graphics (TOG), vol. 32, no. 4, p. 78, 2013.

S. Liu and L. Yuan, present a novel video stabilization method which models cam-era motion with a bundle of (multiple) camera paths. The proposed model is based on a mesh-based, spatially-variant motion representation and an adaptive, space-time path optimization. The motion representation allows us to fundamentally handle parallax and rolling shutter effects while it does not require long feature trajectories or sparse 3D reconstruction.

Based on the proposed motion representation, they construct a bundle of camera paths, each of which is the concatenation of local homographs at the same grid cell over time. The second component smooths all bundled camera paths as a whole to maintain both spatial and temporal coherences. Furthermore, to avoid excessive cropping/geometrical distortion and approximate cinematography favoured path, they adopt a discontinuity-preserving idea similar to bilateral filtering [Tomasi and Manduchi 1998] to adaptively control the strength of smoothing.

Using image warping techniques for motion representation is an interesting finding in this paper. In the future, they would extend this kind of representation to other video-based applications.

[5] K. Lin, S. Liu, L.-F. Cheong, and B. Zeng, "Seamless video stitching from hand-held camera inputs," in Computer Graphics Forum, vol. 35, no.2. Wiley Online Library, 2016, pp. 479–487.

Images/videos captured by portable devices (e.g., cell phones, DV cameras) often have limited fields of view. Image stitching, also referred to as mosaics or panorama, can produce a wide-angle image by compositing several photographs together. Although various methods have been developed for image stitching in recent years, few works address the video stitching problem. In this paper, they present the first system to stitch videos captured by hand-held cameras. They first recover the 3D camera paths and a sparse set of 3D scene points using CoSLAM system, and densely reconstruct the 3D scene in the overlapping regions. Then, they generate a smooth virtual camera path, which stays in the middle of the original paths.

Finally, the stitched video is synthesized along the virtual path as if it was taken from this new trajectory. The warping required for the stitching is obtained by optimizing over both temporal stability and alignment quality, while leveraging on 3D information at their disposal. The experiments show that the method can produce high quality stitching results for various challenging scenarios.

The results are appealing visually and show that the method is effective under a variety of complex scenarios. To the best of their knowledge, it is the first method that can stitch and stabilize videos at the same time. This initial exploration provides a facility for producing more interesting video content and also points to possible new research directions.

1.7 Organization of the Report

The report is organized into the chapters as follows:

Chapter 2 - System requirement specification: The chapter 2 describes the specific requirement, software and hardware requirements interfaces used in the report. It also gives a brief summary about the current chapter.

Chapter 3 – High level Design: The chapter 3 portrays the Design consideration made, system architecture of implemented system, specification of the implemented system using use case diagram. It also describes module specification, data flow diagram for each and every module. Finally it gives a brief summary about the chapter.

Chapter 4 - Detail design: It depicts the structural chart diagram, detailed functionality and description of each and every module.

Chapter 5 - Implementation: Here, it outlines about the implementation requirements, programming language selection and also coding lines for programming language used in the project. It also presents the pseudo code for main module and sub module.

Chapter 6 – Testing: The unit test cases for each of the modules are discussed in this chapter, along with the referenced snapshots.

Chapter 7 – Results and analysis: The chapter 7 briefs the experimental results.

Chapter 8 - Conclusion and future-enhancement: It provides the conclusion of the implemented system and the future enhancements to be made.

1.8 Summary

The first chapter gives the introduction about the project. It briefs about the developed image compression technique which requires pixel decimation to be done before compression. The motivation to the project is presented in the section1.1. Problem statement is discussed in section1.2 as well as the scope of the project is shown in section 1.3. The objectives are mentioned in the next section 1.4. The section 1.5 records the review of literature, it shares the knowledge gained by referring the related papers and journals required for the project.

Chapter 2

SYSTEM REQUIREMENT SPECIFICATION

A ‘software requirement specification’ (SRS) is a record of description on how a system is expected to perform. It is also a comprehensive description of predetermined purpose and environment of the software that is yet being developed. The SRS explains the functions of the software and how the expected task is performed by the software. The SRS is specified in order to reduce the time and effort put by the developers in order to perform the task and achieve the goals as expected and also reduce the development cost. A better SRS is expected to define, application interaction with hardware of the system, other programs and the end users in the real-world situations.

2.1 Specific Requirements

The essential component required to develop the implemented technique is the MATLAB programming language. The MATLAB platform is designed to solve the engineering problems as well as scientific problems.

2.1.1 MATLAB language

The MATLAB language is matrix-based. Computational mathematics is expressed in a most natural way by using Matlab. To visualize and gain insights from data built-in graphics are used. Prebuilt toolboxes are stored in a vast library. It lets the user to start straight away with the algorithms required to their domain. Experimentation, exploration, and discovery are included in the desktop environment. Thorough testing and design of matlab tools and capabilities are done to perform together.

This language has several numbers of characteristics with high-level programming facility. The characteristics are as follows:

- MATLAB speaks maths
- It is designed for engineers and scientists
- MATLAB toolboxes just work

- It has APPs
- It integrates workflows
- It has been designed to be fast when performing matrix operations
- It is trustworthy
- Scalability

MATLAB helps the user to take the ideas beyond the desktop. On the larger data sets, analyses can be run and, on the clusters, and clouds, scale up can be done. Deployment of algorithms and applications can be done by the user. Few of the important features of MATLAB are as follows:

- Higher level language is designed for scientific computing and engineering computing
- Iterative exploration, design and problem solving works well in MATLAB as desktop environment is tuned to do so.
- To create custom plots, graphics has been designed so as to visualize data and tools
- Apps are developed to perform tasks like curve fiting, data classification, signal analysis and many other domain specific tasks.
- Toolboxes are designed for a vast range of applications in engineering fields and scientific fields
- Tools are designed for application construction with the user interfaces which are accustomed.
- Predefined interfaces are specified to languages and software like C, C++, .NET, Hadoop, Python, SQL and Microsoft Excel.
- MATLAB program sharing with end users is one of the royalty-free deployment options.

MATLAB is a foundation for many products. There are many add-on products available for MATLAB namely image processing, computational finance, parallel computing, signal processing and communication and so on.

2.1.2 Image processing Toolbox

Image processing toolbox performs image processing, it is also known for analysis and algorithm development. For image processing, analysis, visualization and algorithm development, a set of standard algorithm reference, functions, and applications is provided. Image enhancement is an important topic in recent years. Image processing toolbox provided by MATLAB performs image enhancement technique. Various toolbox functions support multi-core processers, GPUs and C-code generation. The capabilities of the image processing toolbox are as follows:

- Image enhancement
- Image analysis
- Image segmentation
- Large image processing and acceleration
- Noise reduction

Across industry and academics, there are millions of MATLAB users. MATLAB users are from various fields of science, economics and engineering.

2.2 Hardware Requirements

➢ Processor	: Pentium IV
➢ RAM	: 8 GB
➢ Speed	: 2.4 GHZ
➢ Secondary Device	: 1 GB

2.3 Software Requirements

➢ Programming Language	: MATLAB
➢ Tool Used	: MATLab 16.0
➢ Operating System	: Windows XP

2.3.1 Functional requirements

- High resolution images must be provided as input as it tends to undergo pixel decimation while the low-resolution images can directly be compressed by JPEG compression
- The configuration of the system used must be of higher end.

2.3.2 Non-functional requirements

- Time efficiency has to be maintained for compression and decompression
- Robustness of the system has to be more

2.4 Summary

The chapter 2 is all about the system requirement. Section 2.1 presents specific requirement of the implemented system. Section 2.2 mentions hardware requirement and 2.3 mentions the software requirements. Subsections then describes about the functional and non-functional requirements.

Chapter 3

HIGH LEVEL DESIGN

High level design is the design which is plotted for designing the software related items. Here the complete system design is generated and how the modules, sub modules and the flow of the data between them are done is initialized. It explains the architecture which is used for development of the system. It is the base phase for any of the actual implementation process. The errors done here will be reflected in the coming processes.

3.1 Design Consideration

The design consideration is formulated so as to give designers an idea of applying the universal accessibility design principles. It briefs about how the system should behave for the boundary environments and what actions should be performed if any unusual case happens. Some of the design considerations are input video, unified video stitching and optimization, multi video background extraction and false features match extraction

A. Unified Video Stitching and Stabilization Optimization

Without loss of generality, the aim is to stitch two videos VA and VB captured by freely-moved hand-held cameras to produce a wide-angle video. Each camera captures a part of the scene with its own camera path. In the following, we introduce how to combine stitching and stabilization into a unified optimization framework.

B. Multi-Video Background Identification

The unified optimization above relies on two kinds of feature matches:

(1) Temporal feature matches between consecutive frames of an input video.

(2) Spatial feature matches between corresponding frames of two input videos.

The former matches are used to estimate camera paths of input videos, while the later ones are used to stitch two input videos together. To make our main component more robust, we require that the feature matches used are all from background regions. This is mainly due to two reasons. First, compared to foreground, the motion of background can better represent the motion of camera, as background is usually static but foreground may have its own movement. Second, it is difficult to stich both foreground and background when their depths are very different.

Thus, we try to stitch background only, with foreground objects being processed later. The problem now becomes how to find background for each input video, and also common background of all the input videos. To solve the background identification problem, we adopt the recently developed method which is composed of two stages: seed identification stage and refinement stage. To make the method more suited to our problem, we improve it in two aspects. First, we find that the refinement stage of the method is very simple, and not always effective. We propose a more effective refinement algorithm. Second, the method is designed for a single video. We extend the method to multiple videos.

C. False Feature Matches Elimination

Here we have a set of feature matches, including both temporal and spatial ones in background regions. When trying to obtain these feature matches, we set a large threshold in the SIFT algorithm, producing more feature matches but inevitably introducing false feature matches. A feature match is said to be false, meaning the pair of features are not from the same position of the same object. Hence, we propose a method to reject false feature matches in the spatial set, as stitching them together would produce wrong panoramic results.

3.2 System Architecture

System architecture is such a model that defines the structure of the system, its behaviour and many more views of a system and also explains overview of system. It explains the working of the overall system, how the data flows in the system, the modules and the techniques that should be implemented in predefined order.

Fig 3.1 shows the system architecture for dynamic video stitching. The system architecture diagram of the implemented video stitching technique mainly consists of three modules. The three different modules are background identification, Unified stitching and stabilization and false feature match elimination.

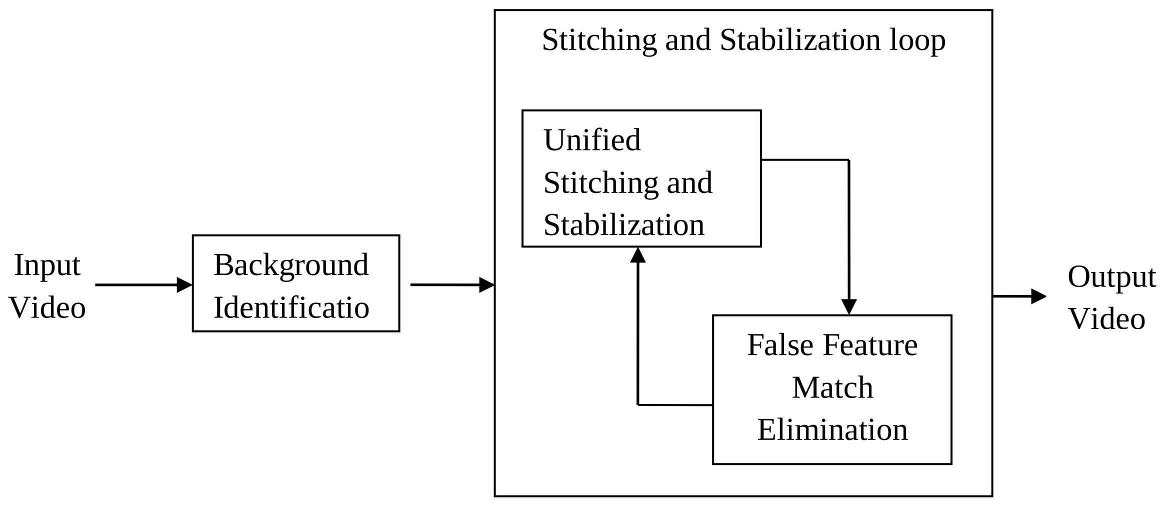

Stitching and Stabilization loop
Unified Stitching and Stabilization
Input Video
Background Identificatio
False Feature Match Elimination
Output Video

Fig 3.1: System architecture for the method of Dynamic Video Stitching Via Shakiness Removal

Input to this system is an video. Video is captured using mobile camera. The core of the method is the unified video stitching and stabilization optimization, that is the main component. Without any other components, the main component itself can produce a panoramic video. The second and third components is used to make the optimization more robust. Among them, the second component extracts background region for each input video, and also find the common background region of all the videos.

In this way, we can input the features only in the background regions into the main optimization. However, we do not expect the extracted backgrounds to be completely accurate, and the features matches in background themselves may be wrong too. The third component which helps reject false matches. The main and third components is encapsulated into a loop scheme and is run iteratively to obtain the optimal results. The second component performs like a pre-processing stage. In contrary, the third component works like a post-processing stage.

1. Image Stitching

Nowadays, image stitching is a practical technique widely used in modern smart phones and has many applications such as collaborative photography. Early methods used a single homograph to align two images. However, they demand the captured scene to be planar, otherwise would fail due to apparent parallax. According to how they handle parallax, recent image stitching works can be roughly divided into two categories. Warping-based methods use more than one model to represent relationships between images.

For example, Gao et al. computed two homographs to stitch images with only two major planes. Zaragoza et al. divided images into grids and computed spatially-variant homographs for these grids. These methods can handle small parallax, but would still retain misalignment if parallax is very large. To handle more severe parallax, seam-based methods find an energy minimum seam in the overlapping region of two images which is viewed as the stitching boundary. We also encounter the problem of large parallax.

To solve it, we explicitly separate objects with different depths, i.e., foreground and background objects. We then stitch background regions by the idea of warping-based methods and use seam-based idea to composite foreground objects.

2. Video stitching

Video stitching has received much less attention than image stitching. Different approaches have been proposed for different camera settings. For example, some works, were designed for static surveillance cameras, while some for moving cameras but the relative geometries between the cameras are still fixed. To stitch videos captured by these cameras, the pose relationships between the cameras can be pre-calibrated to stitch every pair of frames globally, followed by local adjustments such as structure deformation or spatiotemporal consistent warping to eliminate small deviations. R2, R5 and R7 camera heads used in Google Street View are examples of such kind of camera arrays.

Some other works were designed for independently moved mobile devices. The difference is that the works did not consider stabilizing input videos, while some methods took stabilization into consideration for the aim of better stitching. Our work also considers stabilization while stitching, but we propose a different formulation, and our approach is more robust.

3. Video Stabilization

Videos captured by hand-held devices are usually very shaky, which makes the stitching between these videos more challenging. To obtain a good stitching result, stabilization is necessary. There are many video stabilization methods which can be divided into three categories:

3D methods, 2.5D methods, and 2D methods. As well documented in the literature, 3D methods are based on 3D reconstruction which is brittle and time-consuming. 2.5D methods requires long feature trajectories which are suitable only for professional devices but not for our hand-held consumer cameras such as smart phones.

Therefore, we choose the 2D based bundled camera paths framework for our purpose of stabilization, since it is easy to integrate stitching constraints into the framework. Recent progresses in the literature of video stabilization focus on algorithm acceleration, such as Zhang et al. achieved a 10 times speedup by casting video stabilization as finding geodesics in transformation space, and Liu et al. proposed a predicted adaptive path smoothing approach which can stabilize videos online. We retain the acceleration of simultaneous video stitching and stabilization as a future work.

4. Video Stitching and Stabilization

Very few works considered video stitching and stabilization together. In some works, videos were firstly stitched, and then stabilized. Lin et al. did it in an opposite way, i.e., videos were firstly stabilized, and then stitched. The work by Guo et al. is the first to combine video stitching and stabilization into a joint optimization framework. Their work is also based on bundled camera paths stabilization framework. But they optimized stabilization variables only, while we optimize both stabilization and stitching variables.

5. Background Identification

Since it can help recover more accurate camera motions, finding background of a video is a fundamental task for video stabilization, video resizing, etc. For example, to enhance stabilization effects, background was identified manually. We try to find background regions automatically, and there are several technique candidates. Most of existing approaches utilize RANSAC. But the effectiveness of RANSAC-like techniques may be easily interfered by large foreground objects.

Another choice is to estimate depth of scenes by camera calibration and multi-view stereo, and then use the computed depth to select background regions. But depth reconstruction itself is a challenging task. Koh et al. proposed a bi-layer clustering method to find background trajectory cluster based on the location and colour of trajectories.

However, it is a little difficult to balance between the facts of location and colour. Recently, Zhang et al. purposely developed a background identification algorithm based on motion segmentation. We employ this method since it is effective for dynamic videos taken by hand-held devices. However, the method accepts only a single dynamic video. We improve it in several aspects to find common background of multiple input videos.

3.3 Specification using Use case diagram

The graphical depiction of the interactions among the system elements is defined as use case diagram. It is a methodology that is of great significance in system analysis for the identification, clarification and organization of system requirements. The requirements here are mostly the design requirements. Therefore, during system analysis stage, which is done to gather the functionalities the use cases are prepared and the actors are identified. It describes the communication between the actors and the systems during execution of the system.

Fig 3.2 shows the use case diagram for overall system of image stitching using Scale Invariant Feature Transform (SIFT) match technique. Here, in the use case diagram there are four sets of use cases and one actor. Every use case represents the functionality that the system provides. The actor here is involved in the use case by providing the input image as input. The input image can be of any size. The use case SIFT match will take the input image and find out the descriptors and the locations provided in the images which are converted into frames. Then gets a homograph where the images are where the matching location of image1 and image2 are taken and finally the projection is done to project those images based on the respective value of matrix and distance.

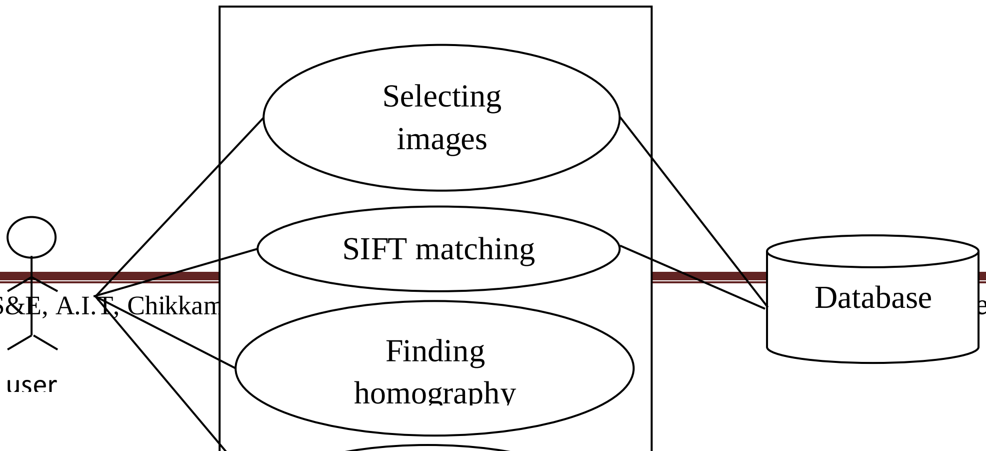


Fig 3.2: Use case diagram for the overall method of image stitching and Stabilization

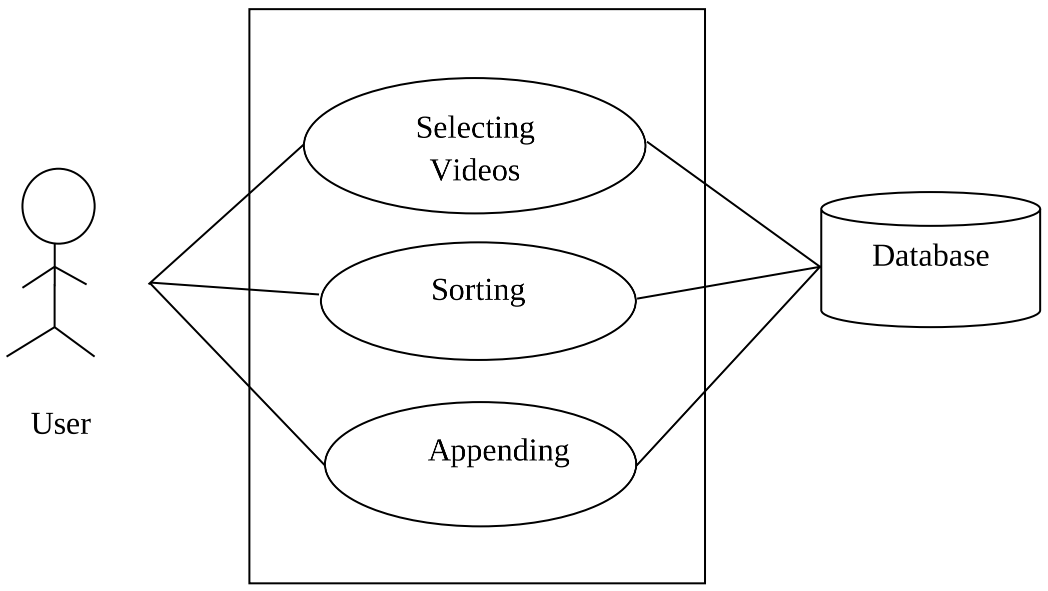


Fig 3.3: Use case diagram for the unified video stitching

Fig 3.3 shows the use case diagram for video stitching. There are three sets of use cases and one actor. Every use case represents the functionality that the system provides. The actor here is involved in the use case by providing the video as input. The input video can be of size 4Mb to 20Mb. The SIFT match will take the input video and find out the descriptors and the locations provided in the videos which are converted into frames. Sorting is done using the nearest neighbourhood method and finally the lines are drawn using appending the images. The mosaic video is formed.

3.4 Module Specification

The module specification is a definitive record of each module which contains the description of the process and method used in each of the modules. In the implemented method of image compression using pixel decimation, there are four modules such as pixel decimation, JPEG compression, JPEG decompression, and the image upscaling. Use case diagrams are used to explain each of these modules in detail.

3.4.1 SIFT (Scale Invariant Feature Transform)

Name of the module: SIFT

Actors: SIFT and imStitch.

Functionality: The main functionality of the system is to take the input image and provide a descriptors and locations.

Description of SIFT: Fig 3.2 and Fig 3.3 shows the use case diagram for SIFT match and mosaic images. Here the actor who is a user in this use case will input an image which has to be stitched according to the matching locations and descriptors provided by the algorithm.

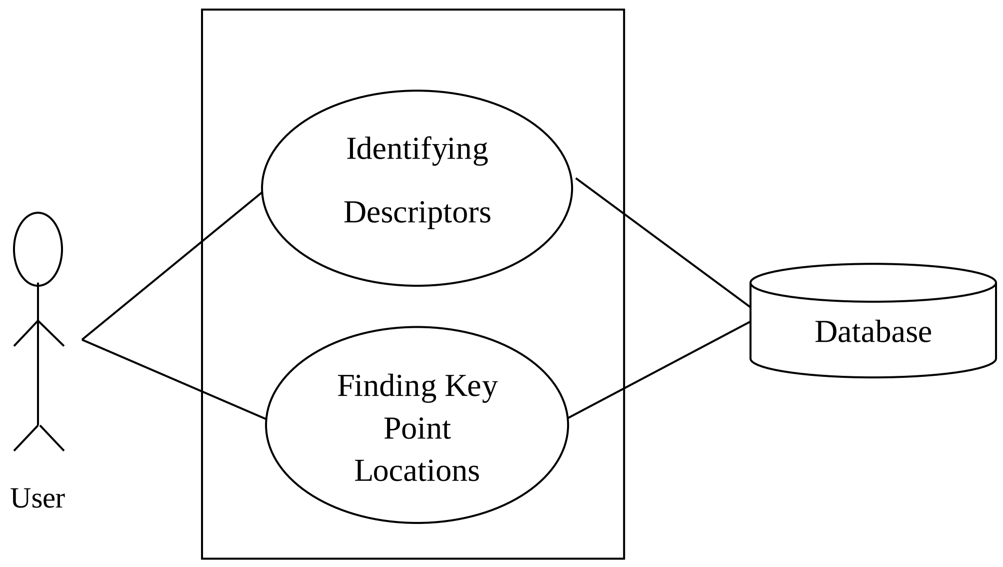


Fig 3.4: Use case diagram of SIFT

3.4.2 Video stitching

Videos captured by hand-held devices are usually very shaky, which makes the stitching between these videos more challenging. To obtain a good stitching result, stabilization is necessary. There are many video stabilization methods which can be divided into three categories: 3D methods, 2.5D methods, and 2D methods. As well documented in the literature, 3D methods are based on 3D reconstruction which is brittle and time consuming. 2.5D methods requires long feature trajectories which are suitable only for professional devices but not for our hand-held consumer cameras such as smartphones.

Therefore, we choose the 2D based bundled camera paths framework for our purpose of stabilization, since it is easy to integrate stitching constraints into the framework.

3.4.3 Stabilization

Very few works consider video stitching and stabilization together. In videos were firstly stitched, and then stabilized. Stabilization is also based on bundled camera paths stabilization frame work. But they optimized stabilization variables only, while we optimize both stabilization and stitching variables. Videos were firstly stabilized, and then stitched. But they optimized stabilization variables only, while we optimize both stabilization and stitching variables.

3.4.4 Background Identification

To enhance stabilization effects, background was identified manually. We try to find background regions automatically, and there are several technique candidates. Most of existing approaches, utilize RANSAC. But the effectiveness of RANSAC-like techniques may be easily interfered by large foreground objects. Another choice is to estimate depth of scenes by camera calibration and multi-view stereo, and then use the computed depth to select background regions. But depth reconstruction itself is a challenging task. A bi-layer clustering method to find background trajectory cluster based on the location and colour of trajectories.

However, it is a little difficult to balance between the facts of location and colour. Recently, purposely developed a background identification algorithm based on motion segmentation. We employ this method since it is effective for dynamic videos taken by hand-held devices. However, the method accepts only a single dynamic video. We improve it in several aspects to find common background of multiple input videos.

3.5 Data Flow Diagram

‘Data flow diagram’ is an illustration of how data is processed in the system in terms of input and output. It is generally a graphical depiction of the stream of information. In the data flow diagram, the flow of information is focused, source of the data, destination of the data and the data storage process. Data transformation takes place at each step before the data is moved further to next step.

The features considered here is edges. The Histogram Transformation. The edges are detected and Trellis graph is built for each frame and is represented in the form of matrix. In the Trellis graph the edges are given weight. For ease description inner edges are linked with Trellis graph. The edges with same weight in the frames of Trellis graph and are considered for stitching. And the stitched frames obtained are mosaic images.

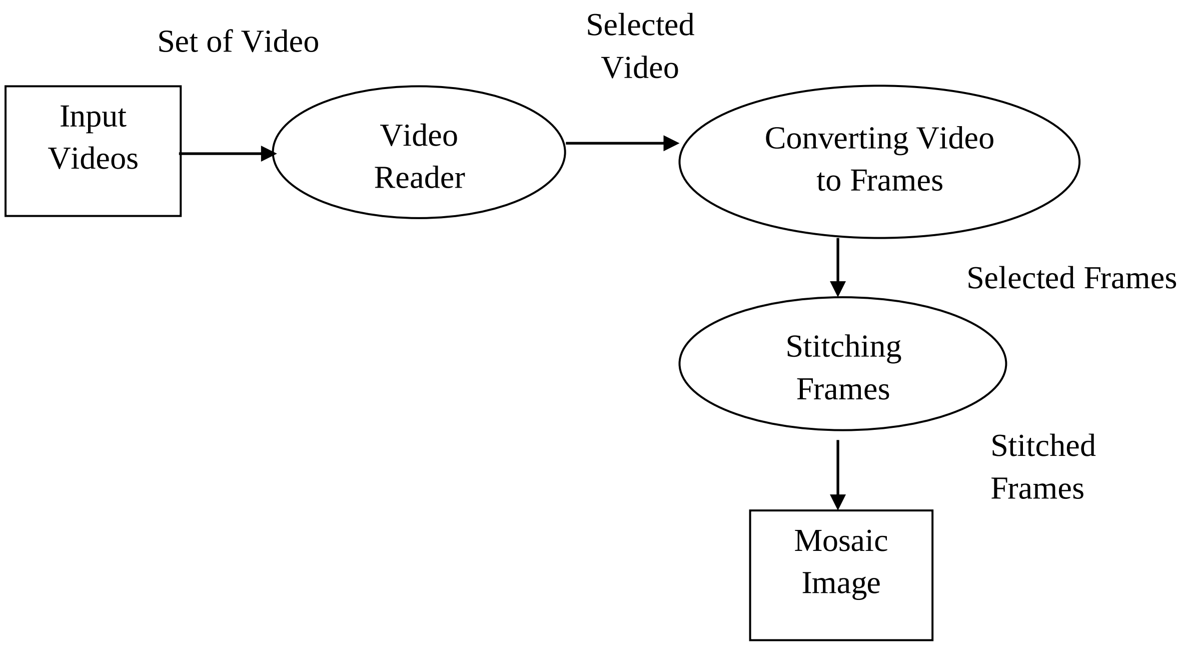


Fig 3.5: Level 0 data flow diagram of the background identification technique

Fig 3.5 illustrates level 0 data flow diagram usage that presents the overview of the architecture of the performed background identification method of dynamic video stitching. Here, we try to find background regions by bi-layer clustering method to find trajectories by location and color of trajectories.

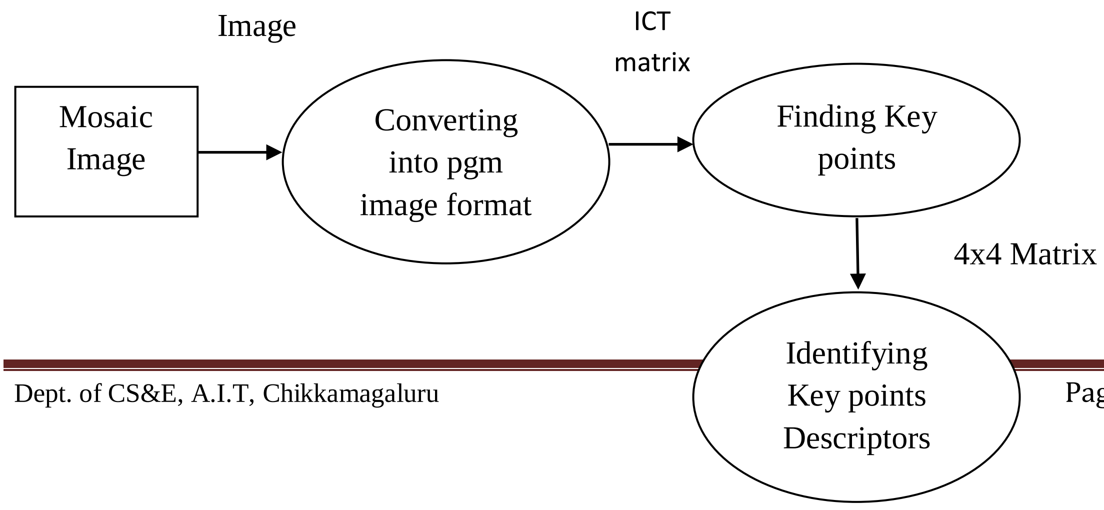


Image
ICT matrix
Mosaic Image
Converting into pgm image format
Finding Key points
4x4 Matrix
Identifying Key points Descriptors

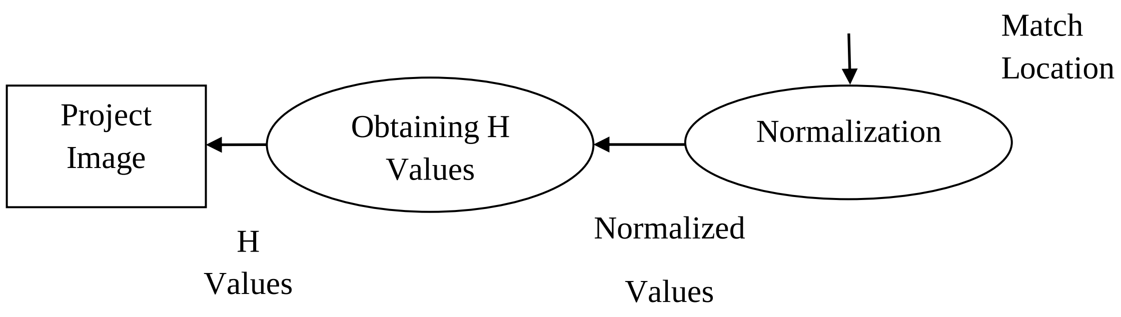


Fig 3.6: Level 1 data flow diagram of Unified Stitching of Videos

Fig 3.6 illustrates level 1 data flow diagram for the video stitching technique using Unified stitching and stabilization of videos to show the detailed view of the architecture of the implemented system of Dynamic Video Stitching. The mosaic image is taken as input, the image is converted into PGM format (Portable Gray Map). The pgm values are represented in the form of gray scale matrix and ICT (Inverse Cosine Transform) matrix. The key points are the key values and these points are descripted and match location are taken found. The matched points are normalized and corresponding H values (Homograph Values) found. The homograph values are projected as images.

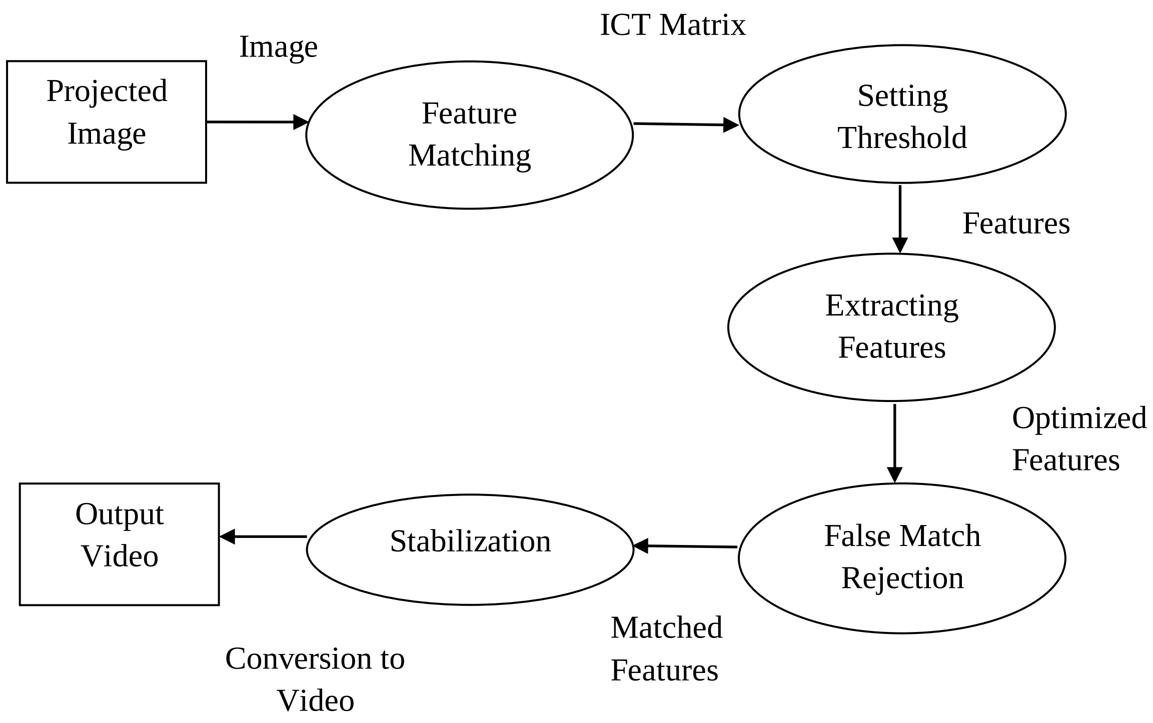


Fig 3.7: Level 2 data flow diagram of Stabilization of the video

Fig 3.7 illustrates level 2 data flow diagram usage that presents the overview of the architecture of the performed stabilization method of dynamic video stitching. The stitched video is given as input. Features such as image descriptors, location of images in pgm file are matched. Feature matching is done by Nearest Neighbourhood Method. Image descriptor is taken in inverse matrix form and key points with same cosine values are matched. After feature matching, threshold is set. The cosine values of key points less than the threshold values are removed and the features are optimized. The cosine values with the false features are eliminated and the features are stabilized and the final output video is obtained.

3.6 Summary

In the third chapter, high level design of the developed method has been discussed. Section 3.1 presents Section 3.1 presents the design considerations for the project. The system architecture of the video stitching technique is illustrated in section 3.2, the next section 3.3 describes specification using use case diagram. Section 3.4 describes module specification using use case diagram for pixel background identification, unified video stitching, false feature elimination. The data flow diagram for the system is explained in section 3.5 followed by data flow diagram for each module.

Chapter 4

DETAIL DESIGN

4.1 Structural Chart

A structural chart is a chart that represents the breakdown of a system to its lowest levels that are easily manageable as explained in software engineering and organizational theory. The arrangement of program modules is done in the form of a tree. Each program module is represented in a rectangular shaped box. The tree structure visualizes the relationships between modules.

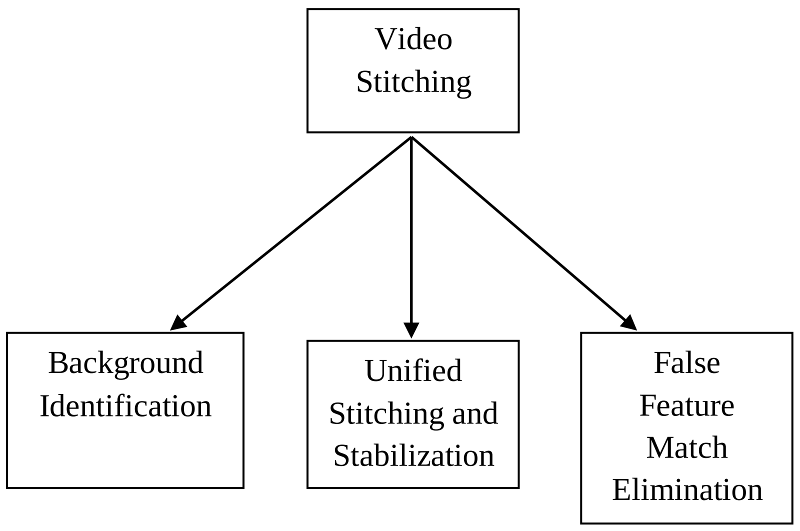

Video Stitching
Background Identification
Unified Stitching and Stabilization
False Feature Match Elimination

Fig 4.1: Structural chart for dynamic video stitching via shakiness removal

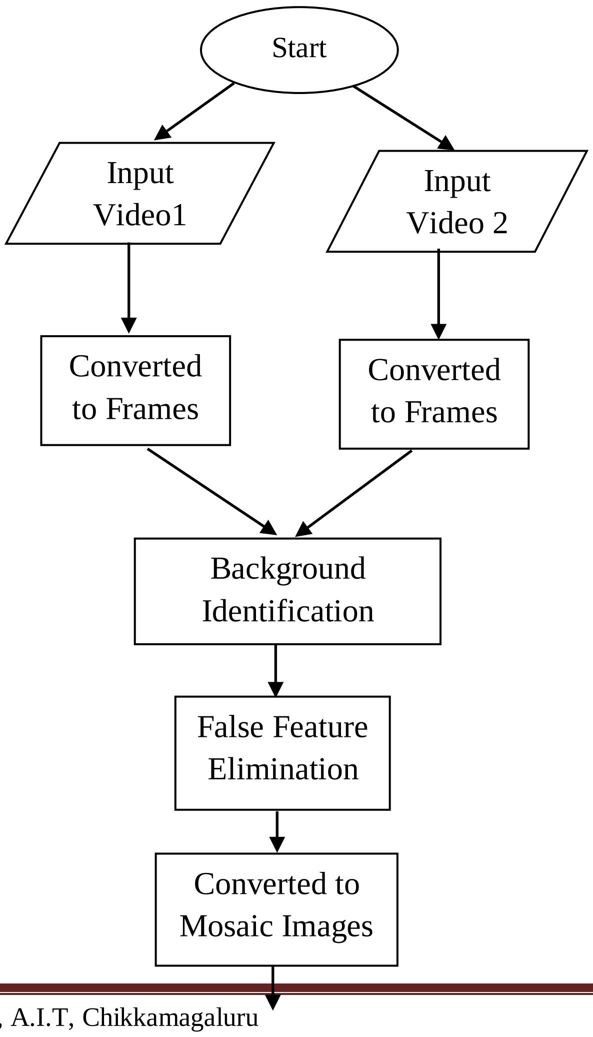
Structural chart of the implemented system of image compression using pixel decimation is illustrated in the Fig 4.1. The system encloses the three main modules which are Background Identification, Unified Stitching and Stabilization, False Feature Match Elimination.

In the first module that is Background Identification, the Identification relies on two j=kind of feature matches (a) temporal features matches between consecutive frames of two input videos (b) spatial feature matches between corresponding frames of two input videos. The former matches are used to estimate camera paths of input videos. The feature match is used for all background regions.

The next module is the false feature match elimination module which comprises of both temporal and spatial ones in background regions. A feature is said to be false, meaning pair of features are not from the same position of the same object. The result is, false feature rejected in the spatial set, as stitching them together. The last module is the unified stitching and stabilization module which gives a stitched video as output.

4.2 Detailed Description of Main Module

Overall system flowchart is as shown in the Fig 4.2. The system implementation starts by reading the two input videos. The input video should be of mp4 format. The next step is the conversion from video to frames. The background identification method is used to identify the background from foreground**.**


Start
Input Video1
Input Video 2
Converted to Frames
Converted to Frames
Background Identification
False Feature Elimination
Converted to Mosaic Images

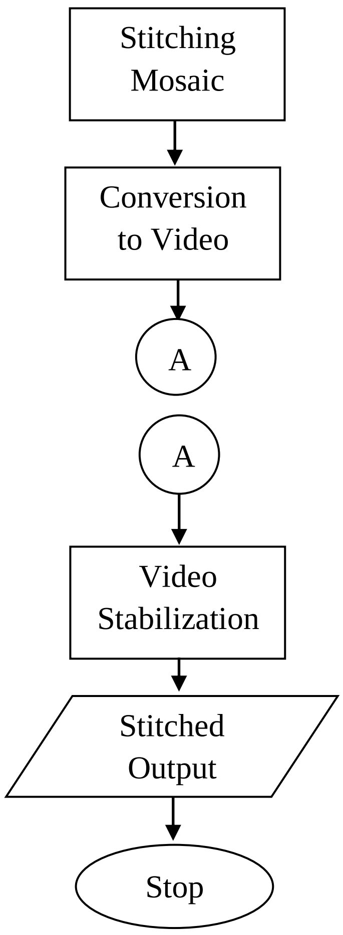


Fig 4.2: Overall flowchart of Dynamic Video Stitching

False feature elimination in which the unwanted background is eliminated. The false feature elimination technique outputs mosaic images. The elimination takes place on 712x712 matrix. The False Feature Elimination technique implemented here uses SIFT algorithm technique which eliminates the unwanted matches which is presently in the spatial features of background identification.

The second process is the Unified stitching of mosaic images, in this step it stitches the mosaic image which is converted in the previous step. The foregoing process is the stitching of mosaic images. The next step is conversion of stitched mosaic images into video. Next, the video is stabilized for removal of shaky paths. The resultant is the stitched video in panoramic view with higher resolution visual quality.

4.3 Detailed Description of Each Module

4.3.1 Detailed Description of Background Identification

To identify the background of the videos, background identification process has been used. The method used in the implemented system is the Trellis process. The refinement stage of the method is very simple, and not always effective. We propose a more effective refinement algorithm designed for a single video. Next, it extends the method to multiple videos.

The Fig 4.4 shows the flowchart for the background identification process. The input video is divided into a series of overlapped temporal windows, with each window having 40 frames, and two adjacent windows having 20 overlapped frames. In the second row, each temporal window is clustered into several regions based on the feature trajectories. Finally, in the bottom row, a trellis graph is built: each column of the graph corresponds to each temporal window; a column contains several nodes that correspond to the segmented regions of the corresponding temporal window; and edges are added between nodes of adjacent columns when there exist at least one feature trajectory that passes through the corresponding regions.

The weight of an edge is defined as: x

$$\omega\ i, j\ k, k+1 = \exp(-\alpha Si, j\ k, k+1) \cdot ri, j\ k, k+1 \qquad \text{Eq. 4.1}$$

where k and k + 1 indicate trellis columns, i and j are indices of nodes in the columns, Si, j k,k+1 is the number of common feature trajectories passing both the regions of nodes ni k and nj k+1, and ri, j k,k+1 is the rank of the matrix formed by the set of common trajectories (lower rank means the two nodes are more likely being background. The parameter α is used to adjust the influence of Si,j k,k+1 which is set as 0.01.Given the well-defined trellis graph, the energy-minimum path of the trellis graph can be computed efficiently by dynamic programming. The video regions correspond to the nodes of the energy-minimum path are viewed as background regions.

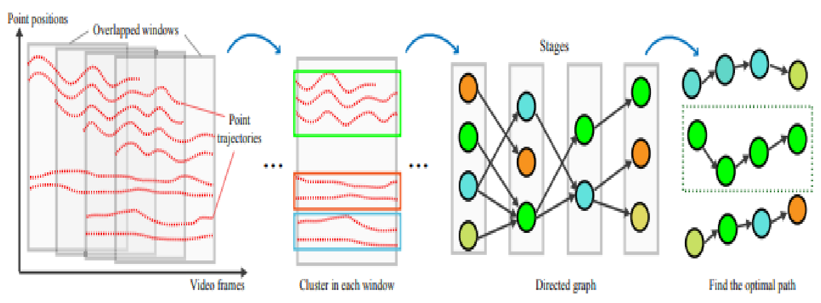


Fig 4.3: Feature trajectories of a video

The fig 4.3 shows the feature trajectories is found by KLT tracker and the video is divided into overlapped temporal windows. Middle row: Each temporal window is segmented into several clusters by the feature trajectories in the window. Bottom row: A trellis graph is constructed from the temporal windows.

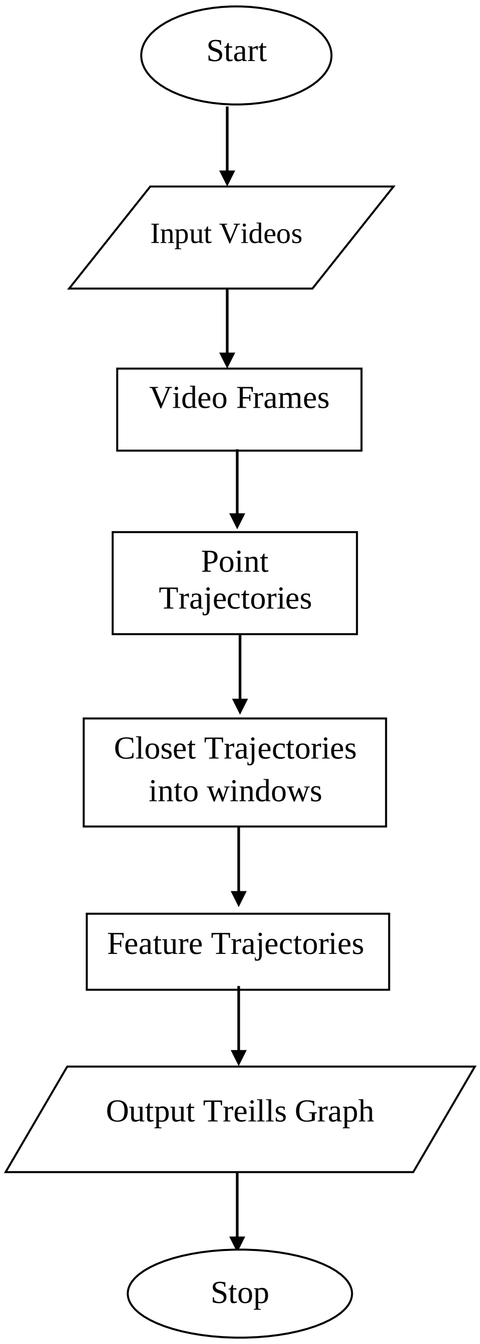


Fig 4.4: Flowchart for Background Identification

Fig 4.4 shows the flowchart for background identification. The given input video are converted to video frames. From the video frames point trajectories are obtained. The point trajectories give the path for the moving objects in the video frames in the form of matrix. The closet trajectories are the distance between point trajectories. The nearest point trajectories are considered for background identification. The closet trajectories are formed by clustering technique. In clustering technique, it gives the path in which the matching features of the background are present. Here features are represented in the form of matrix. The feature trajectories contain only the closet trajectories which have matching values in the matrix. Based upon the obtained feature trajectories the Treills graph drawn. The Treills graph gives the background region. The background region is separated from foreground regions

4.3.2 Detailed Description of False Feature Match Elimination

False Feature Matches Elimination is a set of feature matches, including both temporal and spatial ones in background regions. When trying to obtain these feature matches, we set a large threshold in the SIFT algorithm, producing more feature matches but inevitably introducing false feature matches. A feature match is said to be false, meaning the pair of features are not from the same position of the same object. Hence, we propose a method to reject false feature matches in the spatial set, as stitching them together would produce wrong panoramic results.

Specifically, let $\{vk\ (t)\}$ be the set of spatial feature matches, where $vk(t)$ represents the k pair of features in the tth frame. We use them to compute our stitching and stabilization variables using the below equation.

$$\Theta = (PA\ i,\ PB\ i,\ H)$$

With Θ, we can compute the distance $ek(t)$ between the transformed features of feature match $vk(t)$. We compute the average distance ξ of all the feature matches, and any feature match $vk(t)$ whose distance ei is larger than $3\times\xi$ is viewed as a false feature match. It involves unified optimization and the false match rejection scheme in a loop. After eliminating false matches, we obtain a smaller set of spatial feature matches. We then input them to the unified optimization to compute a new, Θ and use to compute a new set of feature matches. The loop stops after 10 iterations or until the feature set is not reduced.

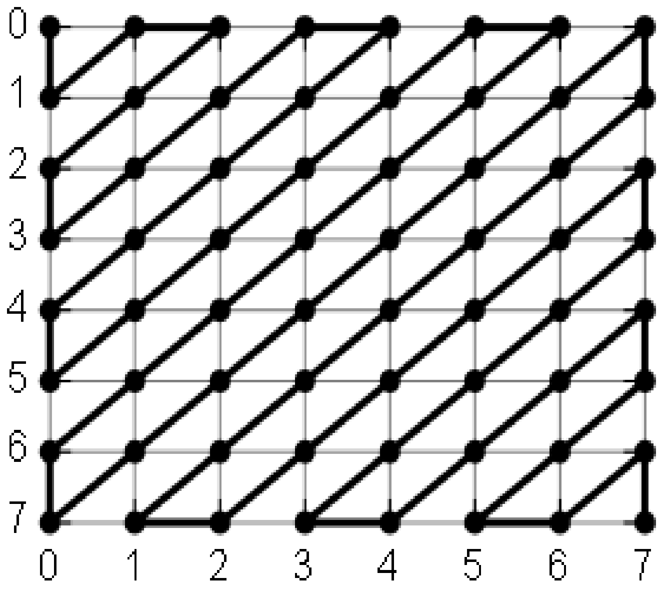


Fig 4.5: Input matrix values of size 8 x 8.

$$\begin{bmatrix}
89 & 59 & 54 & 75 & 84 & 66 & 33 & 25 \\
88 & 74 & 72 & 71 & 85 & 91 & 70 & 37 \\
90 & 87 & 91 & 98 & 88 & 93 & 94 & 74 \\
90 & 87 & 91 & 88 & 95 & 94 & 74 & 73 \\
98 & 82 & 89 & 91 & 97 & 54 & 24 & 23
\end{bmatrix}$$

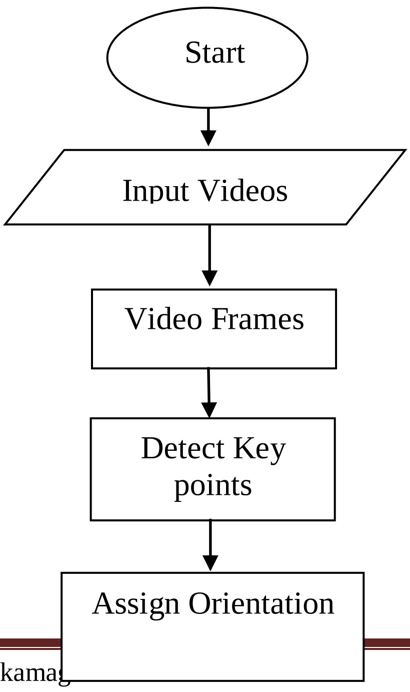
$$\begin{array}{cccccccc}
87 & 77 & 87 & 60 & 29 & 33 & 43 & 50 \\
75 & 76 & 96 & 86 & 90 & 94 & 78 & 50 \\
73 & 15 & 80 & 33 & 25 & 93 & 64 & 37
\end{array}$$

Fig 4.6: 8 x 8 Matrix Features for Stitching

$$\begin{bmatrix}
89 & 75 & 84 & 66 \\
88 & 74 & 85 & 91 \\
91 & 98 & 93 & 94 \\
90 & 91 & 95 & 94
\end{bmatrix}$$

Fig 4.7: Key Points for Stitching

Fig 4.6 represents the matrix obtained from the given frames of the input videos, which is formed by imread function, a built function in MATlab. The Fig 4.7 represents the key points obtained from the matrix Fig 4.6.



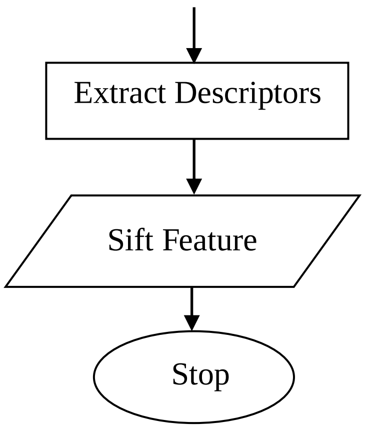
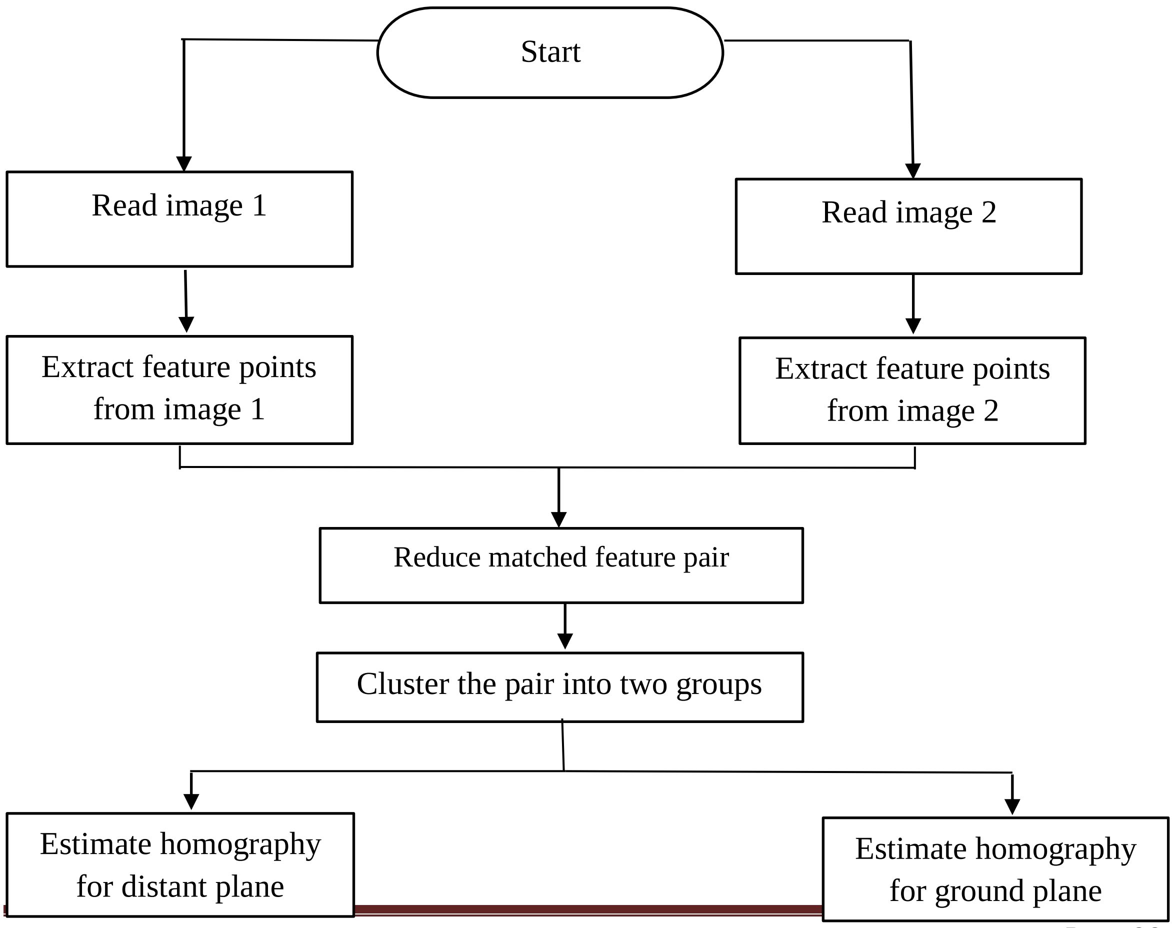


Fig 4.8: Flowchart for False Feature Match Elimination

Fig 4.8 illustrates the flow of the False Feature Match Elimination, where the input videos are converted into frames and the key points of those frames are detected using which appropriate key points are taken into considerations.

4.3.3 Detailed Description of Unified Stitching and Stabilization

The below Fig 4.10 is the flowchart of unified stitching process, the obtained frames will be stitched by extracting the feature points.


Start
Read image 1
Read image 2
Extract feature points from image 1
Extract feature points from image 2
Reduce matched feature pair
Cluster the pair into two groups
Estimate homography for distant plane
Estimate homography for ground plane

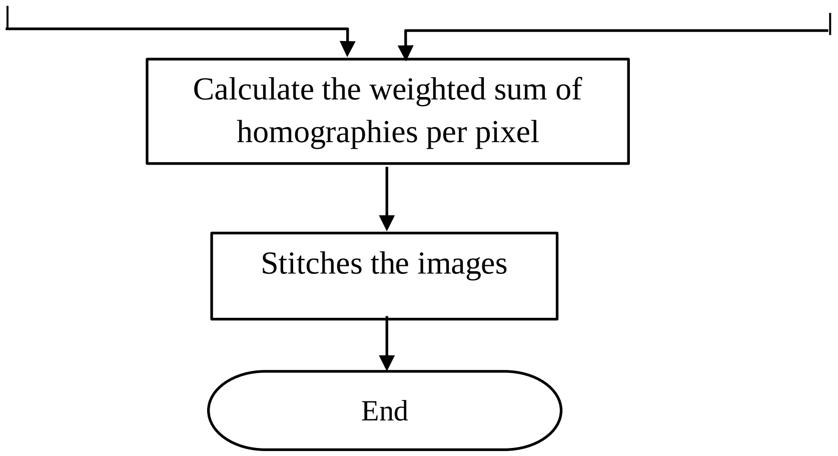
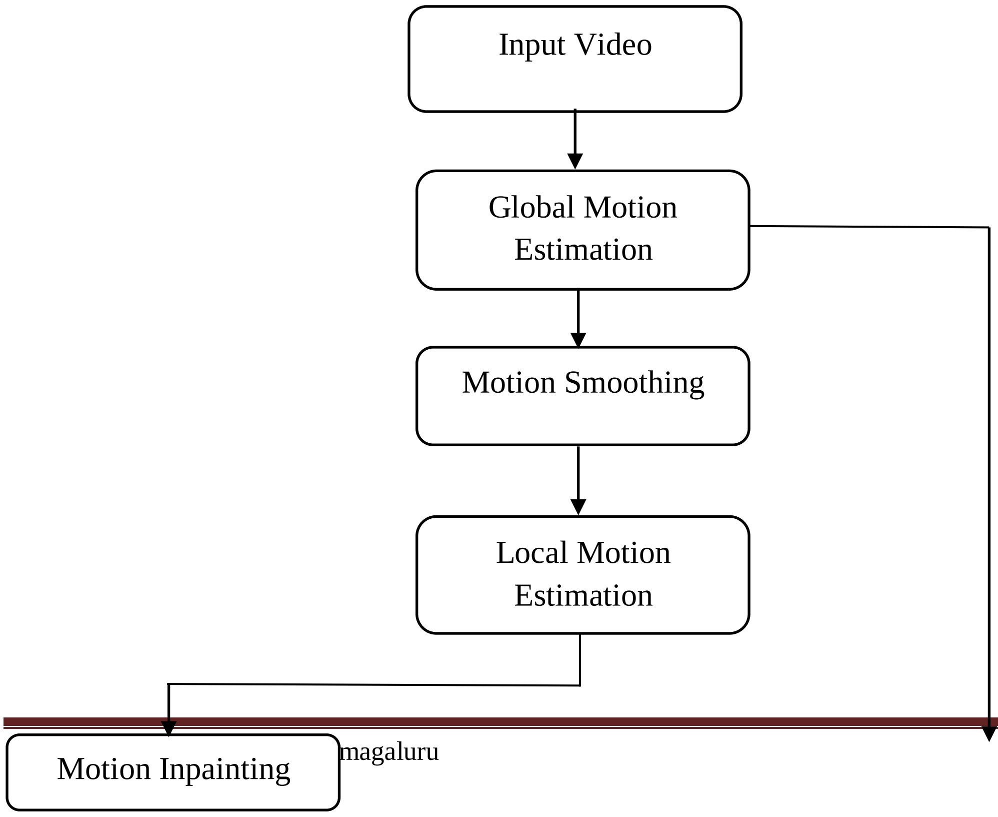


Fig 4.9: Flowchart for Unified Stitching

Fig 4.9 represents the stitching process, where it reads two images that are to be stitched. Then the feature points are extracted from both images, later produces the matched feature pair to form the cluster pair of two groups. The next step is to estimate the homography for distant plane and ground plane. Finally, the weighted sum of homograhies per pixel is calculated and stitches the images.

The overall unified stitching process is carried out without loss of generality, the aim is to stitch two videos VA and VB captured by freely-moved hand-held cameras to produce a wide-angle video. Each camera captures a part of the scene with its own camera path. In this it introduces to combine stitching and stabilization into a unified optimization framework.



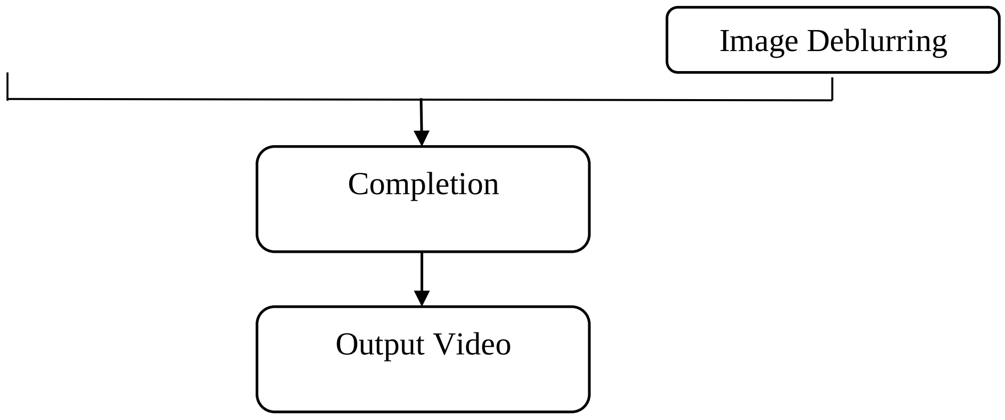


Fig 4.10: Flowchart of Stabilization

Fig 4.10 represents the flow of image being stabilized, where the input video is taken, then global motion of those input videos is estimated and later the estimated motion is smoothened. The local motion estimation is done in order to make motion inpainting and image deblurring. Hence, the stabilized video is obtained.

To stabilize an input video, input video is divided into grids spatially (as shown in Fig.4.5.). For each grid i and at frame t, a transformation matrix Fi(t), i.e., a homograph, is computed, which represents the camera motion of that grid from frame t to t + 1.

The camera path of the grid i at frame t, denoted by Ci(t), is then the accumulation of the homographs of the frames before (please refer to the inset for the relationships of these notations):

$Ci(t) = Fi\,(t)\cdot Fi\,(t-1)\cdots Fi\,(1), 1 \leq T, 1\ i \leq m^2$ …. Eq. 4.2

where

T = total number of frames

m2 = is the size of the m-by-m grids.

Ci (1) = an identical matrix.

In other words, $Ci = \{Ci\,(t)|1 \leq T\}$ is the camera path of grid i. To stabilize grid i, we just need to compute a smooth camera path

$Pi = \{Pi\ (t)\ |1 \leq t \leq T\}$

by optimizing the following equation:

$$© = (Pi) = \sum (\| Pi (t) - Ci (t)\|)^2 + \lambda\cdot \sum w_{t,r} \| t\ \omega t,r \cdot Pi (t)- Pi (r) \qquad \text{Eq. 4.3}$$

where the data term $\| P_i (t) – C_i (t) \|^2$ enforces the new path to be close to the original one, and the temporal smoothness term $\| Pi (t)- P_i (r) \|^2$ works like a filter to avoid shakiness, λ , ωt and r weights to balance the two terms, and t represents adjacent frames (±30 in our experiments) of frame t. If all the grids are taken into consideration, the equation to optimize becomes:

$$EStable\ P = \sum_i^t (P) + \sum \| Pi (t) - Pj (t) \|^{2.} \qquad \text{Eq. 4.4}$$

where $P = \{Pi|1\leq i \leq m^2\}$, j ∈ N(i) means grid i and j are neighbouring grids, and the newly added term Pi(t)−Pj(t)2 is a spatial consistency term to make spatially adjacent grids undergo similar adjustments.

After implementation, the better stabilized results can be achieved if changing Eq. 4.2. In Eq. 4.3, letting Pi (t) directly equal to Pj (t) performs not that good to preserve spatial consistency. Instead, we change it to $\| Bi\ (t) – Bj\ (t) \|^2$. Eq. 4.5

where Bj (t) = Pi (t) · Ci (t) − 1 is the transformation that finally renders the stabilized position of grid i. The reason that we choose Bi(t) rather than Pi(t) is illustrated in the right inset. A and B are two vertices of original grid i. C and D are two vertices of grid j adjacent to i. A coincides with C, and B coincides with D. After stabilization transformation, A, B, C, and D become A′, B′, C′, and D′, respectively. To keep the first-order continuity of the grid mesh, we need A′ to coincide with C′, and B′ to coincide with D′. Since it is Bi (t) and Bj (t) that transform the two grids to their target positions, we therefore choose to let Bi (t) equal to Bj (t) directly.

Here we combine video stitching and stabilization together. Recall that we have two input videos VA and VB. To stabilize and stitch them, we introduce three variables:

(1) PA: the variable used to stabilize video VA;

(2) PB: the variable used to stabilize video VB, and

(3) H: a single homograph used to stitch video VA and VB. To directly get the optimization function that finds the best values for the three variables:

$$E(P^a, P^B, H) = Estable(P^A) + Estable(P^B) +\beta \cdot E^{stitch(PA,} PB, H) \qquad \text{Eq. 4.6}$$

Here, we employ the bundled camera paths framework to perform stabilization, i.e., the Stable of Eq. 4.1 is defined as Eq. 4.6 in the above subsection. As for stitching, we us assume a set of spatial feature matches (computed by SIFT) between every pair of corresponding frames of VA and VB.

Let $v_k^A(t)$ be the k^th feature point in the frame t of video VA, and vB k (t) be the corresponding feature point in video VB, the stitch term Estitch is defined to make every pair of features as close as possible in the panoramic video:

$$\text{Estitch (PA, PB, H)} = \sum \| Pi^A\ (t)\cdot Ci^{A\hat{}}\ (t)^{-1}\cdot v_A^K\ (t) - H\cdot\ Pj^B\ \ (t)\cdot Cj^B B\ (t) - 1\cdot\ v_K^B\ (t) \| \qquad \text{Eq. 4.7}$$

Where ˆ i, ˆ j are the grids in which the feature points v_k^A (t) and v_k^B (t) are located. Please be noted that, since the videos are stabilized, the positions of the features points are transformed accordingly, thus we measure difference between Pi^A (t) · Ci^A $(t)^{-1}\cdot v_k^A$ (t) and H·PB ˆ j (t)·CB ˆ j (t)−1·vB k (t), rather than between v_k^A (t) and H · v_k^B (t).

We explain Eq. 4.7 in a more pictorial way. The variable H functions as a translation that moves video VB in the spatial space to make VB roughly registered with VA. However, with H only, there are two problems left. First, the input videos are still very shaky. Second, a single homography cannot align all the spatial feature matches, yielding bad stitching results. We thus introduce the stabilization variables PA and PB to solve the two problems. They slightly warp input videos in a spatial-variant way frame by frame, like as-projective-as-possible image warping in the method of yielding smoothed camera paths and aligned feature matches. In summary, the stitching variable makes input videos coarsely aligned, while the stabilization variables compensate it by much finer adjustments.

a) Solver: Since the stitch term Eq. 4.7 correlates elements of matrices of PB and H, Eq. 4.6 is a nonlinear problem. It is slow to solve Eq. 4.3 directly by traditional gradient decent methods. Instead, we propose an alternating optimization strategy to approximate it, and make every step of the strategy being a linear least square problem. Specifically, we compute PA, PB, and H iteratively. Firstly, we solve Eq. 4.3 alone to stabilize VA and VB separately, and obtain initial PA and PB. The solution of Eq.4.3 can be found. With known PA and PB, we solve Eq. 4.5 alone to obtain H. The stitch term actually measures relations between two sets of points by H which thus can be easily computed by Direct Linear Transformation.

Then with known PB and H, we can optimize the following function to obtain a new PA:

$E(PA) = Estable\ (PA) + \beta \cdot Estitch\ (PA).$ Eq.4.8

where Estitch (PA) is identical to Estitch (PA, PB, H) of Eq. 4.5 in form, except that PB and H are known. Although Eq. 4.6 is very simple, we further simplify it by two steps. Firstly, with known PB and H, we optimize Eq. 4.5 to obtain a temporary camera path PA∗ for video VA. Then, we re-define the stitch term as:

$Estitch*\ (PA) = \| Pi^{A}\ (t) - Pi^{A}\ (t) \|^{2}.$ Eq.4.9

Finally, we optimize the following function to obtain a new PA:

$E(PA) = Estable\ (PA) + \beta \cdot Estitch * (PA).$ Eq.4.10

With known PA and H, we can obtain a new PB in the same way. Then with new PA and PB, the iteration can be continued to compute a new H. To stop the iteration, we compute average stitching error of all feature matches. The iteration is going on until being executed more than 10 times or the average stitching error is less than 2 pixels.

4.4 Summary

This chapter presents detailed design for each module; structural chart of the implemented scheme is presented in section 4.1. Section 4.2 gives information on detailed design of background identification module, unified video stitching module, false feature elimination module with flowchart for each module.

Chapter 5

IMPLEMENTATION

Implementation is a stage where the project is made to work and demonstrated. In implementation step, the system handles the real operations used for the working of the system. Each consequence completed by the implemented system will be enormous, if processes are properly executed according to the planned flow of the system.

5.1 Implementation Requirements

To implement the image compression using pixel decimation process, software used are-

- Language used for coding the implemented system is MATLAB 16.0
- Operating system-Windows XP or higher

5.2 Programming Language Used

Programming language which has been used to design and implement the proposed system is MATLAB. The abbreviation of MATLAB is given as Matrix laboratory which is 4th generation language for high level programming. MathWorks has designed and developed MATLAB. MATLAB is easier to learn as the coding in MATLAB is quite simpler. Easy Environment is provided by MATLAB to furnish computing and visualization of programming. Previously, it was used by researchers and scientists. But in recent years, it is mostly used by many technical people and researchers to implement their work. It also supports languages like C/C++, Python, Java. MATLAB is most compatible for matrix operation. It has many built-in functions and commands which allows the user to execute mathematical calculations and it also allows plotting graphs. MATLAB tool is mostly used in Image Processing field, communication process and Signal Processing. It contains many inbuilt functions that are useful for the beginners to understand and learn easily.

5.2.1 Characteristics of MATLAB

This language has several numbers of characteristics with high-level programming facility. The characteristics are as follows:

- MATLAB speaks maths
- It is designed for engineers and scientists
- MATLAB toolboxes just work
- It has APPs
- It integrates workflows
- It has been designed to be fast when performing matrix operations
- It is trustworthy
- Scalability

5.2.2 Key Features of MATLAB

MATLAB helps the user to take the ideas beyond the desktop. On the larger data sets, analyses can be run and, on the clusters, and clouds, scale up can be done. Deployment of algorithms and applications can be done by the user. Few of the key features of MATLAB are as follows:

- Higher level language is designed. for scientific computing and engineering computing
- Iterative exploration, design and problem-solving works well in MATLAB as desktop environment is tuned to do so.
- To create custom plots, graphics has been designed so as to visualize data and tools

- Apps are developed to perform tasks like curve fitting, data classification, signal0analysis and many other domain0specific tasks.
- Toolboxes are designed for a vast range of applications in engineering fields and scientific fields
- Tools are designed for application construction with the user interfaces which are accustomed.
- Predefined interfaces are specified to languages and software like C,aC++, .NET, Hadoop, Python, SQL and MicrosoftExcel.
- MATLAB program sharing with end users is one of the royalty-free deployment options.

5.3 Graphical Representation

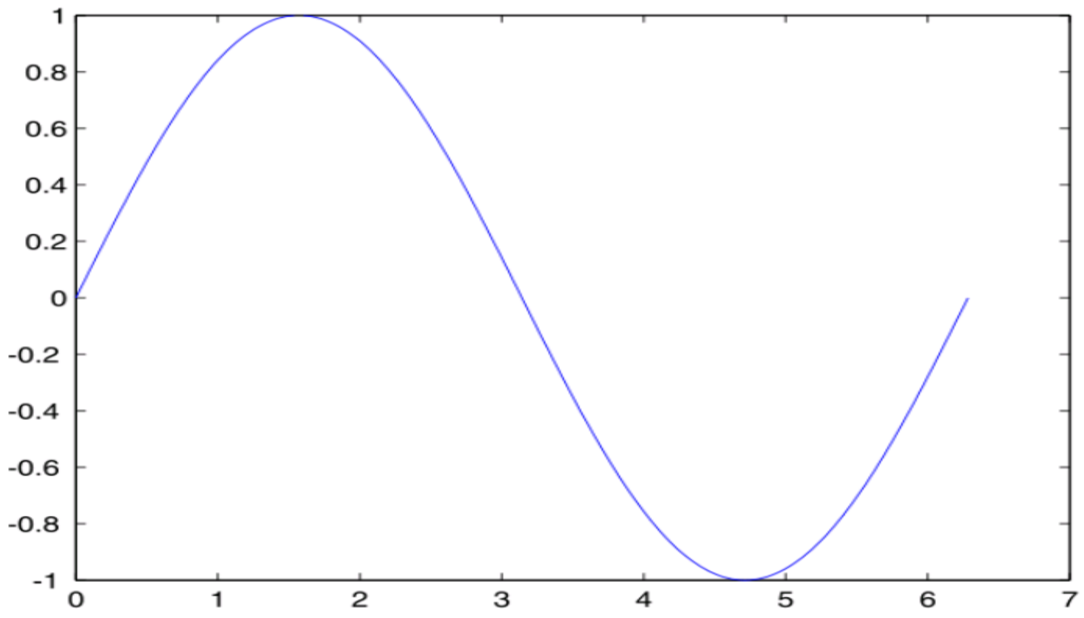


Figure 5.1: Graphical Representation of Sine Function

In MATLAB, graphical user interfaces can be programmed with the GUI design environment (GUIDE) tool.

Object-oriented programming

MATLAB's support for object-oriented programming includes classes, inheritance, virtual dispatch, packages, pass-by-value semantics, and pass-by-reference semantics.

```
Class def hello
   methods
      function doit(this)
         disp('Hello!')
      end
   end
end
```

When put into a file named hello. m, this can be executed with the following commands:

```
>> x = hello;
>> x.doit;
Hello!
```

Interfacing with other languages

MATLAB can call functions and subroutines written in the C programming language or Fortran. A wrapper function is created allowing MATLAB data types to be passed and returned. The dynamically loadable object files created by compiling such functions are termed "MEX-files" (for **M**ATLAB **ex**ecutable).

Libraries written in Java, ActiveX or .NET can be directly called from MATLAB and many MATLAB libraries (for example XML or SQL support) are implemented as wrappers around Java or ActiveX libraries. Calling MATLAB from Java is more complicated, but can be done with a MATLAB extension, which is sold separately by MathWorks, or using an undocumented mechanism called JMI (Java-to-MATLAB Interface), which should not be confused with the unrelated Java Metadata Interface that is also called JMI.

As alternatives to the MuPAD based Symbolic Math Toolbox available from MathWorks, MATLAB can be connected to Maple or Mathematica.

Libraries also exist to import and export MathML.

Pseudo code for main module:

```
fID = fopen('f.txt','w');
fclose(fID);
clear
close all
vid1=VideoReader('1.mp4'); // Video is read
% numFrames = vid1.NumberOfFrames; // Detect how many no of frames generated.
 n=10;
 for i = 1:1: n
 frames = read (vid1, i);
 inwrite (frames, ['vid' int2str(i), '.jpg']);
 im(i)=image(frames); // Displaying
 end
fID = fopen('f.txt','a+');
fprintf(fID,'%d',n);
fclose(fID);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
```

```
close all

 n=load('f.txt');

 vid2=VideoReader('2.mp4'); // Video is read.

 for i = 1:1: n

 frames = read (vid2, i); // Detect how many no of frames generated.

 imwrite (frames, ['mov' int2str(i), '.jpg']);

 im(i)=image(frames); // Displaying

 end %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mkdir ('0'); // Creates a file location.

movefile ('*.jpg',’. /0/'); // Files are moved to folder ‘0’.
```

The above code takes two input videos and converts into into frames , the video are read using video reader and the frames are generated by the object which detects how many number of frames are generated from input videos and then are written and displayed and hence the same process is carried for the second input video.

Pseudocode for stitching:

Video stitching has received much less attention than image stitching. Different approaches have been proposed for different camera settings. For example, some works, were designed for static surveillance cameras, while some for moving cameras but the relative geometries between the cameras are still fixed. To stitch videos captured by these cameras, the pose relationships between the cameras can be pre-calibrated to stitch every pair of frames globally, followed by local adjustments such as structure deformation or spatiotemporal consistent warping to eliminate small deviations.

```
function [ imgout ] = imStitch( img1,img2,adjColor ) // User defined function.

[matchLoc1 matchLoc2] = siftMatch(img1, img2); // Match points of image1 and image2.
```

```
[H corrPtIdx] = findHomography(matchLoc2',matchLoc1'); // Creates a spatial transform structure using H values.

H // To display in the command prompt.

tform = maketform('projective',H'); // Creates H values.

img21 = imtransform(img2,tform); // reproject img2

[M1 N1 dim] = size(img1);

[M2 N2 dim] = size(img2);

% do the mosaic

Yoffset = 0; // Y offset of H values.

if up <= 0

        Yoffset = -up+1;                    // To adjust H values of Y positions.

        up = 1;

end

left = round(min(x2));

Xoffset = 0; // X offset of H values.

if left<=0

        Xoffset = -left+1; To adjust the H values of X positions.

        left = 1;

end

[M3 N3 dim] = size(img21);

imgout(up:up+M3-1,left:left+N3-1,:) = img21;
```

```
    % img1 is above img21
imgout(Yoffset+1:Yoffset+M1,Xoffset+1:Xoffset+N1,:) = img1;
end
```

Pseudocode for SIFT Match:

The SIFT match takes the converted frames and their locations of those matching points (H values) and then identifies the descriptor locations of the two images taken and finds the distance ratio among them and precomputes the transpose matrix and vector of dot products and take inverse cosine and sorts the result.

```
function [matchLoc1 matchLoc2] = siftMatch (img1, img2) // User defined function
[des1, loc1] = sift(img1);
[des2, loc2] = sift(img2);
distRatio = 0.6;
des2t = des2’; // Precompute matrix transpose
matchTable = zeros (1, size(des1,1));
for i = 1: size(des1,1)
  dotprods = des1(i, :) * des2t; // Computes vector of dot products
  [vals,indx] = sort(acos(dotprods)); // Take inverse cosine and sort results
if (vals(1) < distRatio * vals(2))
    matchTable(i) = indx (1);
  else
    matchTable(i) = 0;
  end
```

end

5.4 Summary

This chapter gives description about the implementation of the dynamic video stitching using unified video stitching and stabilization optimization. Implementation requirements are shown in section 5.1. Section 5.2 details about the programming languages used. Next section 5.3 presents the pseudo code of main module and sub modules.

Chapter 6

SYSTEM TESTING

6.1 Testing

System testing is an important step where testing of a software product which is completely integrated is carried out. Software is one of the components of a large computer system; later it is made to interface with other software and hardwre systems. System testing is said to be a sequence of various tests where the purpose is to get the complete system.

System testing includes testing the code of the software for the following:

- Testing the integrated0applications which is called end-to-end testing scenario.
- Verify testing thoroughly for each input in the application so as to get desired output.
- Testing based on the user’s experience.

6.2 Unit Test Cases

Unit testing is testing the program by breaking the program into pieces and then each individual piece is subjected to series of tests. Unit test is designed to be performed in a simple manner, usually the tests are given in the form of functions and checked whether the output obtained is as expected. Test cases written for each unit are the unit test cases. These are individually tested to check whether the output obtained is as expected by the programmer. The unit test cases performed for this implemented system are as follows.

6.2.1 Video Stitching module testing

Table 6.1 shows the unit test cases for low-pass prefiltering step in the video stitching module. Once the image has been read, the image has to be prefiltered using low-pass prefilter. The output must be a mosaic image.

Table 6.1 Video Stitching module testing

Test case Sl no	1
Test Name	Video stitching Module
Test feature	Stitches the mosaic images
Output Expected	Smoothened image
Output Obtained	Stitched mosaic image converted into video as shown in snapshot no 7.7.
Result	Successful

6.2.2 Image Stabilization module testing

Table 6.2 shows the test case for the image stabilization which reduces the shakiness of the frames which are generated from the input videos and hence the stabilized images will be produced.

Table 6.2 Image Stabilization module testing

Test case Sl no	2
Test Name	Image Stabilization
Test feature	Reduces the shakiness of the input video frames.
Output Expected	Frames with no shakiness.
Output Obtained	Frames with very less shakiness as shown in snapshot no 7.6.
Result	Successful

6.2.3 Discrete-Cosine Transform Module Testing

Jpeg image stitching module comprises of discrete-cosine-transform. Table 6.3 describes the discrete-cosine-transform steps.

Table 6.3 SIFT match encoding

Test case Sl no	4
Test Name	SIFT match
Test feature	Identify the descriptors and locations.

Output Expected	Provides exact descriptors and locations.
Output Obtained	Required descriptors and locations of selected frames as shown in snapshot no 7.7.
Result	Successful

6.2.4 Image Stitching Module Testing

Table 6.5 shows the image stitching module testing where the videos taken as the input are converted into frames and this helps to stitch those two frames based on the matching points provided by the SIFT and produces a corresponding matrix of those images.

Table 6.4 Image stitching testing

Test case Sl no	5
Test Name	Stitching
Test feature	Stitches the given two frames.
Output Expected	Creates a new matrix used in creation of mosaic image.
Output Obtained	Matrix of the mosaic image as shown in snapshot no 7.8.
Result	Successful

Image stitching module stitches the image which was given as input. Table 6.5 shows the image stitch testing.

6.2.5 False Match Elimination Module Testing

False Match Elimination module testing gives the required match points of the images and removes the unwanted match points which are not necessary. Table 6.6 shows the false match elimination step testing.

Table 6.5 Image Upscaling testing

Test case Sl no	6
Test Name	False Match Elimination
Test feature	Eliminates the false match points.
Output Expected	Match points of the input frames generated.
Output Obtained	Required match points for SIFT as shown in snapshot no 7.7.
Result	Successful

6.3 Summary

This chapter presents unit test cases and briefs about the testing for each module. Section 6.1 presents few points about testing. Section 6.2 describes the unit test cases for each module.

Chapter 7

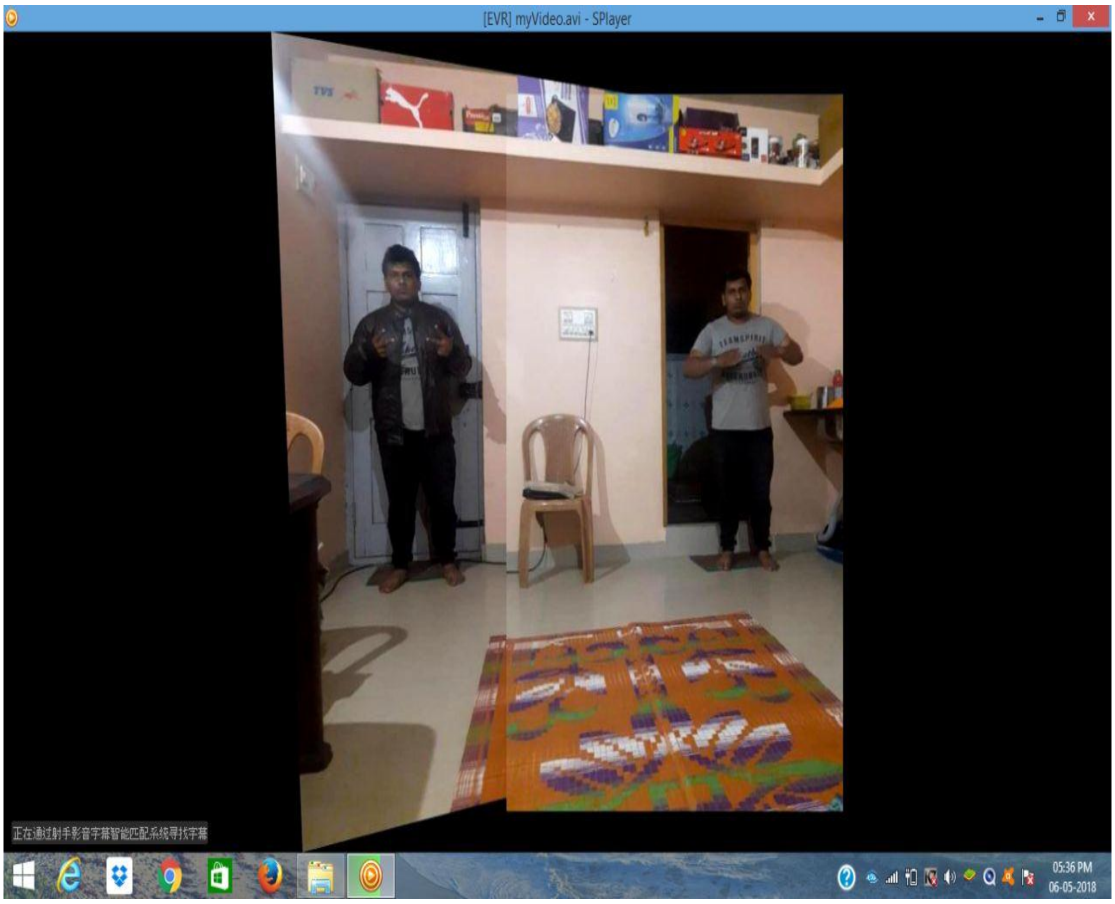
RESULTS AND DISCUSSIONS

7.1 Snapshots

The snapshot gives an overall idea of the working of the system. The snapshots presented here are the shots taken when the implemented system is actively executing.

7.1.1 The Output of the Dynamic Video Stitching via Shakiness Removal

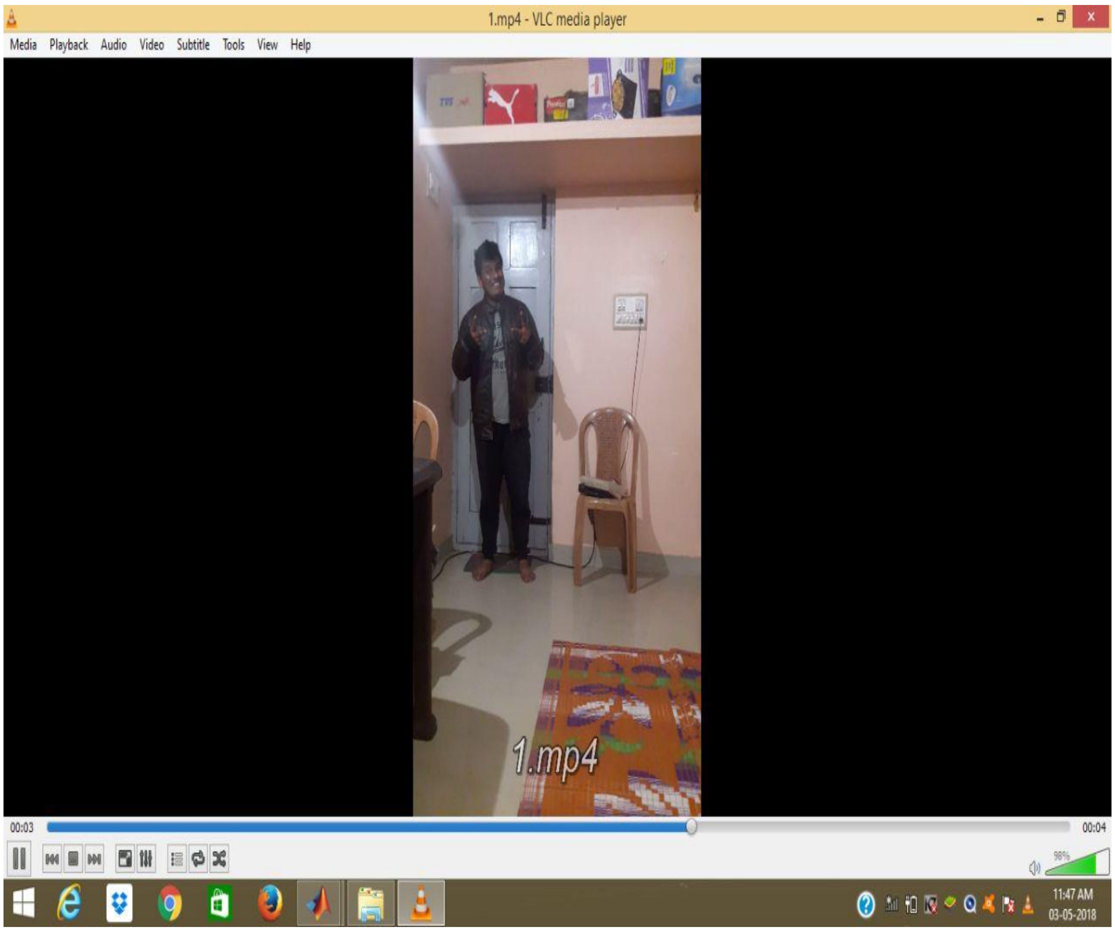
The snapshot 7.1 shows the output of Video Stitching. The output video will be in avi format. The output video is played in the desktop video player.


[EVR] myVideo.avi - SPlayer
正在通过射手影音字幕智能匹配系统寻找字幕
05:36 PM
06-05-2018

Snapshot 7.1: Output Video of dynamic video Stitching Via shakiness Removal

7.1.2 Input Video1 Selected for Video Stitching

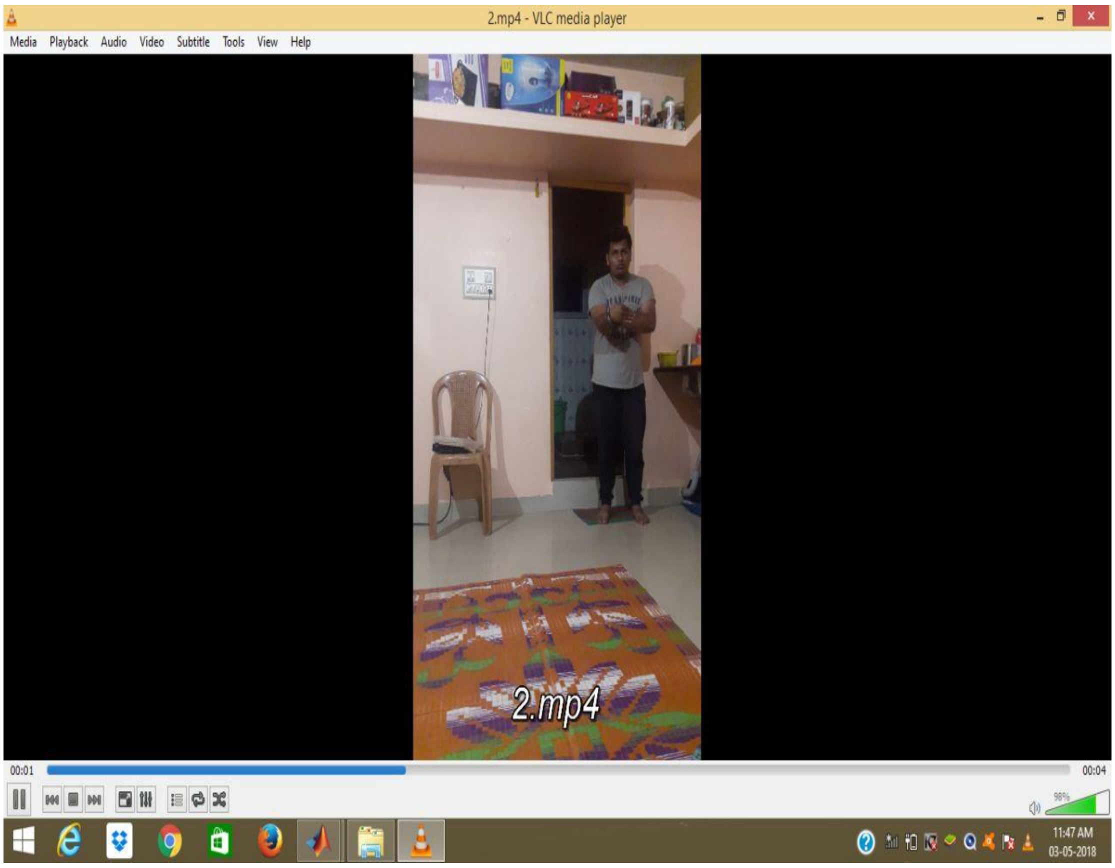
The snapshot 7.2 shows the image compression page with the input video 1 ready to be stitched. The video 1 is selected for stitching. Input video that has to be stitched is stored in particular file on the desktop. Next encoding has to be done, where the input video has to be stitched.



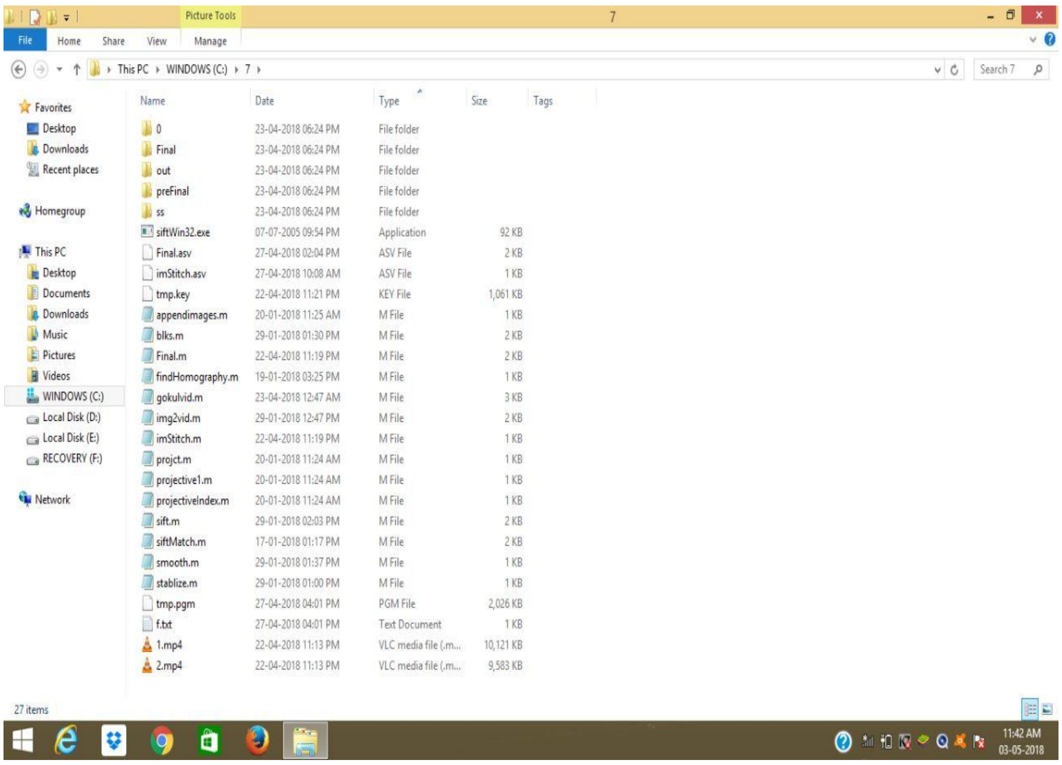
Snapshot 7.2: Input image 1 selected for stitching

7.1.3 Input Video 2 Selected for Video Stitching

The snapshot 7.3 shows the input video 2 selected for stitching videos. Input video 2 is stored in the same folder where video 1 is stored in particular file on the desktop. Next encoding has to be done, where the given input videos has to be stitched.



Snapshot 7.3: Input image 2 selected for stitching

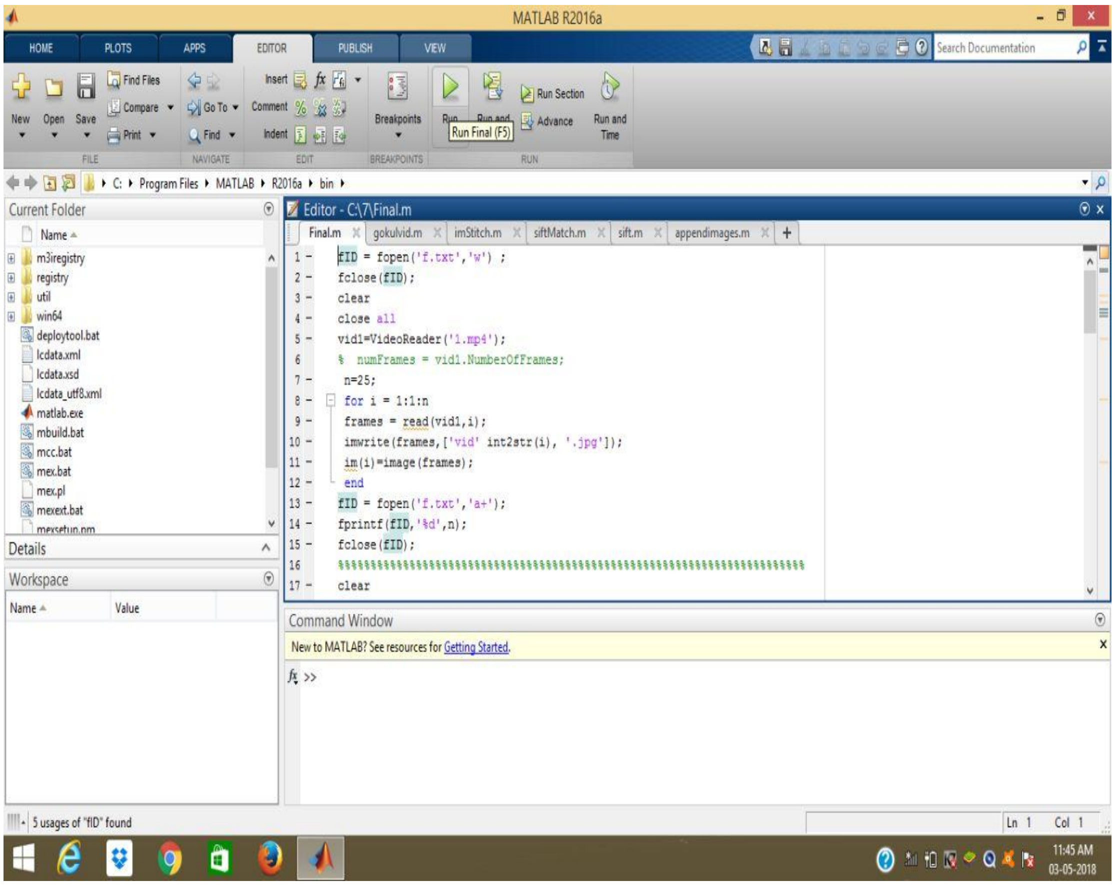


Name	Date	Type	Size	Tags
0	23-04-2018 06:24 PM	File folder		
Final	23-04-2018 06:24 PM	File folder		
out	23-04-2018 06:24 PM	File folder		
preFinal	23-04-2018 06:24 PM	File folder		
ss	23-04-2018 06:24 PM	File folder		
siftWin32.exe	07-07-2005 09:54 PM	Application	92 KB	
Final.asv	27-04-2018 02:04 PM	ASV File	2 KB	
imStitch.asv	27-04-2018 10:08 AM	ASV File	1 KB	
tmp.key	22-04-2018 11:21 PM	KEY File	1,061 KB	
appendimages.m	20-01-2018 11:25 AM	M File	1 KB	
blks.m	29-01-2018 01:30 PM	M File	2 KB	
Final.m	22-04-2018 11:19 PM	M File	2 KB	
findHomography.m	19-01-2018 03:25 PM	M File	1 KB	
gokulvid.m	23-04-2018 12:47 AM	M File	3 KB	
img2vid.m	29-01-2018 12:47 PM	M File	2 KB	
imStitch.m	22-04-2018 11:19 PM	M File	1 KB	
projct.m	20-01-2018 11:24 AM	M File	1 KB	
projective1.m	20-01-2018 11:24 AM	M File	1 KB	
projectiveIndex.m	20-01-2018 11:24 AM	M File	1 KB	
sift.m	29-01-2018 02:03 PM	M File	2 KB	
siftMatch.m	17-01-2018 01:17 PM	M File	2 KB	
smooth.m	29-01-2018 01:37 PM	M File	1 KB	
stablize.m	29-01-2018 01:00 PM	M File	1 KB	
tmp.pgm	27-04-2018 04:01 PM	PGM File	2,026 KB	
f.txt	27-04-2018 04:01 PM	Text Document	1 KB	
1.mp4	22-04-2018 11:13 PM	VLC media file (.m...	10,121 KB	
2.mp4	22-04-2018 11:13 PM	VLC media file (.m...	9,583 KB	

Snapshot 7.4: Input Video location

7.1.4 Input Videos Stitching Process

The snapshot 7.5 shows the MATLAB code for stitching. The method takes given two input videos for stitching. During stitching, the mosaic images for both the videos are generated separately. It is necessary to give the number of frames to be obtained from given input videos. In the implemented system, the frames obtained will be stored in desktop folder.



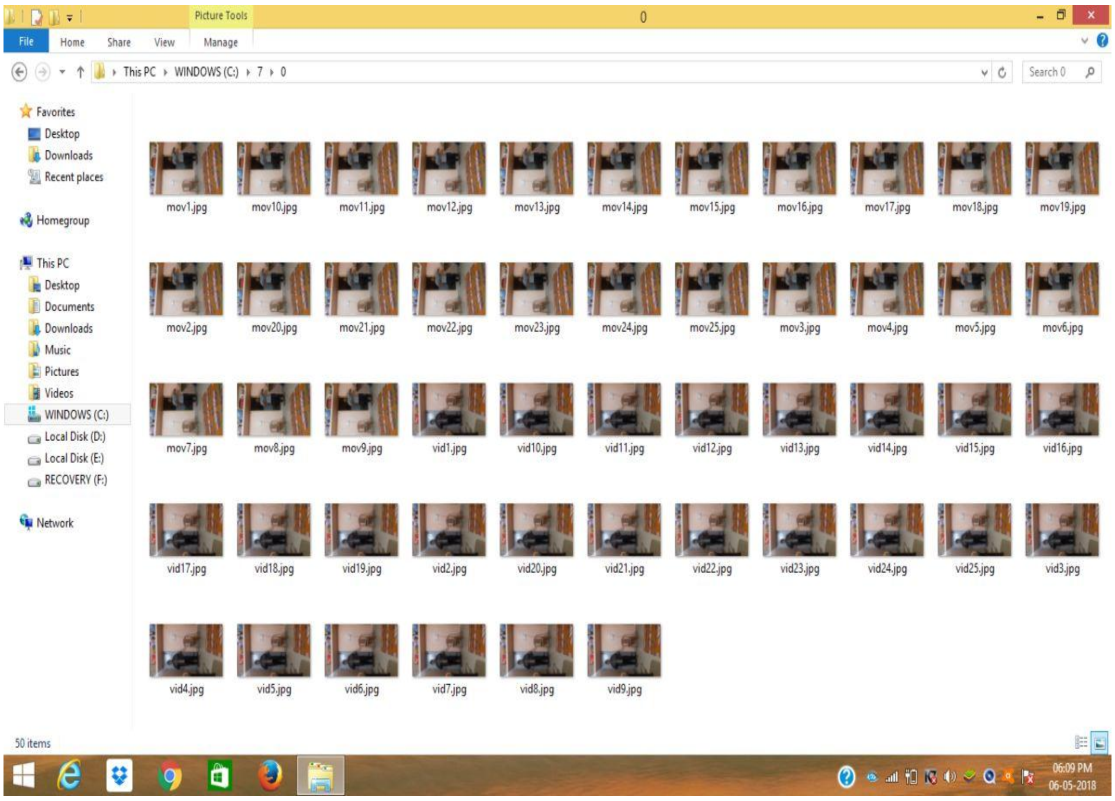
```
fID = fopen('f.txt','w') ;
fclose(fID);
clear
close all
vid1=VideoReader('1.mp4');
%  numFrames = vid1.NumberOfFrames;
 n=25;
 for i = 1:1:n
 frames = read(vid1,i);
 imwrite(frames,['vid' int2str(i), '.jpg']);
 im(i)=image(frames);
 end
fID = fopen('f.txt','a+');
fprintf(fID,'%d',n);
fclose(fID);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
```

```
>>
```

Snapshot 7.5: Code to Run MATLab

7.1.5 Generation of Frames from Input videos for Stitching

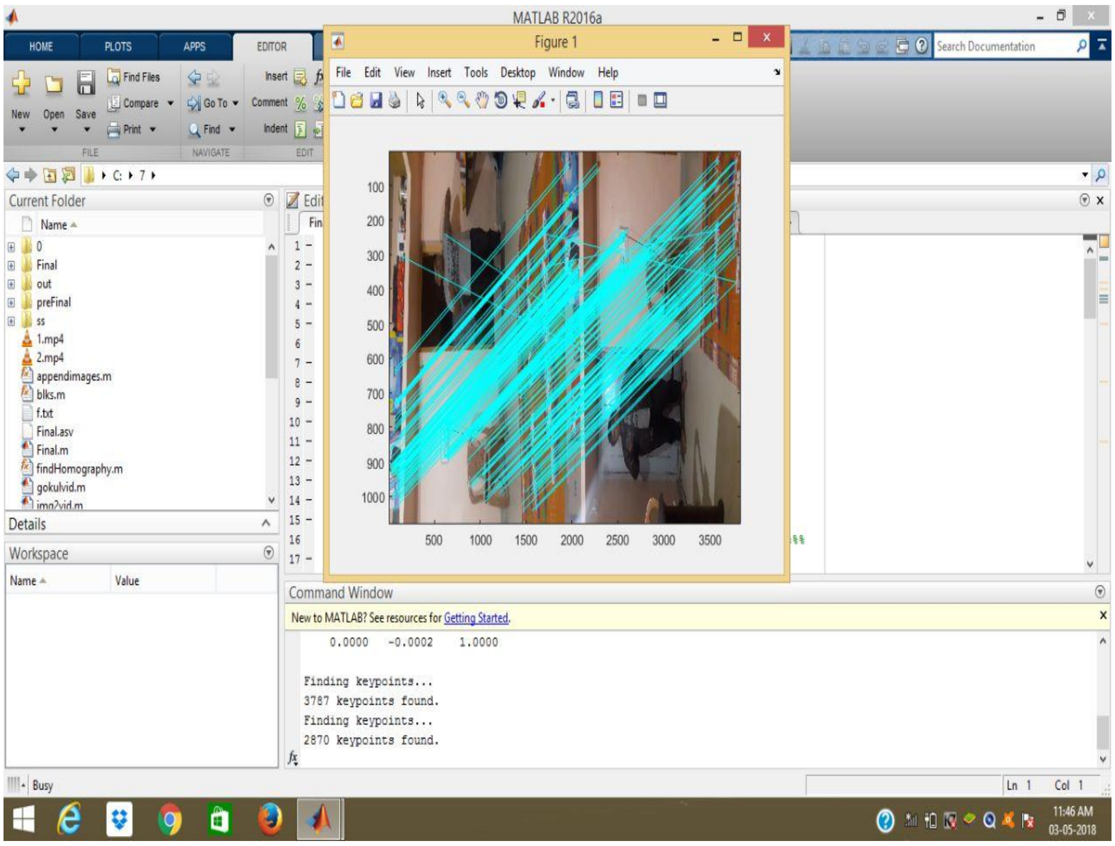
The snapshot 7.6 shows the frames generate during stitching of two videos. It takes generates the frames from the input video1 and video 2. As a result of stitching mosaic images is generated. The number of mosaic images generated depends on the number of matching frames stitched from video 1 and video 2.



Snapshot 7.6: Frames generated from video 1 and video 2

7.1.6 Matching points generation for Stitching

The snapshot 7.7 shows the matching points generated during stitching of two videos. It takes generates the two matching frames from the input video1 and video 2. As a result of stitching mosaic images is generated. The number of mosaic images generated depends on the number of matching frames stitched from video 1 and video 2.



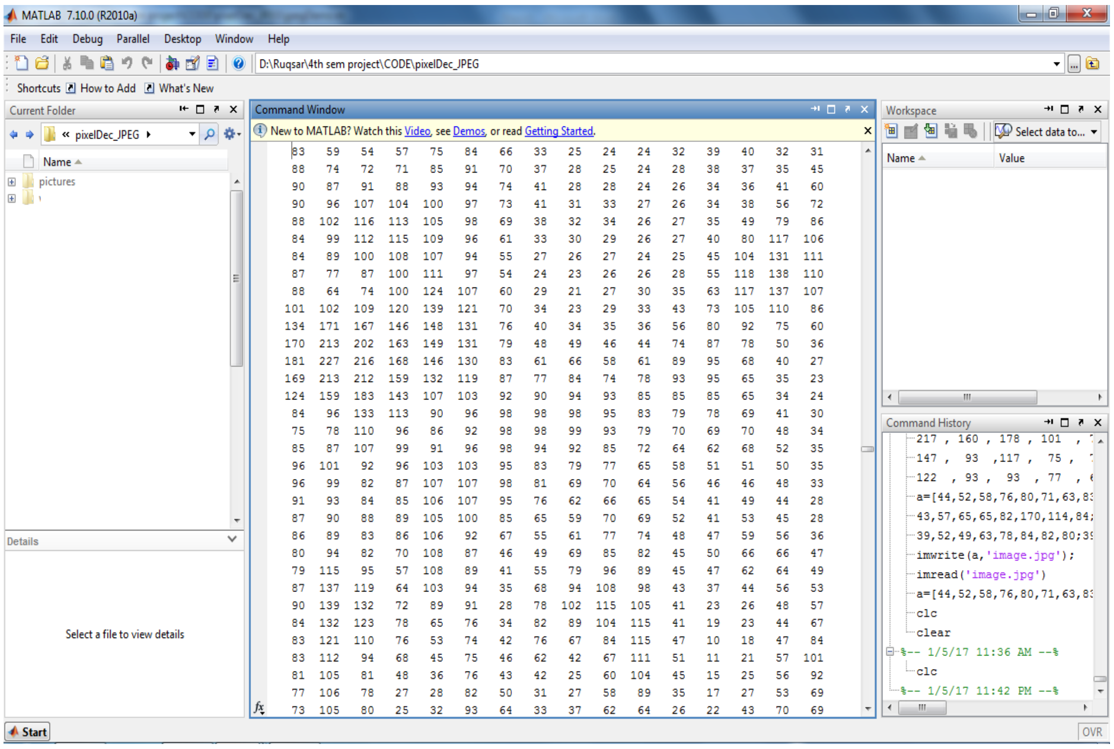
```
     0.0000   -0.0002    1.0000

Finding keypoints...
3787 keypoints found.
Finding keypoints...
2870 keypoints found.
```

Snapshot 7.7: Matching points from video 1 and video 2

7.1.7 Matrix H Values to Stitch

The snapshot 7.8 shows the matrix H values of the mosaic image selected. From the matrix H, values obtained will undergo through stitching process and is converted into video format


MATLAB 7.10.0 (R2010a)
File Edit Debug Parallel Desktop Window Help
D:\Ruqsar\4th sem project\CODE\pixelDec_JPEG
Shortcuts How to Add What's New
Current Folder
pixelDec_JPEG
Name
pictures
Details
Select a file to view details
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Workspace
Select data to...
Name
Value
Command History
Start
OVR

```
 83   59   54   57   75   84   66   33   25   24   24   32   39   40   32   31
 88   74   72   71   85   91   70   37   28   25   24   28   38   37   35   45
 90   87   91   88   93   94   74   41   28   28   24   26   34   36   41   60
 90   96  107  104  100   97   73   41   31   33   27   26   34   38   56   72
 88  102  116  113  105   98   69   38   32   34   26   27   35   49   79   86
 84   99  112  115  109   96   61   33   30   29   26   27   40   80  117  106
 84   89  100  108  107   94   55   27   26   27   24   25   45  104  131  111
 87   77   87  100  111   97   54   24   23   26   26   28   55  118  138  110
 88   64   74  100  124  107   60   29   21   27   30   35   63  117  137  107
101  102  109  120  139  121   70   34   23   29   33   43   73  105  110   86
134  171  167  146  148  131   76   40   34   35   36   56   80   92   75   60
170  213  202  163  149  131   79   48   49   46   44   74   87   78   50   36
181  227  216  168  146  130   83   61   66   58   61   89   95   68   40   27
169  213  212  159  132  119   87   77   84   74   78   93   95   65   35   23
124  159  183  143  107  103   92   90   94   93   85   85   85   65   34   24
 84   96  133  113   90   96   98   98   98   95   83   79   78   69   41   30
 75   78  110   96   86   92   98   98   99   93   79   70   69   70   48   34
 85   87  107   99   91   96   98   94   92   85   72   64   62   68   52   35
 96  101   92   96  103  103   95   83   79   77   65   58   51   51   50   35
 96   99   82   87  107  107   98   81   69   70   64   56   46   46   48   33
 91   93   84   85  106  107   95   76   62   66   65   54   41   49   44   28
 87   90   88   89  105  100   85   65   59   70   69   52   41   53   45   28
 86   89   83   86  106   92   67   55   61   77   74   48   47   59   56   36
 80   94   82   70  108   87   46   49   69   85   82   45   50   66   66   47
 79  115   95   57  108   89   41   55   79   96   89   45   47   62   64   49
 87  137  119   64  103   94   35   68   94  108   98   43   37   44   56   53
 90  139  132   72   89   91   28   78  102  115  105   41   23   26   48   57
 84  132  123   78   65   76   34   82   89  104  115   41   19   23   44   67
 83  121  110   76   53   74   42   76   67   84  115   47   10   18   47   84
 83  112   94   68   45   75   46   62   42   67  111   51   11   21   57  101
 81  105   81   48   36   76   43   42   25   60  104   45   15   25   56   92
 77  106   78   27   28   82   50   31   27   58   89   35   17   27   53   69
 73  105   80   25   32   93   64   33   37   62   64   26   22   43   70   69
```

```
217 , 160 , 178 , 101  ,
147 ,  93  ,117 ,  75 ,
122  , 93 ,  93  , 77  ,
a=[44,52,58,76,80,71,63,83
43,57,65,65,82,170,114,84;
39,52,49,63,78,84,82,80;39
imwrite(a,'image.jpg');
imread('image.jpg')
a=[44,52,58,76,80,71,63,83
clc
clear
%-- 1/5/17 11:36 AM --%
clc
%-- 1/5/17 11:42 PM --%
```

Snapshot 7.8: Matrix H values generated

7.3 Summary

This chapter gives description about snapshots in the implemented system and the performance analysis which gives the PSNR graph and the image size graph. Section 7.1 gives the snapshots present in the system and the section 7.2 gives the performance analysis.

Chapter 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 Conclusion

We have introduced a novel method that can stitch hand taken shaky videos effectively. Our method executes video stitching and stabilization simultaneously. The computed stabilization is the best for stitching, based on which we produce better stitching results compared to previous methods. We also propose two auxiliary components, i.e., background identification component, and false feature elimination loop scheme to further improve the effectiveness and robustness of our method.

8.2 Future Enhancement

Various stitching and stabilization methods can be applied for stitching results. To achieve stability, stabilization can be optimized iteratively until the best optimization is obtained. Other interpolation techniques can be used for image upscaling. We will work on the performance efficiency of our current system for mobile applications in the further research.

REFERENCES

[1] M. Zheng, X. Chen, and L. Guo, “Stitching video from webcams,” in International Symposium on Visual Computing. Springer, 2008, pp.420–429.

[2] F. Perazzi, A. Sorkine-Hornung, H. Zimmer, P. Kaufmann, O. Wang,S. Watson, and M. Gross, “Panoramic video from unstructured camera arrays,” in Computer Graphics Forum, vol. 34, no. 2. Wiley Online Library, 2015, pp. 57–68.

[3]H. Guo, S. Liu, T. He, S. Zhu, B. Zeng, and M. Gabbouj, “Joint video stitching and stabilization from moving cameras,” IEEE Transactions on Image Processing, vol. 25, no. 11, p. 5491, 2016

[4] S. Liu, L. Yuan, P. Tan, and J. Sun, “Bundled camera paths for video stabilization,” ACM Transactions on Graphics (TOG), vol. 32, no. 4, p. 78, 2013

[5] K. Lin, S. Liu, L.-F. Cheong, and B. Zeng, “Seamless video stitching from hand-held camera inputs,” in Computer Graphics Forum, vol. 35, no. 2. Wiley Online Library, 2016, pp. 479–487.

[5] Rekhil M Kumar, “A Survey on Image Feature Descriptors”, Dept. Of Computer and Information Science College of Engineering, Poonjar, vol 7, PP-109-112, April 2012.

[6] Faraj Alhwarin, C. W, “Improved SIFT-Features Matching for Object Recognition”, In E. Gelenbe, S. Abramsky, and V. Sassone (Ed.), BCS Int. Acad. Conf. (pp. 178-190), March 2015.

[7] Mrs. Hetal M. Patel, Asst. Prof. Pinal. J. Patel, Asst. Prof. Mr. Sandip G Patel, “Comprehensive Study and Review of Image Mosaicing Methods”, IJERT (International journal of Engineering and Research), vol 7, PP-143-147, July 2014.

[8] Yanfeng Li, Y. W, “Automatic Image Stitching Using SIFT”, IJERT (International journal of Engineering and Research), vol 9, PP-163-167, Jan 2011.

[9] G. Yu and J.-M. Morel, “ASIFT: An Algorithm for Fully Affine Invariant Comparison, Image Processing on Line”, IJIRSET (International Journal of Innovative Research in Science, Engineering and Technology) vol. 1, PP-156-159, Dec 2011.

LIST OF PUBLICATIONS

National Conference

[1] Gokul K.B, Muskaan, Amulya, Ashik MS "Dynamic Video Stitching Via Shakiness Removal", 3rd National Conference on Emerging Trends and Advances in Information Technology, A.I.T, Chikkamagaluru, April 2018.