RV COLLEGE OF ENGINEERING®
BENGALURU - 560059
(Autonomous Institution Affiliated to VTU, Belagavi)

CCTV Video Summarisation in Urban Environments

PROJECT REPORT

Submitted by

AS Karthik 1RV15CS001
Abhishek Krishna 1RV15CS007
Gagan Deep G 1RV15CS053

Under the Guidance of
Dr. Ramakanth Kumar P
Professor & HoD
Dept. of CSE, RVCE
Bengaluru - 560059

In the partial fulfillment for the award of degree of
Bachelor of Engineering
in
COMPUTER SCIENCE & ENGINEERING
2018-19

RV COLLEGE OF ENGINEERING®
BENGALURU - 560059

(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the major project work titled **CCTV Video Summarisation in Urban Environments** has been carried out by **AS Karthik (1RV15CS001), Abhishek Krishna (1RV15CS007) & Gagan Deep G (1RV15CS053)** who are bonafide students of RV College of Engineering, Bengaluru 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 2018 - 2019. It is certified that all corrections/-suggestions indicated for the internal assessment have been incorporated in the report deposited in the departmental library. The major project report has been approved as it satisfies the academic requirements in respect of project work prescribed by the institution for the said degree.

Dr. Ramakanth Kumar P
Prof. & Head of Department,
Department of CSE
RVCE, Bengaluru - 59

Dr. Subramanya KN
Principal,
RVCE, Bengaluru - 59

External Viva

Name of the Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

RV COLLEGE OF ENGINEERING®
BENGALURU - 560059
(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, **AS Karthik (1RV15CS001), Abhishek Krishna (1RV15CS007) & Gagan Deep G (1RV15CS053)**, students of Eighth Semester B.E., Computer Science and Engineering, RV College of Engineering, Bengaluru hereby declare that the major project titled **CCTV Video Summarisation in Urban Environments** has been carried out by us and submitted in partial fulfillment for the award of degree of Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belagavi during the academic year 2018-2019. We declare that matter embodied in this report has not been submitted to any other university or institution for the award of any other degree or diploma.

Place : Bengaluru **Signature**

1. _____

2. _____

Date: _____ 3. _____

Acknowledgement

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this project work. We would like to take this opportunity to thank them all.

We deeply express my sincere gratitude to our guide **Dr. Ramakanth Kumar P**, Professor & Head of Department, Department of CSE, RVCE, Bengaluru, for his able guidance, regular source of encouragement and assistance throughout this project. We also thank him for his valuable suggestions and expert advice.

We would like to thank **Dr. Subramanya. K. N**, Principal, RVCE, Bengaluru, for his moral support towards completing our project work.

We thank our Parents, and all the Faculty members and non-teaching staff of Department of Computer Science & Engineering for their constant support and encouragement.

Last, but not the least, we would like to thank my peers and friends who provided us with valuable suggestions to improve our project.

Abstract

The use of CCTVs has been on the steady rise in in the last decade, being used for the security of public and private establishments alike. With increasing crime rates in cities, there is a need for a smarter method to survey the video being recorded, rather than manually going through hours of footage, saving both time and labour. We propose a method to generate a summary of the footage. The project also aims to simplify the task further, by generating video summaries based on user query, thus showing exactly only the desired type of events.

The process is carried out in two phases, real-time phase and query phase. Real-time phase works on the raw CCTV footage working to detect motion, separate the foreground and background, extract the ”tubes of interest” and store it. YOLO object detector is then run on the these tubes and the metadata is also stored. The query phase works on this intermediate data. The user selects the objects (ex: car, bike etc) required in the summary. The required events are then selected, before being rearranged by an optimization algorithm like simulated annealing. A time-lapsed background of the input footage is simultaneously generated along with these processes. Finally, the rearranged tubes are blended into the background using poisson blending to generate the summary video.

Both objective and subjective parameters have been used to evaluate the results. Objectively, the exhaustiveness (inclusion of all events), compression factor, and overlap factor have been considered. A compression factor of 4-5x was successfully achieved, while all significant events were selected to achieve excellent exhaustiveness. Most trials also successfully managed to have a very low overlap factor. Subjectively, semantic structuring and realistic appearance of the footage were considered. Interacting events were always shown together (thereby maintaining semantic structuring). The blending of the tubes could however be better leading to a more realistic video.

Contents

List of Tables

List of Figures

Glossary

CCTV	Closed Circuit Television.
DFD	Data Flow Diagrams.
MOG	Mixture of Gaussians.
OpenCV	Open Computer Vision.
RCNN	Recurrent Convolutional Neural Network.
SSD	Single Shot Detector.
YOLO	You Only Look Once.

Chapter 1

Introduction to CCTVVS

The number of Closed Circuit Television (CCTV) surveillance cameras is increasing everyday leading to a huge amount digital video information being captured and stored. Millions of CCTV cameras run 24 hours a day, sometimes even streaming the content over the internet for people to monitor. This data is however in a raw, unprocessed format. In most cases, video with little to no motion is being captured, which wastes lot of storage. The process of watching or analysing the footage is also time consuming and laborious.

This project proposes to build a CCTV video summariser to eliminate this redundancy in stored data and give the user a short summary consisting of important information that is relevant for the specific use case. The system proposes to incorporate a query based summary generation to generate more relevant summaries. Users can choose the required objects and events to generate a short yet precise summary with only relevant information, saving more time.

The process takes place over two phases, the real-time and query phase. The real-time phase reads the CCTV footage, identifies clips of interest, extracts the "flow-tubes", and stores them after tagging them with the respective object tags. A query phase would then involve the user selecting the required tags and objects of interest, which are chosen. The tubes are rearranged using simulated annealing algorithm, before being blended into a generated time-lapsed background using an optimal technique like poisson blending.

This method reduces the manual work of going through hours of footage looking for relevant events by automatically creating a summary. It can be mainly used for security purposes, by the police forces for detection of crimes and suspicious activities.

1.1 State of the art developments

In this section, existing methods for summarizing videos and related information that helps achieve this task are discussed.

In [1], Yael Pritch and Alex Rav-Acha propose a method to effectively generate a synopsis of an endless video stream that can also be used as an index into the main video. An online phase includes tube detection in spatio-temporal domain, insertion of these tubes into an object queue, and removal upon reaching a space limit. The response phase then constructs a time-lapse video of the changing background, selection and stitching of tubes into a coherent video. Min-cut algorithm along with background subtraction has been used for extracting moving objects. Activity, collision and temporal consistency costs have been used a parameters for optimal tube arrangement.

In [2], Shmuel Peleg and Yael Pritch, have presented a dynamic video synopsis technique where most of the activity in the video is condensed by simultaneously showing several actions, even when they originally occurred at different times.

In [3], (CRAM: Compact Representation of Action in Movies), Mikel Rodriguez generates a compact video representation of a long sequence, which while preserving the general dynamics of the video features only the essential components. From the given input video, optical flows are generated. These are then represented as vectors in Clifford Fourier domain. Dynamic regions of flow are then identified within the phase spectrum volume. The likelihood of activities of relevance are then computed by correlating it with spatio-temporal maximum average correlation height filters. The final summary is then generated by a temporal-shift optimization. Although this method could detect specific actions, it couldn't keep all the events in the final summary.

Sarit Ratovitch, Avishai Hendel and Shmuel Peleg, in their paper titled Clustered Synopsis of Surveillance Video [4], present a different approach to generating video summaries, based on clustering of similar activities. Objects with similar activities are easy to watch simultaneously, which also makes spotting of outliers easier. This method is also suitable for creation of ground truth data. This paper covered three main topics, the definition of distance between activities, clustering of similar activities and efficient presentation of video summaries using obtained clusters.

In [5], (A Shortest Path Representation for Video Summarization), a new approach for video summarization is presented to select multiple key frames within an isolated video shot where there is camera motion, causing significant scene change. This can be done by determining dominant motion between frame pairs whose similarities are represented using a directed graph. A* algorithm is used to detect the shortest path and designate key frames. The overall set of key frames depict the essential video content and camera motions.

[6] presents a very successful and highly used method for adaptive pixel-level background subtraction. Each pixel has probability density function separately. A pixel is considered to be part of the background if its new value if well described by its density function. This paper was an improvement on previous models which used Gaussian mixture models with efficient update equations.

In [7], an extremely fast object detection model, the You Only Look Once (YOLO) model is described. While prior object detectors used classifiers to detect, this paper proposes object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network is used to predict both bounding boxes and its class probability, making end-to-end optimization easy. Although YOLO makes more localization errors, it is less likely to predict false positives compared to other object detectors like Single Shot Detector (SSD), Recurrent Convolutional Neural Network (RCNN) and Faster RCNN.

In [8], a combination of camera motion, object motion and text detection are used to identify which scenes are important in the input, and hence should be maintained in the summary.

1.2 Motivation

The prime motivation was the fact that manual analysis of CCTV footage is time consuming and laborious. With the increase in the number of CCTV's being used, it is necessary to have a smart method to generate a summary of footage which highlights what is necessary and removes the rest. The idea of this system was to help the users save tons of time and manual labour.

While there are basic methods like fast forwarding, adaptive fast-forwarding [9], and motion detection, the aim is to improve the efficiency further by generating a shorter and more relevant summary. This project also helped us to explore deeper into the domain of computer vision, towards which we share a common interest.

1.3 Problem Statement

Manual analysis of CCTV footage is extremely time consuming and laborious. Thus there is a need for smart tools to generate useful insights and summaries.

The task at hand is to develop an efficient method to generate a summary from a given input CCTV footage. It must be able to preserve important content while eliminating unnecessary details, thus saving hours of manual work hours spent going through the entire video. In addition, there is a lack of a smart query-based system for summarization. The system must generate a summary based on user query.

1.4 Objective

The objectives of the system are set with the requirements of each module.

- To develop a video summarization technique, that will take efficiently handle overlapping of events in the video.
- To add the functionality of generating a summary based on user query.
- To build a suitable UI and integrate with the available back-end.

1.5 Scope

The main area of application of this project is for security purposes. It can be used for detection of crimes, detecting suspicious activities [10] etc. Hence it will be of use to the security forces and the police.

1.6 Methodology

The project has been split into 2 phases – the real-time phase and query phase.

Real-time phase includes all those processes that work on the input data from the CCTV before storing the intermediate result in the database. These include background masking, to separate the foreground and background, motion detection to capture frames of interest, tube extraction to store these frames of interest, object detection to gather metadata about the events in the video. This phase also includes generating a static background of the input video. The generated data is then stored in the database.

The next phase, the query phase, then takes in user query, selects the tubes based on the query, rearranges the tubes using Simulated Annealing algorithm, and then blends these rearranged tubes into the background using Poisson Blending to generate the video summary.

1.7 Organization of Report

This section gives the overall picture of the many chapters in this report.

- **Chapter 2** gives an overview of the project domain which describes the details of the software and techniques used to carried out the project.
- **Chapter 3** is on Software Requirement Specification which describes the assumptions and dependencies, user characteristics, functional requirements and constraints of the project.
- **Chapter 4** is High Level Design which elucidates the design phase in Software Development Life Cycle. This chapter talks about the design considerations like

architectural strategies, general constraints, development methods and. It explains the project System Architecture and Data Flow Diagrams.

- **Chapter 5** is Detailed Design which explicates the two project modules. The functionality of each module represented as a flowchart is explained in this section.
- **Chapter 6** is Implementation which describes the technology used in the system. This section also explains programming language, development environment, code conventions followed.
- **Chapter 7** is on Software Testing which elaborates the test environment and briefly explains the test cases which were tried out during unit, integration and system testing.
- **Chapter 8** is Experimental Results which mentions the results found by the experimental analysis on different data sets. It talks about the inference made from the results.
- **Chapter 9** is Conclusion conveying the summary, limitations and future enhancements of the project.

1.8 Summary

This chapter deals with the introduction to the topic. The existing systems for video summarization are discussed. Research work, study material and papers on accessory software is also discussed. It also discusses in detail about the motivation, problem statement and the objectives of the project.

Chapter 2

Overview of CCTVVS

The fundamental idea is to generate a short summary which include all the important events[11], [12], [13], [14] to reduce the amount of manual effort and labour.

After having gone through several important papers in this sector, the framework described in this section has been narrowed down to. It mainly has two phases: Real-Time Phase and Query Phase.

2.1 Real-time phase

The real-time phase reads the CCTV footage, identifies clips of interest and performs certain image processing algorithms on the footage of interest to extract "flow-tubes" and tags from clips and stores them in a database[15]. The phase is split into the following steps:

1. **Motion Detection**

   Detect motion in the footage and identify any clips with significant motion while disregarding artifacts due to changing environmental conditions and other insignificant disturbances. Mixture of Gaussians (MOG) is used in this step to see if there is any movement, and if there is significant foreground present in an image, decided by a static threshold, the clip is considered to have motion in it. MOG is specifically useful here due to its dynamic nature and adaptability to gradual changes in the environment very quickly and, additionally, availability of efficient implementations of this very effective algorithm.

2. **Background Masking**

   A foreground extractor like a Mixture of Gaussians [6], [16], [17] is used to extract the subjects of interests in the clips identified by motion detection. The same technique used in previous step is also employed here to generate foreground masks and thereby just extracting the foreground. Several techniques were experimented

on MOG is the most cost effective and accurate technique available for this use case.

In Gaussian Mixture Model (GMM) defined as 2.1, every pixel in the frame is modelled into a Gaussian distribution. Probability of every pixel being the foreground or background is calculated as:

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t}.\eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \tag{2.1}$$

- X_t: current pixel in frame t
- K: the number of distributions in the mixture
- $\omega_{i,t}$: the weight of the k^{th} distribution in frame t
- $\mu_{i,t}$: the mean of the k^{th} distribution in frame t
- $\Sigma_{i,t}$: the standard deviation of the k^{th} distribution in frame t

Where $\eta(X_t, \mu_{i,t}, \Sigma_{i,t})$ is a probability density function defined in 2.2 as:

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}\Sigma^{1/2}} exp^{\frac{-1}{2}(X_t-\mu)\Sigma^{-1}(X_t-\mu)} \tag{2.2}$$

3. **Computation of Objects flow-tubes**

   Flow-tubes are computed from the extracted foreground in previous phase. Flow tubes are extracted by performing morphological operations and several redundant foreground blobs are removed in this step. Furthermore, individual subjects present in each frame are identified, and related back with the subjects present in the previous frame, thereby producing flow-tube arrays.

4. **Object Tagging**

   After actual subjects are identified in the previous phase, the subjects are classified into several popular categories using a popular deep-learning model called "You Only Look Once" model, and these tags are computed.

   A pre-trained 26-layered YOLOv3 model is used as the most common categories present in a common CCTV video footage are already present in the set of categories identifiable on a YOLOv3 trained on the standard COCO dataset.

5. **Metadata storage**

   In this stage, a connection is established to the database and the tags and flow-tubes computed are stored into the database.

2.2 Query Phase

The query phase processes the user input query, extracts the relevant tubes and generates a relevant summary. This phase is split into the following steps:

- **Tube Selection**

  The user query containing various parameters such as time period, tags and length of summary required are taken from user and relevant flow-tubes are selected from the database.

  This stage is easily implemented by writing logic to create a query with all the parameters the user specifies in the input query.

- **Rearrangement** An optimisation algorithm, in this case simulated annealing [18], is used to rearrange the flow-tubes in the time dimension to produce a summary of the desired length.

  While there are several heuristic based search algorithms are recommended for these purposes by different authors, simulated annealing remains to be the most successful and most popularly cited method. Hence, simulated annealing with a custom cost function has been implemented based on the needs.

2.2.1 Cost Function

The heart of the project resides in the rearrangement phase. A heuristic based approach has been adopted to solve this NP combinatorial problem. In order to solve a combinatorial problem using an algorithm like simulated annealing[18], an Energy or a Cost function has to be defined, which embodies the various parameters to be optimised. In this case, the two main parameters to optimise are:

- Collision: The amount of collision between events in the rearranged set of events.
- Length: The total length of the generated summary must be as small as possible.

Collision Cost is calculated as:

$$Cost_{Collision} = \frac{Collision}{TotalPixels - Collision} \tag{2.3}$$

where,

$$Collision = \sum_{i=0}^{N} \sum_{j=i}^{N} \left(\sum_{k=\max(s_i,s_j)}^{\min(e_i,e_j)} T_i[k].T_j[k] \right) \tag{2.4}$$

$$TotalPixels = \sum_{i=0}^{N} \sum_{j=i}^{N} \left(\sum_{k=\max(s_i,s_j)}^{\min(end_i,end_j)} \sum \left(T_i[k == w] + \sum T_j[k == w] \right) \right) \tag{2.5}$$

e_i: the time at which the clip i ends
s_i: the time at which the clip i ends
T_i: the 3D array (tube) representing the event in a boolean map format
w: the value of all foreground pixels in the T_i

Length Cost is calculated as:

$$Cost_{length} = \frac{Length - Lowerlimit}{Upperlimit - Lowerlimit} \tag{2.6}$$

where

$$Length = \max_{\forall i \in \tau}(end_i) - \min_{\forall i \in \tau}(s_i) \tag{2.7}$$

$$Lowerlimit = \max_{\forall i \in \tau} len_i \tag{2.8}$$

$$Upperlimit = \sum_{i}^{N} end_i \tag{2.9}$$

end_i: the time at which the clip i ends
s_i: the time at which the clip i ends

The total cost is given as:

$$TotalCost(W, Cost) = sigmoid(W.Cost^T) \tag{2.10}$$

where,

- W is weight vector of the form $[Weight_{Collision}, Weight_{Length}]$ assigning different priorities for the two factors
- $Cost$ is Cost vector of the form $[Cost_{Collision}, Cost_{Length}]$

– *sigmoid* is defined as -

$$sigmoid(x) = \frac{1}{1+e^{-x}} \tag{2.11}$$

- **Time-lapsed background generation** In this step a background is generated based on the time period and summary length required by user.

  A weighted approach, with the periods where there is most activity, is considered more heavily in generating the time-lapsed background.

- **Blending** Poisson blending[19],[20] is used to blend the rearranged flow-tubes with the time-lapsed background to generate the summary video. This summary is then saved onto the user's computer.

2.3 Summary

This chapter gives an overview of how this solution to the problem statement of summarizing CCTV video footage is structured. However there is no strict constraints on following the above prescribed framework for solving this problem. This structure optimizes the amount of time required to generate summaries and also help in rapid prototyping and validating of various techniques/models during the course of developing the application.

Chapter 3

Software Requirement Specification of CCTVVS

Software Requirements Specification is detail description of system behavior that is constructed. It integrates the functional and nonfunctional requirements for software to be constructed. The functional requirements describe what exactly the software must do and the non-functional requirement includes the constraint on the design or implementation of the system. A function is described as a set of inputs, the behavior, and outputs. A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors.

3.1 Overall Description

This section describes the general factors which affects system and the requirements. The software developed should provide means to configure order assignment, generate master files with required restrictions and must handle concurrent assignment of orders to reconciliation specialists. This section also deals with user characteristics, constraints on using the system and dependencies of the system on other applications.

3.1.1 Product Perspective

The system should be versatile and easy to use. It should be flexible and the response time should be quick. The system is composed of several modules performing different tasks and they must be well co-ordinated. The system developed should be easy to deploy and maintain. The intended end users for this are mostly CCTV operators and security managers. The system must be self-explanatory as it will be used by laymen or people with minimum technical experience.

3.1.2 Product Functions

The sole purpose of the product is to generate semantic, concise and relevant summaries for the specified time periods. The product must be versatile enough to handle sophisticated queries which considers several parameters like type of subjects, time period, summary length, degree of importance of events, duration of events to be considered etc.

3.1.3 User Characteristics

The end users of the system are mostly of to two roles, CCTV operators and security managers. Since they are not technical specialists, the user interface is expected to be simple and easy to use. It can be assumed that the general user of this application is versed in form-based inputs and has a grasp on basic English to use the interface accordingly. It is also assumed that the user can make all kinds of queries, both valid and invalid, and it is up to to the software to validate the queries and provide responsive feedback accordingly.

3.1.4 Constraints and Dependencies

The dependencies and constraints that the project has are:

- The first major constraint is the memory constraint. The number of flow-tubes it can hold in memory is directly proportional to the amount of events which can be optimized at a time.
- The system is highly dependent on having accurately tuned hyper-parameters, whether it be the threshold for motion detection or the learning (exploration/exploitation) rate used in the optimization technique, must be tuned specifically for each use-case.
- The number of categories it can recognize in a video is constrained on the diverseness of the dataset used to train the deep learning model used.
- A major dependency of the project is the Open Computer Vision (OpenCV) parallel implementation of several advanced algorithms like MOG and Poisson Blending.

3.2 Specific Requirements

This section covers all software requirements with sufficient details to be made use of by the designers to build a system to satisfy requirements. The product perspective and user characteristics do not state the actual requirements needed that the system but rather state how the product should work with respect to user convenience. The specific requirements

are actual data with which the customer and software provider can agree. The final system is expected to satisfy all the requirements mentioned here.

3.2.1 Functional Requirements

The functional requirements of the system are:

- The system must automatically validate user queries before processing them
- The system must produce video summaries based on the user query given
- The system must label each event shown in summary with relevant time stamps
- The system must be able to identify and tag events in real-time
- The system must be able to merge and rearrange events without losing meaning
- The system must be able to seamlessly blend subjects with generated time-lapsed background
- The system must be able to generate a relevant background based on the density of events distributed across the queried period of time
- The system must be able to generate summaries not longer than ten percent of the queried period length
- The user query form must be versatile enough to encompass all forms of relevant queries
- The system must have a feature to manually tune the various possible hyper-parameters from the user interface

3.2.2 Performance Requirements

Performance requirements for the system include the following:

- The system should be able to transmit the video feed with minimal loss of packets
- The delay with respect to the video transmission should be minimal
- The control signal should be transmitted at real time
- The obstacle detection should happen in real time

3.2.3 Supportability

The UI will be web-based and hence can be used on any system. While the system is built on python and open-source tools which are ubiquitous in modern day PCs, however it is recommended to use computers with large amount of RAM for fast performance, as the algorithm developed for optimization requires the system to hold significantly large arrays in memory.

3.2.4 Software Requirements

The different software requirements required by the application are as follows:

- Operating System:
  - OS X Yosemite and higher versions
  - Windows 7 and higher versions
  - Linux 16.04 LTE and higher
- Platform: Python Virtual Environment
- Language: Python
- Interface: Python GUI
- IDE/tool:
  - Visual Studio Code
  - Jupyter Notebook

3.2.5 Hardware Requirements

The various hardware requirements of the system are:

- Platform/CPU
  - Modern multi-core processor (Ex: Intel Core i7 6700HQ) on x86/ARM platform
- Memory
  - Minimum: 8GB
  - Recommended: 16GB or higher
- Storage
  - 64GB or higher for storing an entire day of video (depending on video resolution)

3.2.6 Design Constraints

The system is designed to be versatile and flexible. The design constraints include a user friendly interface. It should be able to handle requests in short time periods. The system must ensure that database is always in a consistent state. Also, the system must be built to handle all possible inputs so as to make the feature comprehensive enough for non-technical end users. The system must be able to validate input queries.

3.2.7 Interfaces

This section describes the interfaces in detail. The two types of interfaces involved are User Interfaces and Software Interfaces.

User Interfaces of the system

There is a single user interface from which the software accept parameters from the user and downloads the summary created. It must be form based and have the following parameters:

- Input source & output file path
- Time period
- Summary length
- List of tags of subjects to include
- Learning rate
- Threshold for size of events

Software Interfaces of the system

The following software interfaces are implemented in the application:

- The different modules developed in the system uses standard pre-defined data structures as interfaces between each other.
- Each module is developed as objects hence, the data internally is exposed by helper functions.

3.2.8 Non-Functional Requirements

These are the requirements that specify criteria that can be used to judge or evaluate the operation of the system rather than specific behavior and are not directly concerned with the specific functions delivered by the system.

- Efficiency: The system shall perform at best possible efficiency in all internal operations.
- Availability: The system must be up and running at all times and must accommodate users
- Security: The system must be well built with no security threats.
- Uniformity: The system must perform on any standard browser and any OS in the same way, while producing fairly similar summaries each time a similar query is made.
- Speed: The system must have a short latency period and must be responsive

3.3 Summary

The specific requirements and constraints that must be kept in mind while building the application have been detailed in this chapter. These include the hardware requirements, software requirements and functional requirements for automation of reconciliation. Also, this chapter cites the various assumptions being made by the developer of the system for auto assignment of orders. All these have to be managed while building and running the system.

Chapter 4

High Level Design of CCTVVS

Design is significant phase in development of software. It is basically a creative procedure which includes the description of the system organization, establishes that it satisfies the functional and non-functional system requirements. Larger systems divided down into smaller sub-systems contain services that are related to each other. The output in design phase describes the architecture of software to be used for the development of the common endpoint service. This section depicts the issues that are required to be covered or resolved before attempting to devise a complete design solution. The detailed design includes an explanation for all the modules. It throws light on the purpose, functionality, input and output. The software specification requirements have been studied to design an appropriate and efficient software to handle a multitude of users belonging to different user groups simultaneously accessing the system.

4.1 Design Considerations

There are several design consideration issues that need to be fixed before designing a solution for the system to be implemented. The following sections describe constraints that have heavy impact on the software, a method or approach used for the development and the architectural strategies. It also describes the overview of the system design.

4.1.1 General Constraints

General constraints which need to be considered are:

- The user should have access/authorization to use the system
- The user should have knowledge of the required inputs and the formats of the inputs
- The system must be tuned for the use-case before using it

4.1.2 Development Methods

The design method employed is highlighted in this chapter. The Data Flow Diagrams (DFD) has been the design method employed for development of the system. A data flow model is modeling system based on data transformation that takes place as the data is being processed. The notations used represent functional processing and data stores. Data flow models gives the better understanding of how data is associated with the particular process by tracking and providing the documentation.

The project has been split up into independent modules which can be independently developed, tested and validated.

4.2 Architectural Strategies

The overall organization of the system and its high level structure is provided by this section and this section also provides the key insight into the mechanism and strategies used in system architecture. The overall architecture was designed keeping in mind the need for rapid-prototyping required for a project of this scale which needs to be accomplished in a fairly short period of time.

4.2.1 Programming Language

The system involves two major segments, the frontend and backend which are built using Python, Javascript, HTML, CSS. Python is a object oriented programming languages which support wide range of data types and application programming interface for handling the data. The user interfaces are developed using HTML, CSS and JavaScript.

Other important packages which were vital for developing this project were OpenCV implementations of several image processing algorithms, Jupyter's interactive python interpreter and NumPy package used for performing fast and vectorised operations on images.

4.2.2 User Interface Paradigm

The GUI of the system is a form based interface. It is light-weight and fairly simple to develop. However, it is built to only take valid input queries from the user. Therefore the UI paradigms to be achieved are consistency, clarity and simplicity.

4.2.3 Error Detection and Recovery

Error detection and recovery is an important aspect of the implemented project. User form validation has been implemented in the very beginning of the pipeline. Almost all

kinds of errors will be weeded out in the very beginning of the pipeline. Internal error detection is done using the exception handling clauses provided by Python and using plenty of datatype asserts wherever possible. Since the project is divided into clearly defined modules with pre-defined interfaces, debugging and detecting errors is very easy.

4.3 System Architecture

This section is focused on basic structure of model in the system. It aims to identify major modules in system and communication flow amongst these modules.

We have used DFDs to architect the system. The data associated with the system is identified, and the flow of the data between the different processes is studied.

4.4 Data Flow Diagram

A Data Flow Diagram (DFD) is graphical representation of the ”flow” of data through an information system. Data Flow models describe how data flows through a sequence of processing steps. DFD is composed of four elements, which are process, data flow, external entity and data store. With data flow diagram, the users can easily to visualize the operations within the system, what can be accomplished using the system and implementation of the system. DFDs provide the end users an abstract idea regarding the data that is given as input to the system, the effect that the input will ultimately have upon the whole system.

The level 0 diagram shows the main data streams in the system. The level 1 diagram shows the input and output data in the two main processing pipelines. The level 2 diagram explains the data transformation in each pipeline step-by-step.

4.4.1 Data Flow Diagram – Level 0

The level 0 DFD describes general operation of the system. It represents the system and user and the inputs and outputs between the user and the system.

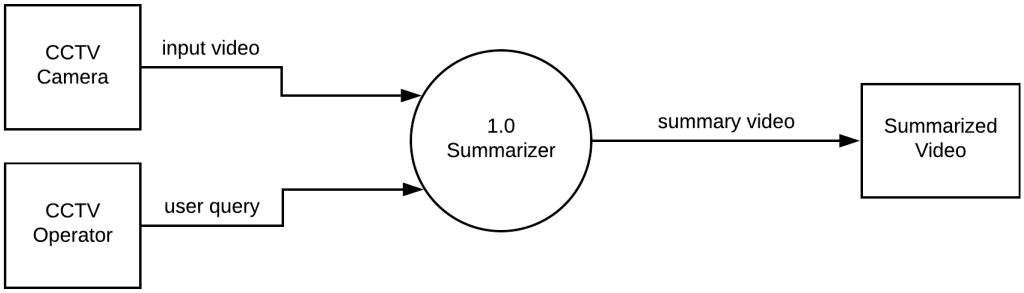


Figure 4.1: Data Flow Diagram - Level 0

Figure 4.1 gives an abstract representation of the system. The input video from CCTV is fed into the summariser which gives the video summary as output, based on the query from the CCTV operator/user.

4.4.2 Data Flow Diagram – Level 1

The Level 1 DFD describes the system more in detail than the Level 0 DFD. It specifies the two main phases involved in the system. This is as shown in fig 4.2.

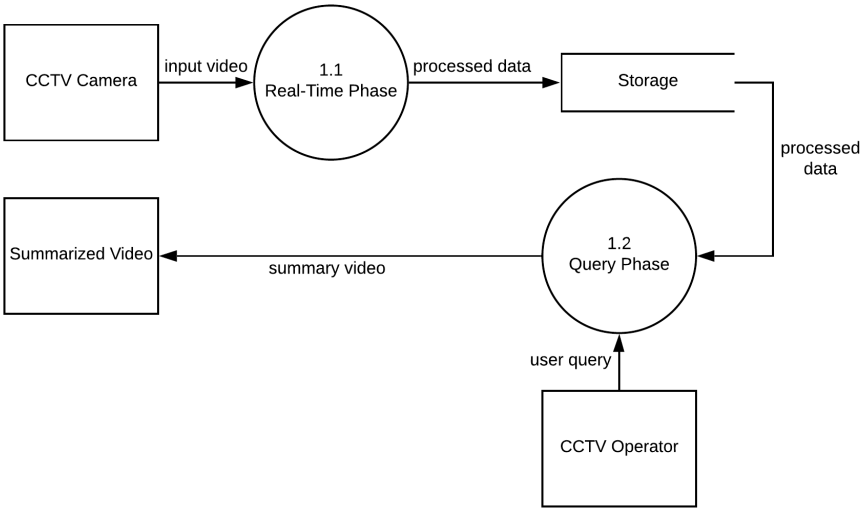


Figure 4.2: Data Flow Diagram - Level 1

The real-time phase takes in the input video from the CCTV camera and processes the data for the next phase. Different methods like motion detection, background masking, tube extraction and object detection are applied on the given input video. The preprocessed data is then stored in the database. Next, a query is given as input for the next phase. Using the preprocessed data from phase one, a summarized video is generated.

4.4.3 Data Flow Diagram – Level 2

Figures 4.3, 4.4 represent the level 2 data flow diagrams.

The processes in level 1 are expanded here. The Level 2 DFD for Real-time phase of the video summariser is shown in figure 4.3.

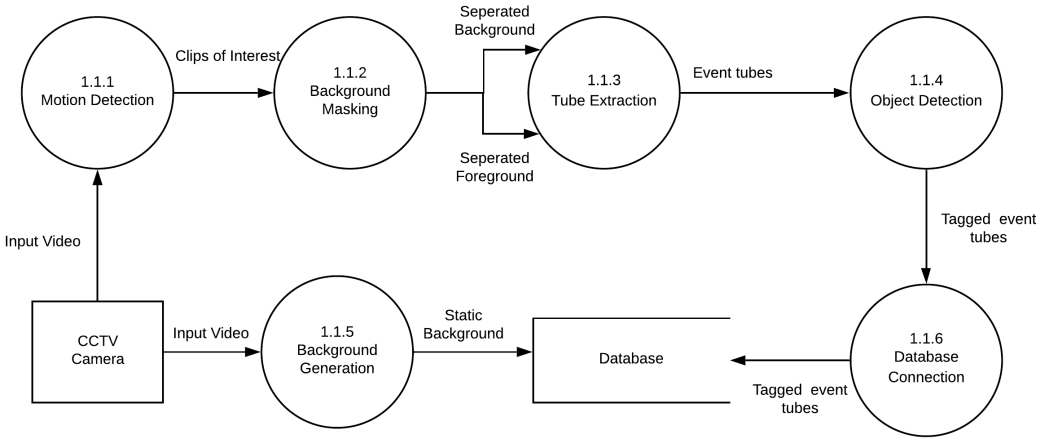


Figure 4.3: Data Flow Diagram - Level 2 (Depth image processing)

Motion is first detected frame by frame be using a threshold of number of active pixels in the binary mask. From these clips of interest, the foreground is extracted using MOG2. Events of interest are then extracted as tubes. YOLO object detector is then applied to these tubes to create tagged even tubes which are stored in the database. Simultaneously, a static background is generated from the input footage, which is also stored.

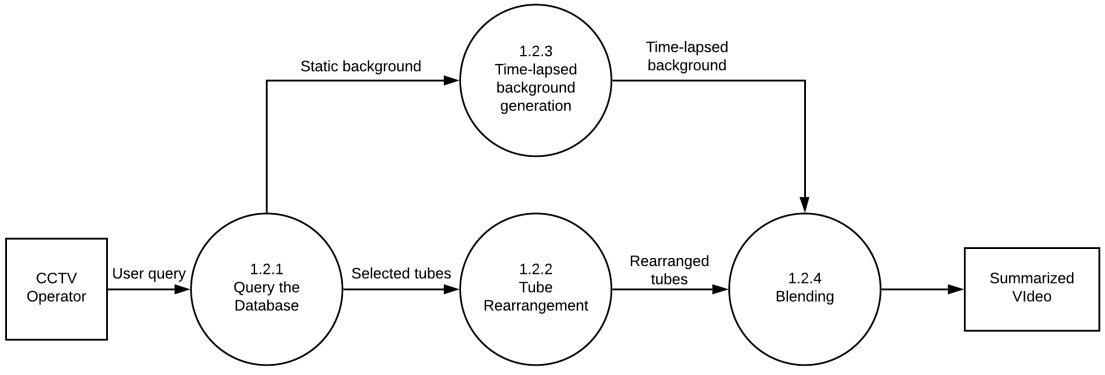


Figure 4.4: Data Flow Diagram - Level 2 (Colour image processing)

Figure 4.4 shows level 2 DFD for Query phase of the Video summariser. Based on the user query, tubes of interest are first selected from the database. These tubes are then rearranged using simulated annealing. While this process is being completed, a time-lapsed background is also formed. In the next step, the rearranged tubes are blended into

the time-lapsed background by Poisson blending. The generated summarized video is then stored.

4.5 Summary

The above data models depict how data is processed by the system. This constitutes the analysis level. The notations applied above represent functional processing, data stores and data movement amongst the functions. The purpose of chapter is to describe major high- level processes (the two phases – Real time and Query) and their interrelation in the system. All the above mentioned levels of DFDs illustrate these.

Chapter 5

Detailed Design of CCTVVS

In the Detailed Design phase, the internal logic of every module specified in High Level Design (HLD) is determined. Specifically, in this phase the design of each module, the low-level components and subcomponents are described. After determining HLD graphical representation of the software system being developed is drawn. Each module‘s input and output type, along with the possible data structures and algorithms used are documented during the detailed design phase. The following sections provide such information of the modules.

5.1 Structure Chart

The structure chart shows the control flow among the modules in the system. It explains all the identified modules and the interaction between the modules. It also explains the identified sub-modules. The structure chart explains the input for each modules and output generated by each module.

In the system, there are six sub modules. They are Video file reader, Background creator, Motion Detector, Tube generator, Tube rearrangement module and Summary maker. The description of the sub modules, the flow of data and the results of each sub module are shown in Figure 5.1. The modules each receive input and process the input and produce output which is seen to end users in a web dashboard.

The video is read from an input source, frame by frame, it is then sent to the motion detection module and identified as either clip of interest or ignored. These clips are then processed to extract useful information such as tags and flow-tubes which are then stored into a database. Further, when a query is made, relevant flow-tubes are gathered, rearranged and a relevant background is simultaneously generated. These two are then seamlessly blended to produce the summary.

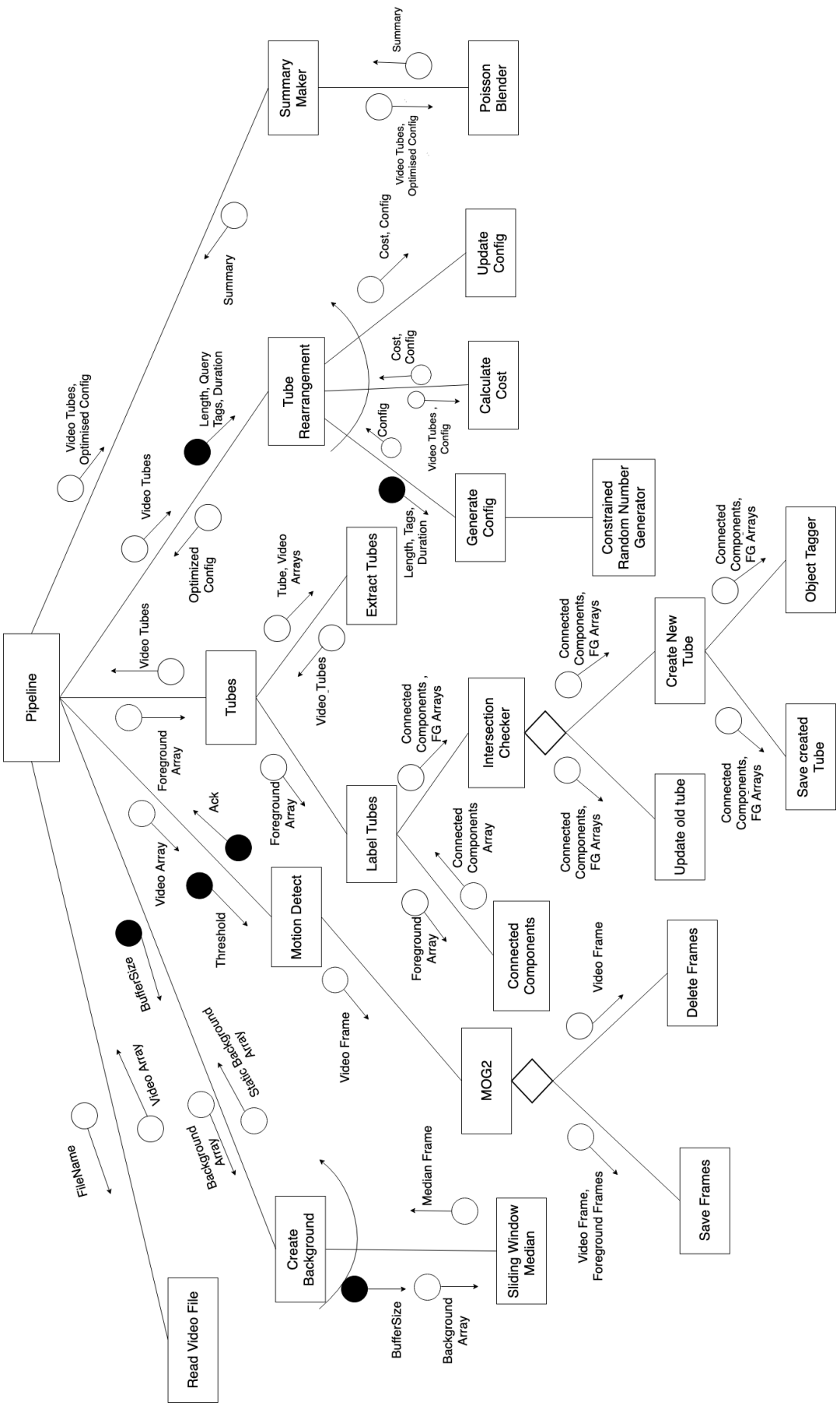


Figure 5.1: Structure Chart

5.2 Functional Description of Modules

The internal working of certain core modules is explained in this section. It also describes the software component and subcomponent of the system.

5.2.1 Motion Detection Module

This is the main module of the system which is responsible identifying clips of interest in a sparse CCTV footage.

- **Purpose:** The purpose of this module is to decide whether each frame is relevant or not.
- **Input:** The input to this module are frames of input video source and timestamp
- **Output:** The output is whether the input should be saved for further processing or ignored
- **Functionality:** The functionality of this module is to detect motion and save those clips of interest
- **Flowchart:** The flowchart shown in figure 5.2 explains the motion detection module. The module starts by reading a frame and applies the stored MOG model onto the frame. If the generated background mask has more foreground pixels than a specified threshold then it is considered as a frame with motion and is stored as clip of interest.

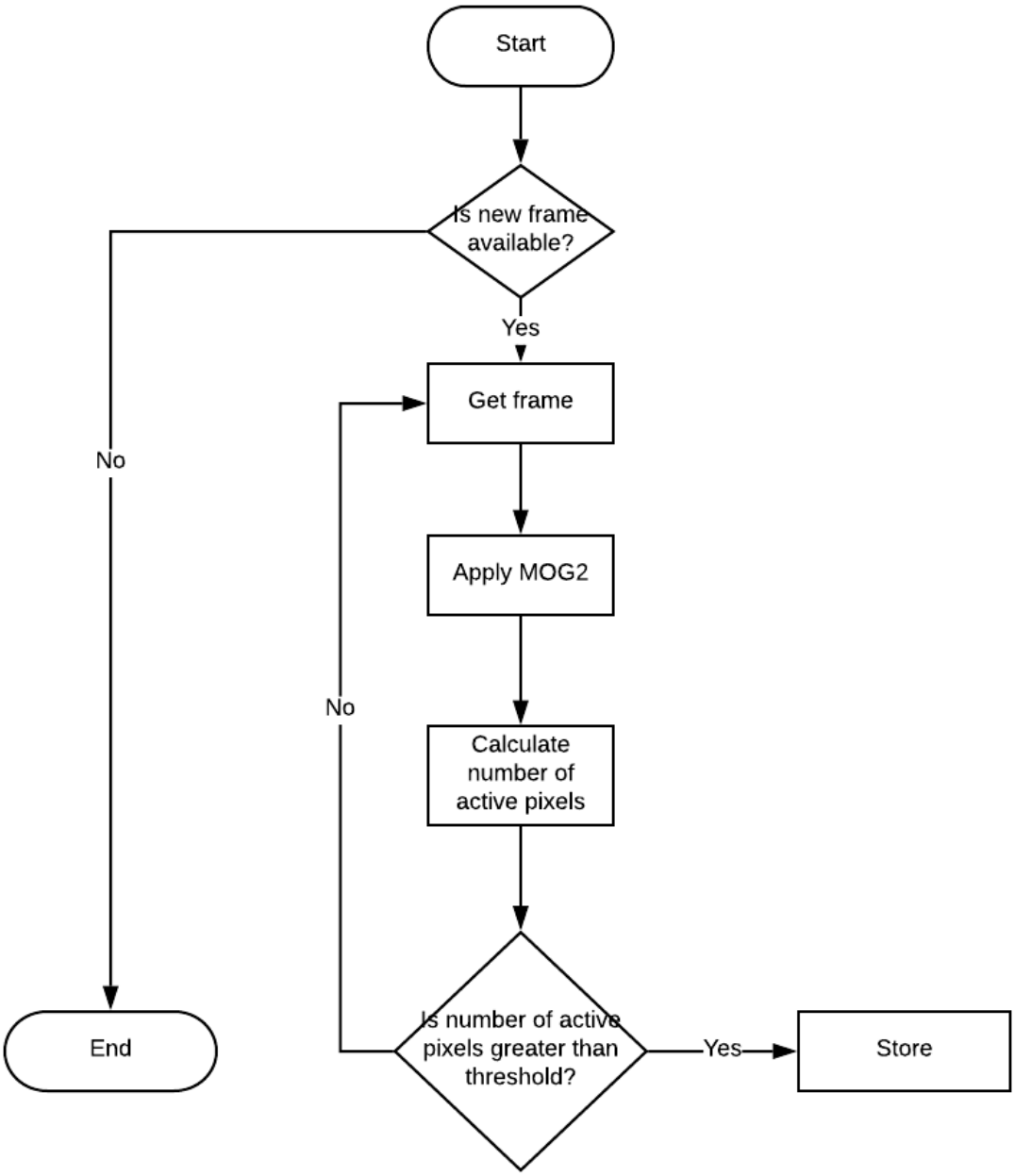


Figure 5.2: Motion Detection Flowchart

5.2.2 Background Creation Module

This is the second module of the system which is responsible for the generation of a time-lapsed background.

- **Purpose:** The purpose of this module is to generate a time-lapsed background for the input time-period and specified duration to overlay the rearranged flow-tubes.
- **Input:** The input is a stream of frames in the specified time-period, and length of

summary which is required to overlay on this background.

- **Output:** A background clip representing the background for the given duration condensed into a clip as long as the rearranged summary generated.
- **Functionality:** The module implements a queue type buffer, which is continuously updated and median value for each pixel row in the buffer is calculated and stored as background.
- **Flowchart:** The flowchart shown in the Figure 5.3 explains the procedure followed in the Background creation module. The module uses a buffer which stores the history of frames from previous 4 minutes or more, depending on the length of summary, and background is calculated by pixel-wise calculation of the median across the whole buffer. The buffer is updated as new frames are read into the queue data structure. The median implementation is parallelized to optimize performance.

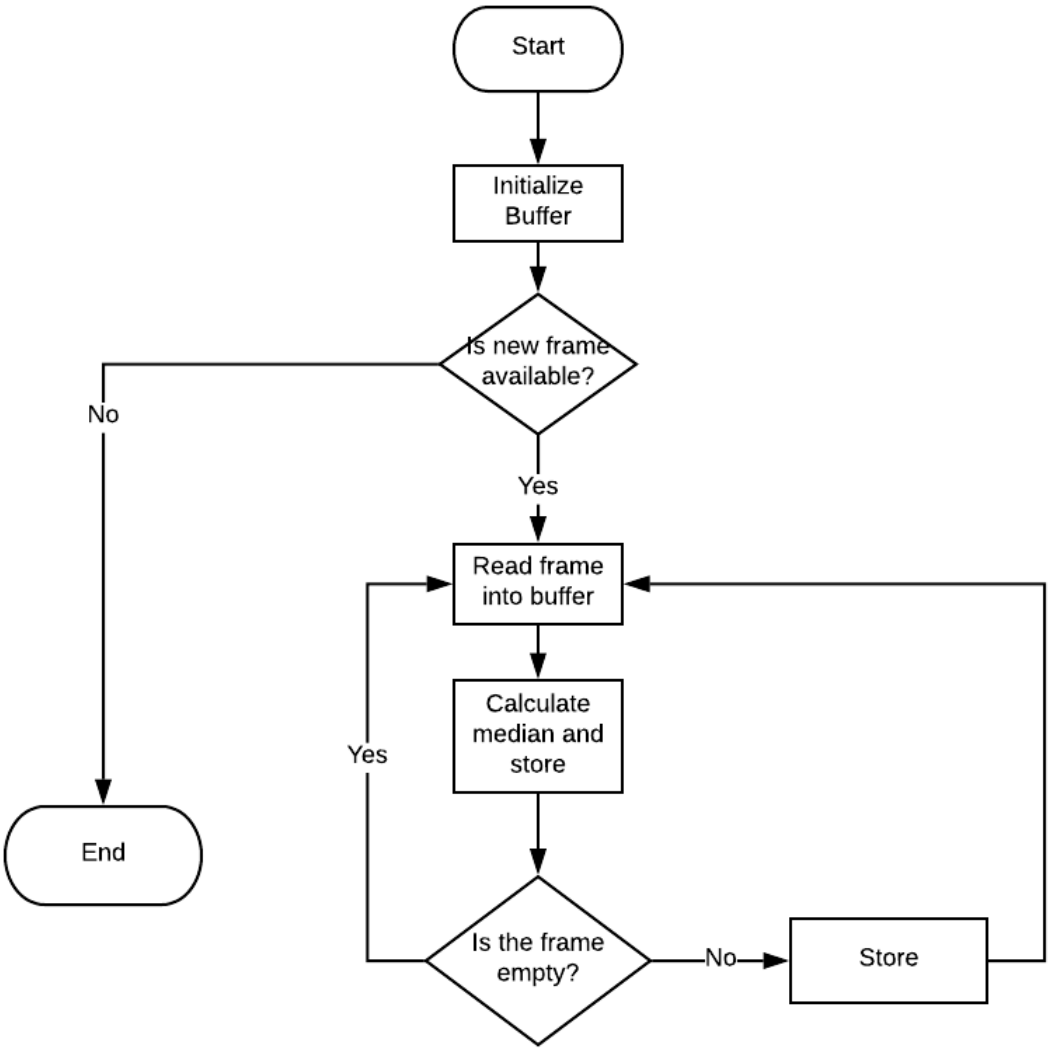


Figure 5.3: Background Creation Flowchart

5.2.3 Optimisation Module

This is module of the system handles the optimization and rearrangement of selected tubes.

- **Purpose:** The purpose of this module is to rearrange tubes and produce a compact meaningful summary.
- **Input:** 3D arrays of flow-tubes extracted and their original timestamps.
- **Output:** The module computes an optimal configuration of the flow-tubes based on a pre-defined cost function to generate the summary.
- **Functionality:** The module generates an optimized configuration of flow-tubes which is both condense yet meaningful in nature.

- **Flowchart:** The flowchart shown in the Figure 5.4 explains the procedure followed in the optimization module. A popular heuristic based search algorithm has been used to implement this module called Simulated Annealing. The algorithm trades-off from exploration to exploitation as the number of iterations and epochs increases. The module uses a pre-defined cost function to evaluate a fitness score for each configuration and the global optimized configuration (GOC) is updated as per the sigmoid value obtained from applying the sigmoid function on the difference in fitness value from the current and previous globally optimized configurations. Higher the sigmoid value, higher are the chances of the GOC getting updated.

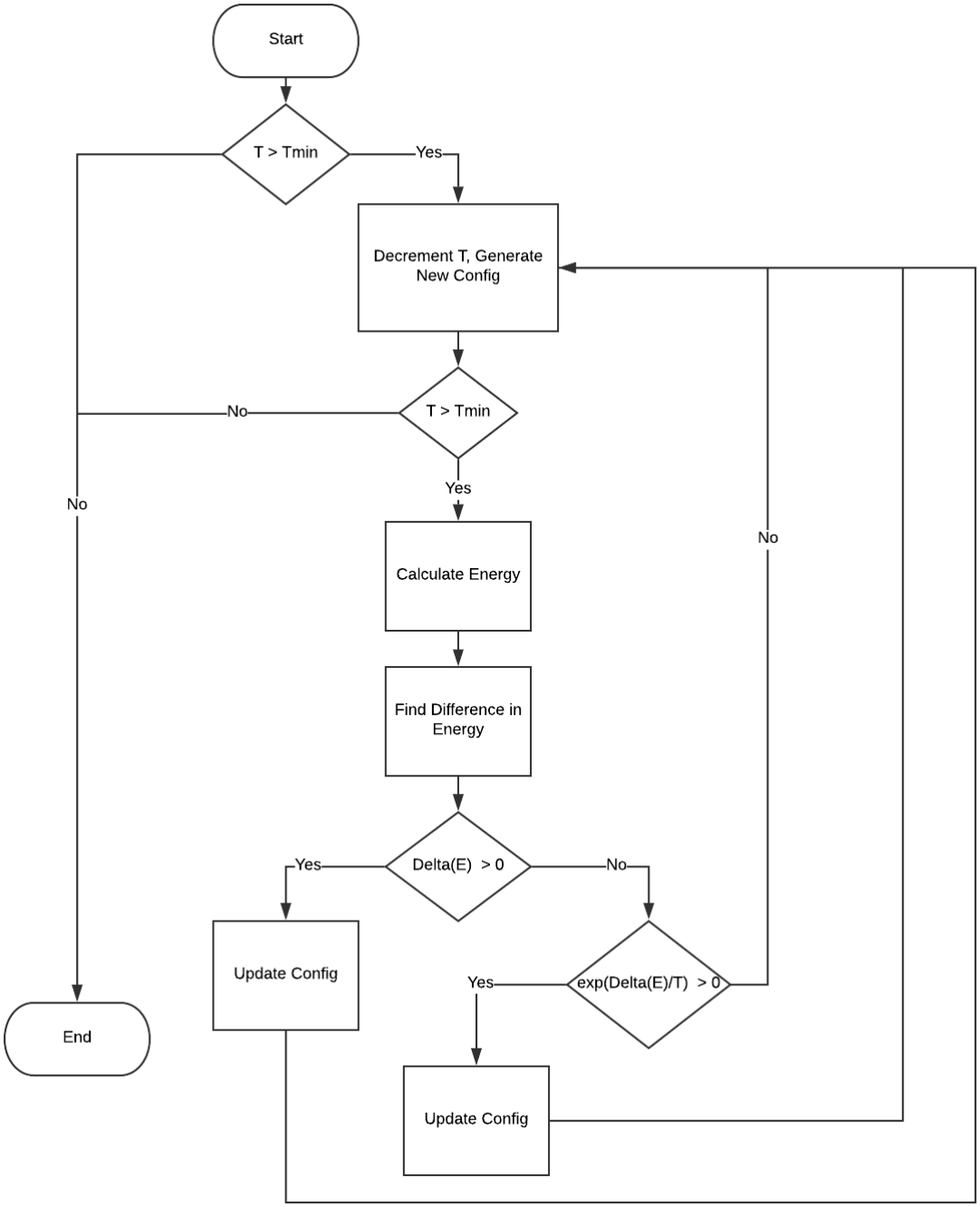


Figure 5.4: Optimisation Module Flowchart

5.2.4 Tube Extraction Module

This module of the system which handles the extraction of flow-tubes of several subjects present in the clips of interest identified by motion detection module.

- **Purpose:** The purpose of this module is to identify and extract flow-tubes from each clip of interest.

- **Input:** 3D arrays of MOG masks of video frames of the clips of interest.
- **Output:** 3D arrays of masks which represent the flow-tubes for each subject tracked in each clip of interest.
- **Functionality:** The module identifies flow-tubes whose length is beyond the specified user-threshold and extracts the flow-tubes as mask arrays and stores them for further processing later.
- **Flowchart:** The flowchart shown in the Figure 5.5 explains the procedure followed in the tube extraction module. The module's core component is the connected component search implementation. Each frame is processed to identify the number of individual blobs present in the frame and are correlated with the blobs seen in the previous frame to track existing subjects and create new subjects as and when they occur in each clip of interest. After having individually identified each subject by annotating the subjects as such in the input video feed, individual mask flow-tube arrays are created for each subject and then stored with their respective timestamps.

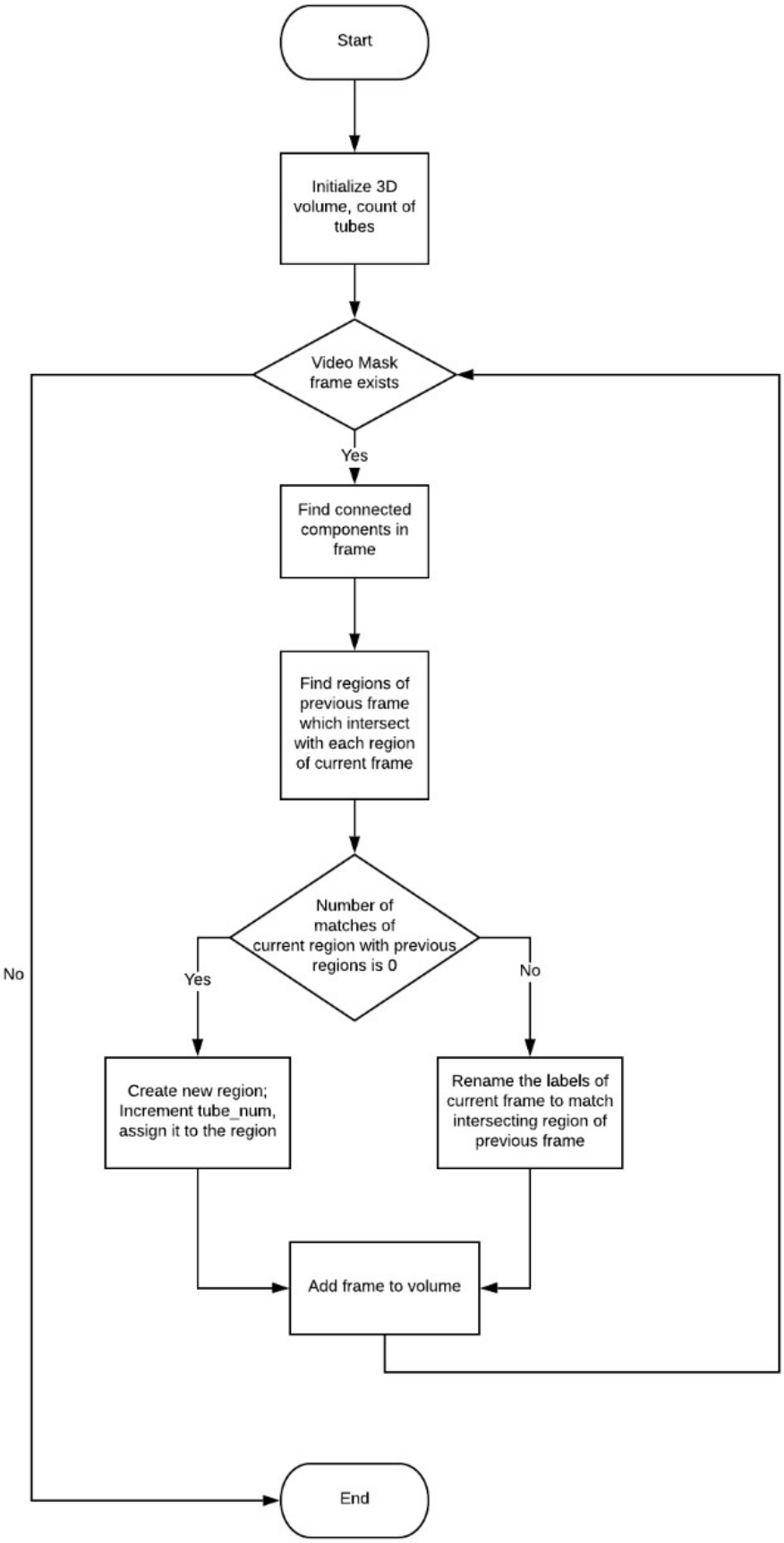


Figure 5.5: Tube Extraction Flowchart

5.3 Summary

The internal working of the application's three modules with the necessary data flow through each of them has been described in this chapter. A clear view on control flow within system was conveyed by the structure chart with the functionality of its modules being explained. The flow charts explain the working of each module with flow of control in the module specified which gives a complete understanding of the functioning.

Chapter 6

Implementation of CCTVVS

The implementation phase is significant phases in the project development as it affords final solution that solves the issues. In this phase the low level designs are transformed into the language specific programs such that the requirements given in the software requirements specification are satisfied. This phase entails actual implementation of ideas that were described in analysis and design phase. The technique and the methods that are used for implementing software must support reusability, ease of maintenance and should be well documented.

6.1 Modules and Classes

We have split the code into different functional modules for easy development, testing and maintenance of the code.

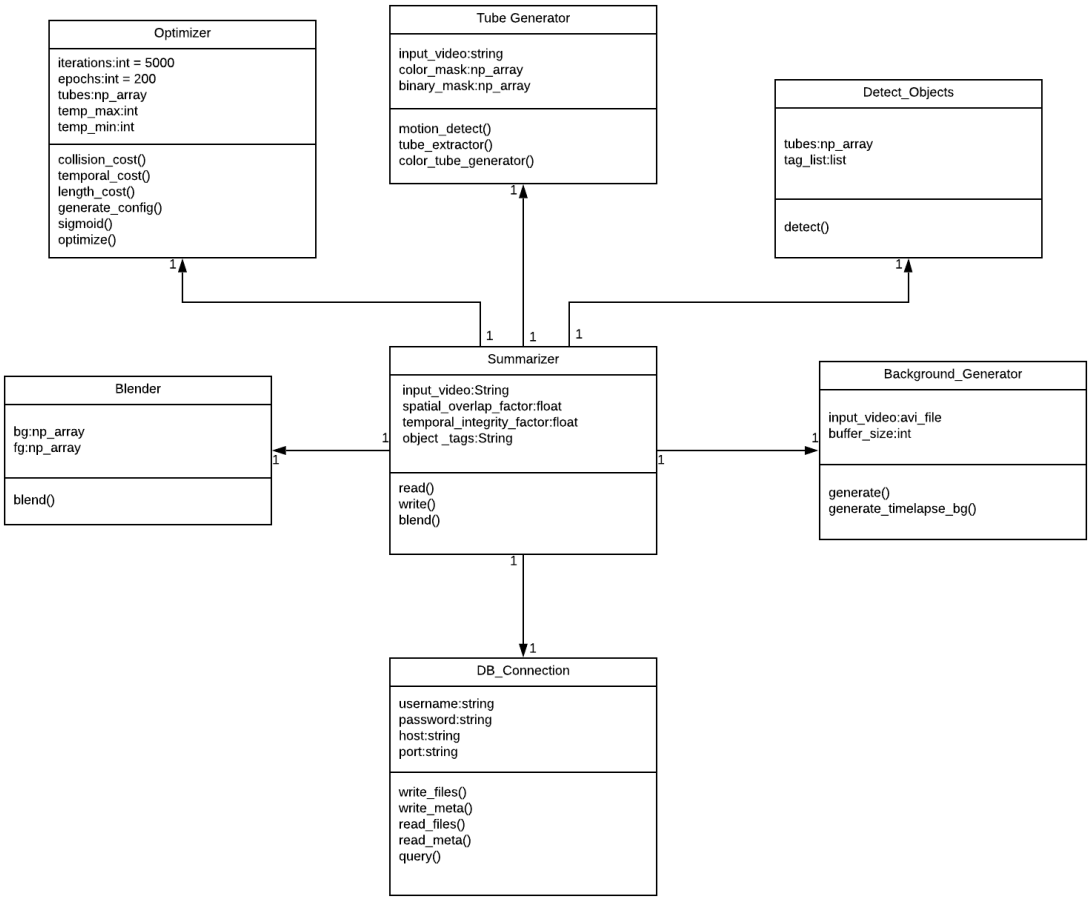


Figure 6.1: Classes / Modules of CCTVVS

The classes of the summarisation system are as shown in Figure 6.1. The central summariser object creates objects other modules mentioned in the figure and uses them in a sequence mentioned in chapter 2. Each constructor of the classes defined above, have strict assert statements to ensure the data they receive are of the proper type before any kind of operations are performed on them.

6.2 Algorithms

6.2.1 Simulated Annealing Algorithm

Algorithm 1 shows the simulated annealing algorithm which is used for the tube optimisation process.

6.2.2 Tube Extraction Algorithm

Algorithm 2 shows the tube extraction algorithm which is used for the extracting the event tubes from the labelled motion volume.

Result: Optimised Configuration
Let $Config_{current} = Config_{inital}$; **for** $T \leftarrow T_{max}$ **to** T_{min} **do**
  $E_{current} = E(C_{current})$ $C_{next} = next(C_{current})$ $E_{next} = E(C_{next})$
  $\Delta E = E_{next} - E_{curr}$ **if** $\Delta E > \theta$ **then**
    $C_{current} \leftarrow C_{next}$
  else if $e^{\frac{-\Delta E}{T}} > rand(0,1)$ **then**
    $C_{current} \leftarrow C_{next}$
end

Algorithm 1: Simulated Annealing

Result: 3D space-time volume with connected labelled events
/* a 3d array representing the time-space volume with
foreground represented as 1 and background as 0 */
input: VideoMask
$volume \leftarrow empty\_list()$ $tube\_num \leftarrow 0$ $prev\_frame \leftarrow zeros\_like(VideoMask[1])$
$prev\_count \leftarrow 0$
for $FrameinVideoMask$ **do**
  $labelled\_frame, count = findConnectedComponents(Frame)$
  for $iinrange(0, count)$ **do**
    $matches \leftarrow empty\_list()$ **for** $jinrange(0, prev\_count)$ **do**
      if $\frac{Sum((labelled\_frame[k==i]*prev\_frame))}{Sum(prev\_frame)} > Threshold$ **then**
        $matches.append(j)$
      end
    end
    $matches\_dict[i] \leftarrow matches$
  end
  $prev\_frame \leftarrow Frame$ $prev\_count \leftarrow count$
  for $regioninmatch\_dict$ **do**
    if $len(match\_dict[region])>1$ **then**
      for $indexinmatch\_dict[region]$ **do**
        for $frameinvolume$ **do**
          /* Update labels of tubes in previous frame */
          $frame[frame == index] = tube\_num$
        end
        /* Update the dict */
        $match\_dict[region] = [tube\_num]$
      end
    end
  end
  /* There are no intersection issues; Proceed normally */
  for $regioninmatch\_dict$ **do**
    $cur\_img \leftarrow labelled\_frame$ $cur\_img\_output \leftarrow labelled\_frame.clone()$
    $cur\_labels = empty\_list()$ **if** $len(match\_dict[region]) == 0$ **then**
      /* New region found since there is no match to any
      previous region */
      $tube\_num+ = 1$ $cur\_img\_output[cur\_img == region] = tube\_num$
    end
    else
      /* Some previous region was found, relabel the region
      */
      $cur\_img\_output[cur\_img == region] = match\_dict[region][0]$
    end
  end
  $volume.append(cur\_img\_output)$
end

Algorithm 2: Tube Extraction

6.3 Programming Language Selection

The programming languages chosen to implement the project is Python. Python is one of the most useful languages in the current age and time with extensive support for fast development. Some of the benefits that Python provides which were key for choosing the same are:

- Simple, easy and highly readable program syntax
- Rapid development and prototyping process
- Jupyter notebook supports step-by-step execution of code for easier debugging
- Support of OpenCV wrapper

6.4 Platform Selection

The system has a basic user interface for the user input & query, and the backend system to process the input and generate the video summary. Python runs on all 3 major computing platforms Windows, Linux and MacOS. The code has been developed and tested on Windows and Mac platform using Python 3.6.

6.5 Code Conventions

This section discusses the coding standards followed throughout the project. It includes the software applications that are necessary to complete the project. Proper coding standards should be followed because large project should be coded in a consistent style. This makes it easier to understand any part of the code without much difficulty. Code conventions are important because it improves readability in software, allowing the programmers to understand code clearly.

6.5.1 Naming Conventions

Naming conventions helps programs in understandable manner which makes easier to read. The names given to packages, scripts, graphs and classes are to be clear and precise so that their contents can easily be understood. The project uses both Java and Python, and the naming convention followed in the two are slightly divergent from each other.

The conventions followed for this project are as follows:

- **Classes:** Class names are nouns. The upper camel casing method is followed, in which the first letter of every word is in capital, including the first word. Example: SimulatedAnnealing.

- **Methods:** Methods should be a verb. For methods, snake case is followed, where the names are in lowercase and multiple words are separated by underscores. Example: make_summary()
- **Variables:** snake case is followed, where the names are in lowercase and multiple words are separated by underscores. Example: selected_tubes

6.5.2 File Organization

The code used to implement the project was organized into multiple files based on the functionality and the module to which the methods belonged. Each file contains a class which contains the required attributes and methods. Ex: SimulatedAnnealing class contains attributes like T_max, T_min, iterations etc and methods like calc_cost(), optimize() etc

6.5.3 Declarations

Standard declaration conventions are followed while coding. Standard names are given which make it easy to understand the role of each entity declared. Multiple declarations per line are not allowed because of commenting and to reduce ambiguity.

6.5.4 Comments

Comments are necessary part of any coding conventions as it improves the readability of the code developed. In the project files, thanks to the integrated development environments, commented areas are printed in grey by default, so they are easy to identify. In Python comments start with a '#'. There are keyboard shortcuts and mouse options provided to comment out or uncomment blocks of code with ease. Comments are used for explaining what function a certain piece of code performs especially if the code relies on implicit assumptions or otherwise perform subtle actions.

6.6 Difficulties Encountered and Strategies Used to Tackle

This section discusses some of the difficulties encountered while developing this project.

6.6.1 Static Background Generation

The system has to generate a static background from the moving background, and a temporal median is proposed for the same. Since the same was computationally intensive, the Mixture of Gaussians in OpenCV has been used for estimating the static background, which performs better.

6.6.2 Simulated Annealing

The video summariser optimizes the tubes which are extracted from the input video, and this is a NP-complete problem, where every single possibility can't be examined. A heuristic based optimization algorithm such as simulated annealing is required. A set of cost functions have been created which calculate the collision cost and length cost, which are used by the optimization algorithm. Simulated annealing, which is a probabilistic technique, is used for estimating the global optimum. The cost of a configuration is evaluated and then the decision to update the configuration is based on other parameters. The simulated annealing algorithm is optimised by using a resized image, and by parallelising it for faster processing.

6.7 Summary

This chapter deals with the programming language used which is Python, the development environment and the code conventions followed in the two languages during implementation of the application. It also explains the difficulties encountered in the course of implementation of the system like background generation and simulated annealing and the strategies used to handle them.

Chapter 7

Software Testing of CCTVVS

The aim of Software Testing is to detect defects or errors testing the components of programs individually. During testing, the components are combined to form a complete system. At this particular stage, testing is concerned to demonstrate that the function meets the required functional goals, and does not behave in abnormal ways. The test cases are chosen to assure the system behavior can be tested for all combinations. Accordingly, the expected behavior of the system under different combinations is given. Therefore test cases are selected which have inputs and the outputs on expected lines, inputs that are not valid and for which suitable messages must be given and inputs that do not occur very frequently which can be regarded as special cases. For testing software, various test strategies are to be used such as unit testing, integration testing, system testing and interface testing.

In this chapter, several test cases are designed for testing the behavior of all modules. When all modules are implemented completely, they are integrated and deployed on Tomcat server or on Jetty Container. Test cases are executed under same environment. Test cases mainly contain tests for functionality of all modules. Once the application passes all the test cases it is deployed on the production environment for actual real time use.

7.1 Test Environment

The test environment is important to get right, because a major problem faced by most python tools are dependency issues. The Anaconda distribution of python is used to create a virtual environment with all the packages pre-installed. This way these environments can be easily exported to other systems as well. In the future, the possibility of writing a dockerfile to generate an linux image with all the existing dependencies pre-installed can be explored. Additionally, Jupyter notebook was used for modular development and testing as it provides an interactive way of developing and testing code.

7.2 Unit Testing of Main Modules

Unit test is the verification effort on the smallest unit of software design, the software modules. Unit testing ensures that the bugs that occur can be pinpointed easily since the code tested on is a small unit. The section describes some of the unit tests run with test case details and brief explanations.

7.2.1 Unit testing of real-time phase modules

The following tables show the test cases for Real-Time phase on which this testing is performed.

Read Video Feed

Table 7.1 shows the test case details for testing the Video Reader sub module. This test was successful. Here, the input was the video file which was successfully read.

Table 7.1: Unit Testing of Video Read

Sl. No	**1**
Name of the Test case	Read video
Feature being Tested	Proper loading of video from disk
Description	Loading video smoothly, quickly and without exceptions
Sample Input	Video File
Expected Output	Load list of frames into memory if file exists else error that file not found
Actual Output	As expected
Remarks	Test case passed successfully

Motion Detection Test

Table 7.2 shows the test case details of Motion detection sub-module. The input to this sub-module is the video frame list. The sub-module works by applying a Mixture of gaussians algorithm on a buffer of video frames to generate a mask of frames with action marked as white. This test was successful.

Table 7.2: Unit Testing of Motion Detection

Sl. No	**2**
Name of the Test case	Motion detection and masking
Feature being Tested	Detection of motion and generation of mask of movement in the frame
Description	For each frame, motion should be detected and a mask of which part of the frame has motion should be created
Sample Input	List of video frames
Expected Output	List of binary masked frames
Actual Output	As expected
Remarks	Test case passed successfully

Background Generation Test

Table 7.3 shows

Table 7.3: Unit Testing of Motion Detection

Sl. No	**3**
Name of the Test case	Background video generation
Feature being Tested	Generation of static background
Description	For a given video, a static background is to be generated from the video with movement so that events can be blended into it, to generate the summary
Sample Input	List of video frames
Expected Output	List of frames with minimal movement / no movement
Actual Output	Static frames, with only slow/gradual changes like shadows changing
Remarks	Test case passed successfully

Tube Labelling Test

Table 7.4 shows the test case details for tube labelling. The input is a list of masked frames representing optical flows. This sub-module identifies connected components and labels individual optical flows. It outputs a dictionary with each individual identified optical flow ids in place of each pixel. This test was also successful.

Table 7.4: Unit Testing of Tube Labelling

Sl. No	**4**
Name of the Test case	Tube labelling
Feature being Tested	Labelling of detected motion with unique event IDs
Description	From the motion mask, the different events occurring in the same frame must be distinguished by assigning a unique even ID to each event
Sample Input	List of masked frames with pixel value of 255 representing the motion
Expected Output	List of frames with event ID in each pixel in place of 255 value
Actual Output	As expected, with colliding objects categorized as into the same event
Remarks	Test case passed successfully

Tube Extraction Test

Table 7.5 shows the test case details for tube extraction. This sub-module extracts individual optical flows from the list of labelled frames output by the tube labelling module. The output is a list of individual tube objects. This test was also successful.

Table 7.5: Unit Testing of Tube Extraction

Sl. No	**5**
Name of the Test case	Tube Extraction
Feature being Tested	Extract every event from the labelled volume into separate tubes
Description	From the labelled volume, the different events are extracted into separate tubes with the original start time being stored
Sample Input	List of labelled frames
Expected Output	Multiple tubes, each tube a set of frames, which has the motion mask of individual events
Actual Output	As expected
Remarks	Test case passed successfully

Colour Tube Generation Test

Table 7.6 shows the test case details for colour tube generation. This sub-module is going to extract color tubes from the individual optical flow tubes extracted in the previous sub-module. The extracted color tubes are added into the the attributes of each of the tube object in the list of tube objects passed to the module. The sub-module returns the updated list of tube objects. This test was also successful.

Table 7.6: Unit Testing of Colour Tube Generation

Sl. No	**6**
Name of the Test case	Colour Tube Generation
Feature being Tested	Generate a colour/object tube using original video and individual tube mask
Description	Bitwise AND operation is used to extract only the required part of original video
Sample Input	List of original frames, List of individual event tubes
Expected Output	Multiple tubes, each tube a set of frames, which has the coloured image of individual events
Actual Output	As expected
Remarks	Test case passed successfully

Object Detection Test

Table 7.7 shows the test case details for object detection sub-module. This sub-module takes a list of object tubes and passes each color tube in these objects through a pre-trained convolutional neural network which identifies objects. It updates the tags attributes in each tube object and returns the list of updated tube object list. This test was also successful.

Table 7.7: Unit Testing of Object Detection

Sl. No	**7**
Name of the Test case	Object Detection
Feature being Tested	Detection of objects in colour tubes
Description	YOLOv3 object detector is run on few frames in each colour tube to detect the objects
Sample Input	Colour tubes
Expected Output	Set of tags associated with each tube
Actual Output	All objects detected correctly in majority of cases
Remarks	Test case passed satisfactorily

7.2.2 Unit testing of query phase modules

The following tables show the test cases for query phase on which this testing is performed.

Simulated Annealing Test

Table 7.8 shows the test case details for simulated annealing sub-module test. The input to this module are a list of tube objects. The sub module rearranges these tubes in time such that it optimizes a loss function designed to reduce collision and length of the summary generated. The sub-module generates an optimal configuration for the list of tubes provided. This test was successful.

Table 7.8: Unit Testing of Simulated Annealing

Sl. No	**8**
Name of the Test case	Simulated Annealing
Feature being Tested	Optimize the placement of tubes
Description	The selected colour tubes are to be placed optimally to prevent overlap of events
Sample Input	Tubes with their lengths, with start time set to 0
Expected Output	Tubes with optimal start times
Actual Output	Optimization works well and avoids collision of events in most cases
Remarks	Test case passed successfully; Further testing required for longer videos and videos with more events

Image Blending Test

Table 7.9 shows the test case details for Image blending sub-module test. The module takes input of a list of object tubes and the optimized configuration generated by the previous sub-module. It blends the color tubes present in each tube object to generate a summary which is then blended with a static weighted background generated in the previous phase. The output is a final summary with time-stamps on each event shown. While the module performs relatively well in cases of fixed camera settings but however has problems with shaky footage. It has room for improvement but the test for this module remains largely successful.

Table 7.9: Unit Testing of Image Blending

Sl. No	**9**
Name of the Test case	Image Blending
Feature being Tested	Blending of tubes into the static background
Description	The selected colour tubes are to be blended into the static background at the optimal time determined by simulated annealing
Sample Input	Colour tubes and metadata, static background
Expected Output	Events blended into the background seamlessly at the time determined by simulated annealing
Actual Output	Events blended in at the correct time, but not seamlessly in all cases, especially at the edges of the frame
Remarks	Test case passed, but blending function can be improved

7.3 Integration Testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The

objective is to take unit tested components and build a program structure.

7.3.1 Integration testing of tubes module

In order to standardize the interfaces between each module, a tube class has been created. Each module passes around a list of these tube objects. An iterative development process was adopted which added features along the way, by just changing a single file at a time. In this test, the possible errors in the interfacing between the main function and the tube module is being tested.

Table 7.10 shows the integration testing of tube module and main module. The input is a list of video frames and the output is a list of tube objects which will be used in the rest of the process.

Table 7.10: Integration Testing of Tubes Module

Sl. No	**1**
Name of the Test case	Tube labelling and extraction
Feature being Tested	Interface between main module and tube module
Description	Data is to be passed between the modules using a defined data structure
Sample Input	Input video frames
Expected Output	Tube data structure with extracted tube mask, and colour tubes
Actual Output	As expected
Remarks	Test case passed successfully

7.3.2 Integration testing of detection module

Table 7.11 shows the test in which the interfacing between the detection module and the main module is checked.

Table 7.11: Integration Testing of Detection Module

Sl. No	**2**
Name of the Test case	Object detection
Feature being Tested	Interface between main module and Object detection module
Description	Object detection module should return the detected tags
Sample Input	Tube data structures
Expected Output	Tube data structures with tags of detected objects
Actual Output	As expected
Remarks	Test case passed successfully

7.4 System Testing

System testing is the testing in which all modules, that are tested by integration testing are combined to form single system. The system is tested such that all the units are linked properly to satisfy user specific requirement. This test helps in removing the overall bugs and improves quality and assurance of the system. The proper functionality of the system is concluded in system testing.

The whole system is evaluated in this system testing, with all main modules being tested. Two system tests have been devised, as there are two main modules which are relatively well decoupled. The system testing for the real time phase is as shown in Table 7.12.

7.4.1 System testing of real-time phase

Table 7.12 shows the system testing for real-time phase. Here all the modules in the real-time phase are combined and tested. The system should automatically generate a list of updated object tubes and a static background video from the input video in real-time. The real-time phase performed successfully.

Table 7.12: System Testing of Real-Time Module

Sl. No	**1**
Name of the Test case	Phase 1 (Real-time)
Feature being Tested	Realtime phase of summary generation
Description	From the input video, the tubes are extracted, static background is generated, objects are detected and presented to the user in anticipation of phase 2
Sample Input	Input video
Expected Output	Extracted tubes, static background and tags
Actual Output	As expected
Remarks	Test case passed successfully

7.4.2 System testing of query phase

Table 7.13 shows the system testing for the query phase. Here all the modules in the query phase are combined and tested. The system should automatically generate a summary from the list of objects and the input query given. The query phase performed successfully.

Table 7.13: System Testing of Query Module

Sl. No	**2**
Name of the Test case	Phase 2 (Query)
Feature being Tested	Query phase of summary generation
Description	The user query with required tags is fed in, using which the summary video is generated
Sample Input	User query
Expected Output	Summary video with clips of only selected tags
Actual Output	As expected
Remarks	Test case passed successfully

7.5 Summary

This chapter includes the general testing process, which starts with unit testing of the main modules followed by integration testing wherein the sub-modules and modules are merged together. System testing where the entire system is tested for its functionality and correctness was performed. The tests proved successful in most test cases and abnormal behavior was not traced in any of the modules.

Chapter 8

Experimental Results and Analysis of CCTVVS

In analysis of a process, experiments are commonly used to evaluate the inputs to the process which most times define the output of the process. They also help pinpoint the appropriate and thus target inputs to achieve a desired result. The output of the video summariser is compared with the expected output to verify the correctness of the system. There are several metrics for comparison. Analyzing the experimental output is verifying whether the evaluation metrics are satisfied. This chapter discusses the performance characteristics of the system.

8.1 Evaluation Metric

Evaluation metrics are the criteria for testing different algorithms. The behavior of the algorithms or techniques can be determined using these metrics. In this project, there are not many concrete objective methods for testing the summary, and some metrics are subjective as well. The outputs that are obtained from the different inputs given to the summariser are compared with the expected output to check whether the metrics are satisfied.

8.2 Experimental Dataset

Sample video clips in different urban areas with sparse motion and activity were collected, with only 1-2 events occurring once every few seconds. These clips were used for developing and testing the algorithm.

Figure 8.1 shows some frames from one of our experimental videos. Only 1-2 events are seen occurring at any point of time. People and bikes are the objects that are seen.

Figure 8.2 shows an output frame generated by our video summariser. Many people

Figure 8.1: Selection of input frames from one of our experimental videos

and bikes are seen simultaneously, and the timestamp shows the time at which the event occurred in the original input video. It is apparent from this picture that the density of events is dramatically improved.

Figure 8.2: A frame from the generated summary video

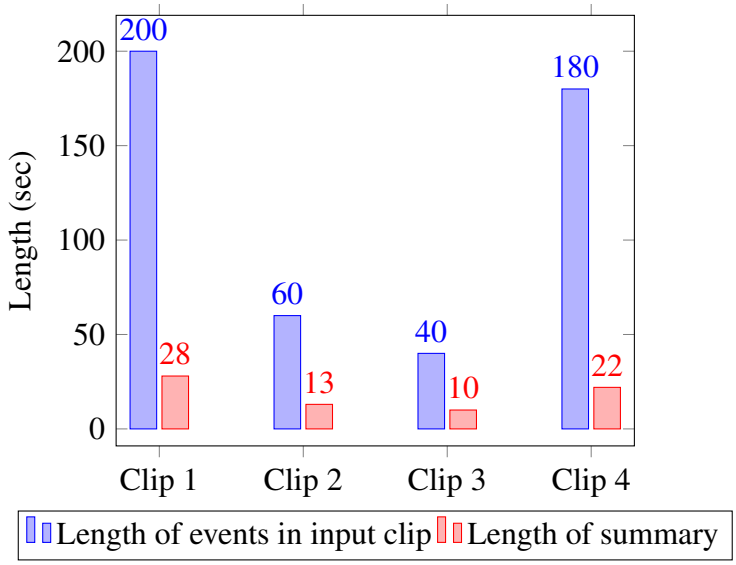
8.3 Performance Analysis

This section explains the experimental results of this project. The system performance was satisfactory and was mostly successful with the tested input clips, with some limitations which will be mentioned in the further sections.

8.3.1 Objective Evaluation Parameters

- **Exhaustiveness of summary** The summariser must retain all the events that occurred or the events of the specified type in the tag-based summary generation.
  Result: All significant events are selected in the events

- **Compression factor** Given by Length of original input/Length of summary. The compression factor must be as high as possible while keeping the overlap factor to a minimum.
  Result: There is a compression factor of 4-7x for our sample videos



- **Overlap factor** A number between 0 and 1 which indicates the amount of overlapping (intersection) between the tubes in the summary video. 0 indicates no overlap and 1 indicates complete overlap of every clip.
  Result: All videos have overlap lower than 0.1

8.3.2 Subjective Evaluation Parameters

- **Semantic structure of events** Interacting events (that occur in the same time and space) must be shown together.
  Result: All the interacting events are always shown together

- **Realistic appearance** The events must be extracted and be blended into the background seamlessly and appear realistic.
  Result: Realistic appearance in most cases, with some halo around object when at the edges of the frame, or when there is intersection.

Performance of Simulated Annealing

The simulated annealing optimisation process is not very effective when there are a large number of tubes to be re-arranged. The process is sensitive to the initial configuration and in some edge cases an optimal solution is not obtained. This causes the events in the summary to be overlapped. Further improvement in the generation of new configurations and an improved cost function will solve the issues faced.

8.4 Summary

This video summary system which has been developed performs in real-time (30fps) for the real-time phase and runs reasonably fast in the query phase as well. It takes about 1 minute for generating a summary video of 10-15 secs. The outputs are acceptable for the regular summarization (with all events) and the tag-based summarization, except a few cases where small or fast-moving objects are not detected. The optimization and blending process can be improved further to ensure that overlap is further reduced.

Chapter 9

Conclusion to CCTVVS

This project aimed to use a novel method to generate video summaries to reduce the amount of time spent in analyzing CCTV video footage. Our implementation of summarization by temporal rearrangement of events improves on other methods of just detecting frames of motion. The tag-based summary selects only the required type of event, reducing the summary length further.

A proof-of-concept has been presented, and it can be extended to be used in commercial CCTV systems with further development.

9.1 Limitations of the Project

While the current system works well for small clips, there are limitations which prevent it from being used in CCTV system:

- Optimization algorithm doesn't work very well when there are many events
- Blending algorithm doesn't make the summary look perfectly natural
- Further performance improvements required for optimization phase, to make it run faster

9.2 Future Enhancement

To overcome the limitations of the project, the following will be taken up in the future:

- Improve performance by parallelising and optimizing the code
- Experimenting and improving the optimization algorithm for better results with lesser overlap
- A simple UI for selecting the input, and setting the parameters for the summary is required, with links to go back to the original video

9.3 Summary

This chapter gives an overlook of the entire video summary system and briefly states the limitation of the project and future enhancements to grow and overcome the limitations of the same.

Bibliography

[1] A. Rav-Acha, Y. Pritch, and S. Peleg, "Making a long video short: Dynamic video synopsis," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, pp. 435–441, IEEE, 2006.

[2] Y. Pritch, A. Rav-Acha, and S. Peleg, "Nonchronological video synopsis and indexing," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 11, pp. 1971–1984, 2008.

[3] M. Rodriguez, "Cram: Compact representation of actions in movies," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3328–3335, IEEE, 2010.

[4] Y. Pritch, S. Ratovitch, A. Hendel, and S. Peleg, "Clustered synopsis of surveillance video," in *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 195–200, IEEE, 2009.

[5] S. V. Porter, M. Mirmehdi, and B. T. Thomas, "A shortest path representation for video summarisation," in *12th International Conference on Image Analysis and Processing, 2003. Proceedings.*, pp. 460–465, IEEE, 2003.

[6] Z. Zivkovic *et al.*, "Improved adaptive gaussian mixture model for background subtraction.," in *ICPR (2)*, pp. 28–31, Citeseer, 2004.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[8] M. A. Smith and T. Kanade, "Video skimming and characterization through the combination of image and language understanding," in *Proceedings 1998 IEEE International Workshop on Content-Based Access of Image and Video Database*, pp. 61–70, IEEE, 1998.

[9] N. Petrovic, N. Jojic, and T. S. Huang, "Adaptive video fast forward," *Multimedia Tools and Applications*, vol. 26, no. 3, pp. 327–344, 2005.

[10] O. Boiman and M. Irani, "Detecting irregularities in images and in video," *International journal of computer vision*, vol. 74, no. 1, pp. 17–31, 2007.

[11] J. H. Oh, Q. Wen, S. Hwang, and J. Lee, "Video abstraction," in *Video data management and information retrieval*, pp. 321–346, IGI Global, 2005.

[12] J. Nam and A. H. Tewfik, "Video abstract of video," in *1999 IEEE Third Workshop on Multimedia Signal Processing (Cat. No. 99TH8451)*, pp. 117–122, IEEE, 1999.

[13] Y. Li, T. Zhang, and D. Tretter, "An overview of video abstraction techniques," tech. rep., Technical Report HPL-2001-191, HP Laboratory, 2001.

[14] W.-S. Chu, Y. Song, and A. Jaimes, "Video co-summarization: Video summarization by visual co-occurrence," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[15] Y. Pritch, A. Rav-Acha, A. Gutman, and S. Peleg, "Webcam synopsis: Peeking around the world," in *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, IEEE, 2007.

[16] P. L. M. Bouttefroy, A. Bouzerdoum, S. L. Phung, and A. Beghdadi, "On the analysis of background subtraction techniques using gaussian mixture models," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4042–4045, IEEE, 2010.

[17] J. Sun, W. Zhang, X. Tang, and H.-Y. Shum, "Background cut," in *European Conference on Computer Vision*, pp. 628–641, Springer, 2006.

[18] X. Yao, "A new simulated annealing algorithm," *International Journal of Computer Mathematics*, vol. 56, no. 3-4, pp. 161–168, 1995.

[19] R. Szeliski, M. Uyttendaele, and D. Steedly, "Fast poisson blending using multi-splines," in *2011 IEEE International Conference on Computational Photography (ICCP)*, pp. 1–8, IEEE, 2011.

[20] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," in *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pp. 313–318, 2003.