VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", BELGAUM – 590014

A Project Report on

"Abnormal Event Detection"

Submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Engineering in
Information Science & Engineering

Submitted by:

AKSHAY ADIGA	**1PI12IS009**
MAHESH S. K.	**1PI12IS050**
PRAJWAL P. VASISHT	**1PI12IS073**

Under the guidance of

Internal Guide
Dr. Shylaja S. S.
Professor and Head of the Department,
Department of Information Science and Engineering,
PESIT

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

PES INSTITUTE OF TECHNOLOGY

100 Feet Ring Road, BSK 3rd Stage, Bengaluru – 560085

January 2016 – May 2016

PES INSTITUTE OF TECHNOLOGY
100 Feet Ring Road, B S K 3rd Stage
Bengaluru-560085

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled **“Abnormal Event Detection”** carried out by **Akshay Adiga**, bearing USN **1PI12IS009**, **Mahesh S. K.**, bearing USN **1PI12IS050, Prajwal P. Vasisht**, bearing USN **1PI12IS073**, are bonafide students of **PES INSTITUTE OF TECHNOLOGY**, Bangalore, an autonomous institute, under VTU, in partial fulfillment for the award of degree of **BACHELOR OF ENGINEERING IN INFORMATION SCIENCE & ENGINEERING of Visvesvaraya Technological University, Belgaum** during the year **2016**. 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 work prescribed for the above said degree.

Dr. Shylaja S. S Professor and Head Department of ISE, PESIT	**Dr. Shylaja S. S.** Professor and Head, Department of ISE PESIT	**Dr. K. S. Sridhar** Principal PESIT

External Viva

Name of the Examiners	**Signature with date**
1.____________________	____________________
2.____________________	____________________

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

P.E.S. Institute of Technology

Department of Information Science and Engineering

Bengaluru – 560085

DECLARATION

We, **Akshay Adiga, Mahesh S K, Prajwal P Vasisht**, students of Eighth Semester B.E., in the Department of Information Science and Engineering, **P.E.S. Institute of Technology, Bangalore** declare that the project entitled **"Abnormal Event Detection"** has been carried out by us and submitted in partial fulfillment of the course requirements for the award of degree of **Bachelor of Engineering** in **Information Science and Engineering of Visvesvaraya Technological University, Belagavi** during the academic year **2015-16**. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

Akshay Adiga - 1PI12IS009

Mahesh S K - 1PI12IS050

Prajwal P Vasisht - 1PI12IS073

ACKNOWLEDGEMENT

We would like to express my gratitude to everyone who has been a part of this unforgettable adventure. When we look back from where our journey began, we recollect all the people who brought this project to completion.

First and foremost, we would like to thank our Principal, **Dr. K N Balasubramanya Murthy** for providing us with the environment and facilities to carry out the project successfully.

We express our great appreciation to our HOD and project guide, **Dr. Shylaja S S** for her guidance, support and valuable inputs along with necessary resources which helped in completion of the project.

We express our gratitude to our project coordinator **Dr. Mamatha H R** for guiding our throughout the course of the project and **Prof. Vathsala M K** for her help in making the project report.

We would also like to thank our fellow student, **Akshay Varun** of the Department of ECE for taking time out and providing valuable input about the project which helped us see the project to its completion.

We also express our sincere thanks to all the teaching and non-teaching staff of the **Department of Information Science and Engineering** for their kind support and ever helping nature.

We would also like to thank our parents and other family members for their continuous support, without which, we would not have been able to achieve whatever we have. We would like to thank all our friends who have directly or indirectly helped me throughout the duration of the project.

ABSTRACT

Abnormal event detection is a growing demand to process a plethora of surveillance videos. Our project helps detect the abnormality in videos with high accuracy, thus saving time for organizations and individuals who would have to go through the entire footage instead.

The algorithm exploits inherent redundancy of videos and constructs a sparse combination learning matrix to identify possible abnormal events. The method implemented in the project provides an optimized and low computational solution to a highly complex problem.

The Usability of our project is high as it produces a decent performance on even ordinary desktop PCs using Python 2.7 and thus can be utilized by everyone with a basic computer system setup.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

Computerized abnormal event detection is a growing need due to its ability to be automated and replace human interference, hence innately inducing a sense of reliability and security. Its flexibility is portrayed in the fact that it need not completely replace manual efforts but can act as a tool to speed-up the process of detection. These benefits coupled with the fact that it can be run on ordinary desktop PCs with high accuracy makes it a highly usable and powerful. The task of detecting abnormal events based on what cameras capture is critical and traditionally labor-intensive and laborious as abnormal events happen with a very small chance, making over 99% of the effort to watch the video go in vain.

Surveillance cameras are very common across various industries throughout the world. The applications of these cameras can range from theft deterrence to weather monitoring and more. Parks, communities, and neighborhoods — all public spaces — should be outfitted with video surveillance systems to help deter crime and enhance public safety. Law enforcement can also view video directly from their smartphones, enabling quicker response times. Video surveillance can help enormously with crowd control as well as prevent crime by providing security staff with real-time images from an event. Zoom in on suspicious behavior before it becomes a problem with modern IP HD surveillance systems. Computerized abnormal event detection can be applied in these scenarios as such events are time sensitive and detecting them with very little delay and high accuracy is critical. Also computerized event detection reduces human prone error and with its high frame rates, has become indispensable.

Traditional image processing algorithms to detect abnormalities are heavy on computations and are thus slow and require very powerful systems to be run on. Sparsity based techniques convert these heavy computations into smaller costless least square optimizations which speeds up the process and provides faster detection rates.

The formal definition of the word “Abnormal” is “*deviating from what is normal or usual, typically in a way that is undesirable or worrying*”. **“Abnormal”**, here, is confined to the boundaries of the environment of where most surveillance is done or used, for instance – A man who suddenly begins to run in an environment where most people are standing still or are walking slowly, or a cyclist who suddenly enters the frame of a video which has only people walking or standing, or a case in which the camera’s vision is obstructed and at a later point regained is also considered “Abnormal” in our case. This confinement is done as the definition of the word “Abnormal” is very broad and it would be a herculean task to try to cover and capture all of the scenarios that the word encompasses.

CHAPTER-2

LITERATURE SURVEY

Abnormal event detection is a major topic of research due to which a lot of people have come up with different algorithm to detect abnormality. We do a detailed study and comparison of each of them and come up with better approach if possible to include with the sparsity learning method to achieve better accuracy.

In [24] author uses clustering-based approach for detecting abnormalities in surveillance video requires the appropriate definition of similarity between events. The HMM-based similarity defined previously falls short in handling the overfitting problem. Author propose in this paper a multi-sample-based similarity measure, where HMM training and distance measuring are based on multiple samples. These multiple training data are acquired by a novel dynamic hierarchical clustering (DHC) method. By iteratively reclassifying and retraining the data groups at different clustering levels, the initial training and clustering errors due to overfitting will be sequentially corrected in later steps. Experimental results on real surveillance video show an improvement of the proposed method over a baseline method that uses single-sample-based similarity measure and spectral clustering.

A surveillance system that supports a human operator by automatically detecting abandoned objects and drawing the operator's attention to such events as addressed in [25]. It consists of three major parts: foreground segmentation based on Gaussian Mixture Models, a tracker based on blob association and a blob-based object classification system to identify abandoned objects. For foreground segmentation, we assume that video sequences of the background shot under different natural settings are available a priori. The tracker uses a single-camera view and it does not differentiate between people and luggage. The classification is done using the shape of detected objects and temporal tracking results, to successfully categorize objects into bag and non-bag (human). If a potentially abandoned object is detected, the operator is notified and the system provides the appropriate key frames for interpreting the incident.

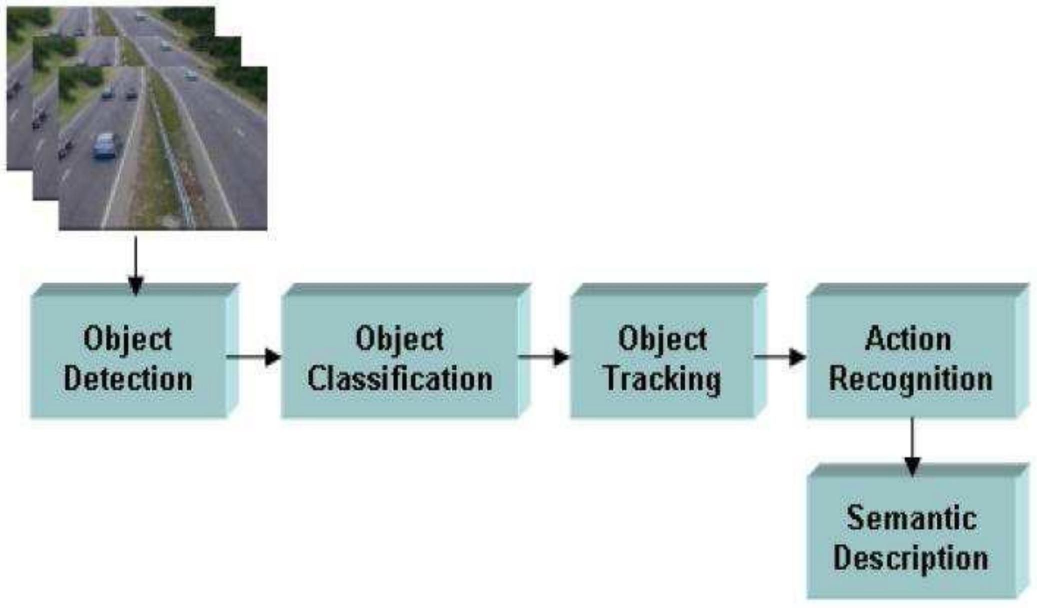


Figure 2.1 A generic framework for smart video processing algorithms

Detection of abnormal events via a sparse reconstruction over the normal bases is proposed in [26]. Given an over-complete normal basis set (e.g., an image sequence or a collection of local spatio-temporal patches), they introduce the sparse reconstruction cost (SRC) over the normal dictionary to measure the normalness of the testing sample. To condense the size of the dictionary, a novel dictionary selection method is designed with sparsity consistency constraint. By introducing the prior weight of each basis during sparse reconstruction, the proposed SRC is more robust compared to other outlier detection criteria. Their method provides a unified solution to detect both local abnormal events (LAE) and global abnormal events (GAE). Author further extend it to support online abnormal event detection by updating the dictionary incrementally. Experiments on three benchmark datasets and the comparison to the state-of-the-art methods validate the advantages of our algorithm.

Human activities taking place in an outdoor surveillance environment can be effectively detected using method described in [27]. Human tracks are provided in real time by the baseline video surveillance system. Given trajectory information, the event analysis module will attempt to determine whether or not a suspicious activity is currently being observed. However, due to real-time processing constrains, there might be false alarms generated by video image noise or non-human objects. It requires further intensive examination to filter out false event detections which can be processed in an off-line fashion. Author propose a hierarchical abnormal event detection system that takes care of real time and semi-real time as multi-tasking. In low level task, a trajectory-based method processes trajectory data and detects abnormal events in real time. In high level task, an intensive video analysis algorithm checks whether the detected abnormal event is triggered by actual humans or not.

In [28], author address the problem of unusual-event detection in a video sequence. Invariant subspace analysis (ISA) is used to extract features from the video, and the time-evolving properties of these features are modeled via an infinite hidden Markov model (iHMM). The iHMM retains a full posterior density function on all model parameters, including the number of underlying HMM states. Anomalies (unusual events) are detected subsequently if a low likelihood is observed when associated sequential features are submitted to the trained iHMM. A hierarchical Dirichlet process framework is employed in the formulation of the iHMM. The evaluation of posterior distributions for the iHMM is achieved in two ways: via Markov chain Monte Carlo and using a variational Bayes formulation. Comparisons are made to modeling based on conventional maximum-likelihood-based HMMs, as well as to Dirichlet-process-based Gaussian-mixture models.

Automatic technique for detection of abnormal events in crowds is presented in [29]. Crowd behavior is difficult to predict and might not be easily semantically translated. Moreover it is difficult to track individuals in the crowd using state of the art tracking algorithms. Therefore author characterize crowd behavior by observing the crowd optical flow and use unsupervised feature extraction to encode normal crowd behavior. The unsupervised feature extraction applies spectral clustering to find the optimal number of models to represent normal motion patterns. The motion models are HMMs to cope with the variable number of motion samples that might be present in each observation window. The results on simulated crowds demonstrate the effectiveness of the approach for detecting crowd emergency scenarios.

The proposed method to detect unusual event in various video scenes in [30] finds out the video clips that are most different from the others based on the similarity measure. Each video clip is represented by the motion magnitude and direction histograms and color histogram. Without searching key-frames, author calculate the similarity matrix by using chi^2 difference or chamfer difference as the similarity measure of features in different clips. Finally, author apply n-cut clustering. Clusters with low self-similarity value are reported as unusual events.

Sparsity-based methods that have been recently applied to abnormal event detection, and have achieved impressive results as discussed in [31]. However, most such methods fail to consider the relationship among coefficient vectors; furthermore, they neglect the underlying "dictionary structure."' The authors' compact low-rank sparse representation (CLSR) method overcomes these drawbacks. Specifically, it adds compact regularization to the sparse representation model, which explicitly considers the relationship among coefficient vectors. The authors utilize the low-rank property to capture the underlying dictionary structure. Their method is verified on Ffrethree challenging databases, and the experimental results demonstrate that it compares favorably to state-of-the-art methods in abnormal event detection.

Simple and effective crowd behavior normality method is proposed in [32]. We use the histogram of oriented social force (HOSF) as the feature vector to encode the observed events of a surveillance video. A dictionary of code words is trained to include typical HOSFs. To detect whether an event is normal is accomplished by comparing how similar to the closest code word via z-value. The proposed method includes the following characteristic: (1) the training is automatic without human labeling; (2) instead of object tracking, the method integrates particles and social force as feature descriptors; (3) z-score is used in measuring the normality of events. The method is testified by the UMN dataset with promising results.

A new event detection technique for detecting abnormal events in traffic video surveillance is put forward in [33]. The main objective of this work is to detect the abnormal events which normally occur at junction, in video surveillance. Our work consists of two phases 1) Training Phase 2) Testing Phase. Their main novelty in this work is modified lossy counting algorithm based on set approach. Initially, the frames are divided into grid regions at the junction and labels are assigned. The proposed work consist of blob detection and tracking, conversion of object location to data streams, frequent item set mining and pattern matching. In the training phase, blob detection is carried out by separating the modelled static background frame using Gaussian mixture models (GMM) and this will be carried out for every frame for tracking purpose. The blobs location is determined by assigning to the corresponding grid label and numbered moving object direction to form data streams. A modified lossy counting algorithm is performed over temporal data steams for discovering regular spatial video patterns. In testing phase, the same process is repeated except frequent item set mining, for finding the spatial pattern in each frame and it is compared with stored regular video patterns for abnormal event detection. The proposed system has shown significant improvement in performance over to the existing techniques.

2.1 Open CV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 7 million. The library is used extensively in companies, research groups and by governmental bodies.

It has C++, C and Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

OpenCV's application areas include 2D and 3D feature toolkits, egomotion estimation, facial recognition system, gesture recognition, human–computer interaction (HCI), mobile robotics, motion understanding, object identification, segmentation and recognition, stereopsis stereo vision: depth perception from 2 cameras, structure from motion (SFM), motion tracking and augmented reality.

To support some of the above areas, OpenCV includes a statistical machine learning library that contains boosting, decision tree learning, gradient boosting trees, expectation-maximization algorithm, k-nearest neighbor algorithm, naive Bayes classifier, artificial neural networks, random forest and support vector machine (SVM).

CHAPTER-3

SYSTEM REQUIREMENT SPECIFICATION

3.1 Hardware Requirement

A desktop with 3.4GHz CPU, 8G memory, 32-bit or 64-bit processor, minimum 8-bit graphic adapter and running windows operation system. Basic hardware like monitor, mouse and keyboard are a must for input and output. A CD-ROM or a USB port to install the nessary soft wares. An additional camera will be required for real-time video processing.

3.2 Software Requirement

Python with the version 2.7.x with the following modules installed, opencv, numpy, os, time, fmatch, pickle. Windows operation system. Camera modules installed for importing video file from camera.

3.3 Non-Functional Requirements

3.3.1 Usability

The system is easy to train and test thus navigates in the most expected way with less delay. Since the algorithm is written in such a way that a lot of parallel computation can be performed, extensively high frame rates can be achieved by inculcating multi-threading. User will be allowed to add his frames easily and direct testing can be started on those frames. Thus it is user friendly, reducing complexity on users and getting them a better result in a faster way.

3.3.2 Assumption

Test frames are either in .tif, .jpg, .png, .bmp formats. Each frame is taken to represent one second of the video. So a 200 second video is assumed to generate 200 frames. The learning rate for training is chosen to be 0.0001 and the acceptance value of error in training is set to 0.04. In the testing phase threshold for reconstruction error is assumed to be 0.00000045 for optimal accuracy during testing.

3.3.3 Performance

Pre-captured video frames will be analyzed fast as a significant amount of time is saved in not capturing the video. On the other hand real-time capturing will extend the required time. Detection of abnormality in the frames is done at a rate of 150fps using parallel processing of data in the testing phase. An accuracy of 90% is achieved with the sparse combination method implemented.

3.3.4 Reliability

The software is tested with varies dataset mentioned in this report and the output is very close to the actual abnormal scenarios thus turning out to be reliable when all the assumptions mentions are considered.

3.4 Functional Requirements

The system should process images in the chosen image formats. (jpg, png, bmp, tif) The system shall detect abnormal frames and display the exact time from the start at which the abnormal event is detected. The length of the test video can vary and no particular limit is imposed. In the presence of no abnormality, user should be displayed with a message that the video is normal. After the whole video is processed the user should be given an option to see the portion of the video where abnormal event was detected. Only the frames to be suspected as abnormal should be played as a video. Option to see the abnormal portion of the video any number of times should be given to the user.

CHAPTER-4

SYSTEM DESIGN

4.1 Block Diagram

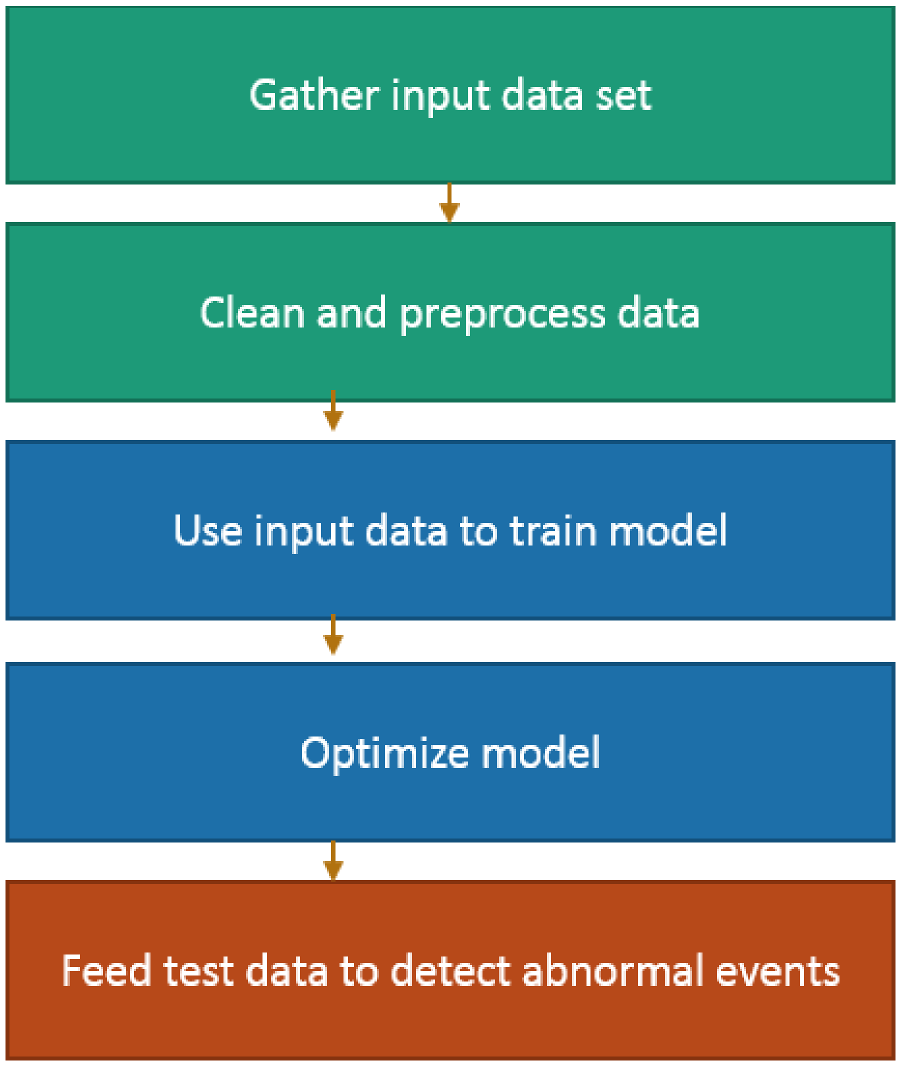

Gather input data set
Clean and preprocess data
Use input data to train model
Optimize model
Feed test data to detect abnormal events

Figure 4.1 Block Diagram of the overall process

4.2 Implemented Modules

4.2.1 User Module

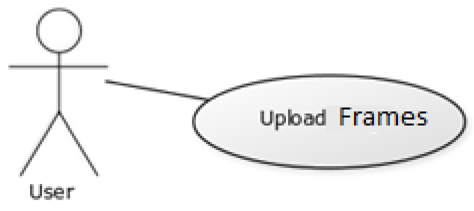


Figure 4.2 User Module

Table 4.1 User Description

User Case	Description
Actor	User
Pre-condition	Input or Upload frames to the frames location in system
Main Scenario	User uploads video frames
Post Condition	Frames successfully uploaded

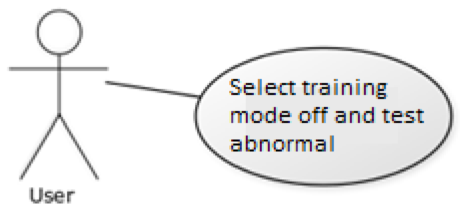


Figure 4.3 User Module (Start Program Phase)

Table 4.2 User Description (Start Program Phase)

User Case	Description
Actor	User
Pre-condition	Training method should be selected to start program
Main Scenario	User selects none to skip training and go to testing
Post Condition	Training skipped and Testing process started

4.2.2 Pre-Process Module

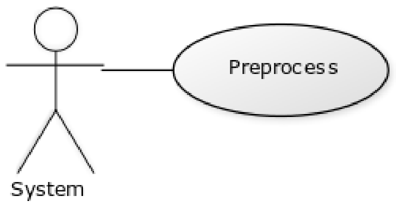


Figure 4.4 Pre-process Module

Table 4.3 Pre-process Description

User Case	Description
Actor	System
Pre-condition	Frames in grayscale format
Main Scenario	Frames undergoes a series of pre-processing to form a 3-D matrix called cubes.
Post Condition	208 Cubes for each 5 consecutive frames

4.2.3 Feature Extraction Module

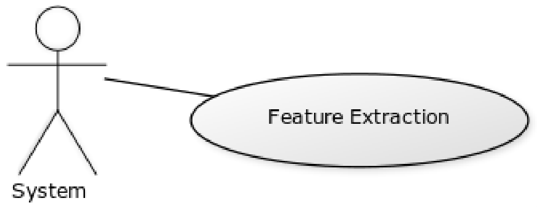


Figure 4.5 Feature Extraction Module

Table 4.4 Feature Extraction Description

User Case	Description
Actor	System
Pre-condition	Cubes (3-D matrix)
Main Scenario	3-D gradient of the cube is calculated for each value and concatenated to form cube
Post Condition	A set of features are generated for each cube

4.2.4 Training Module

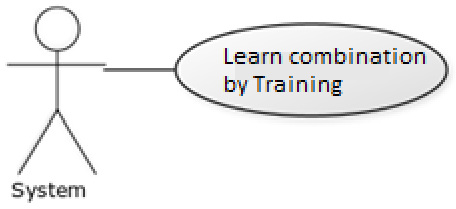


Figure 4.6 Training Module

Table 4.5 Training Description

User Case	Description
Actor	System
Pre-condition	Features
Main Scenario	Learn a set of combination from the features using sparsity based learning
Post Condition	A set of combinations learnt

4.2.5 Testing Module

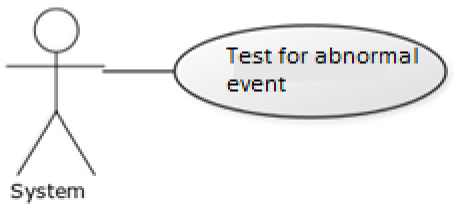


Figure 4.7 Testing Module

Table 4.6 Testing Description

User Case	Description
Actor	System
Pre-condition	Learn Combination Set
Main Scenario	Use the combination to reconstruct test frames and check the error for abnormality.
Post Condition	Output the time at which abnormality is detected

4.2.6 Result Display Module

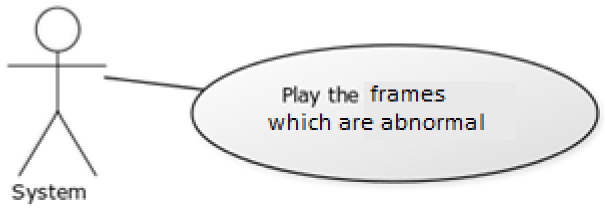


Figure 4.8 Result Display Module

Table 4.7 Result Display Description

User Case	Description
Actor	System
Pre-condition	Time and frames at which abnormality is found
Main Scenario	Respective frames are loaded and displayed to the user as a video
Post Condition	Portion of video having abnormal event

4.3 Data Flow diagram

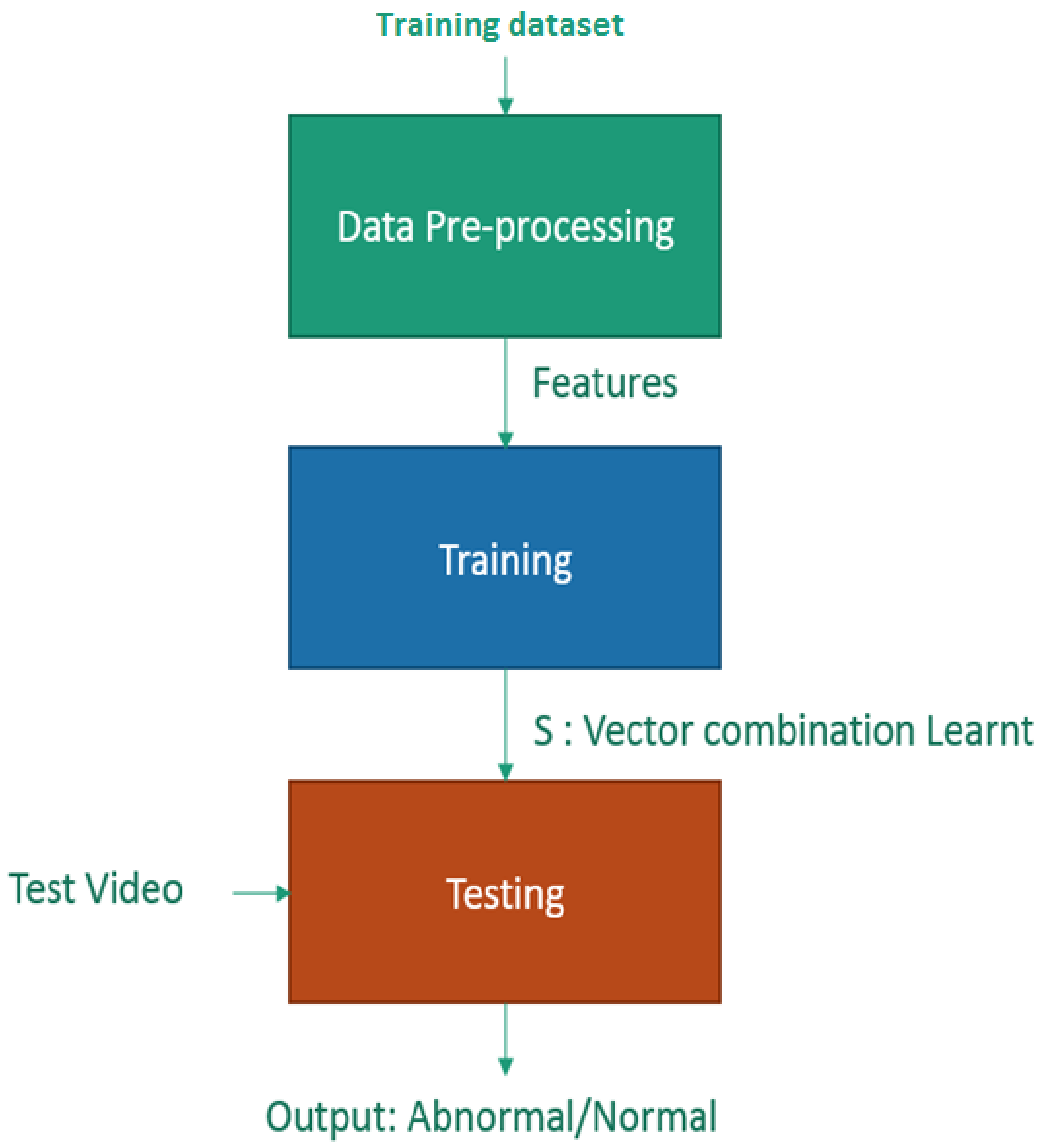


Figure 4.9 Data Flow diagram

CHAPTER-5

IMPLEMENTATION

5.1 Method

Extract usable data by resizing each frame into different scales as [5] and uniformly partition each layer to a set of non-overlapping patches. All patches have the same size. Corresponding regions in 5 continuous frames are stacked together to form a spatial-temporal cube. This pyramid involves local information in fine-scale layers and more global structures in small-resolution ones. With the spatial-temporal cubes, compute 3D gradient features on each of them following [11]. These features in a video sequence are processed separately according to their spatial coordinates. Only features at the same spatial location in the video frames are used together for training and testing.

The goal is to find a sparse basis combination set S = {S1,..., SK} with each Si $\in R^{p\times s}$ containing s dictionary basis vectors, forming a unique combination, where s<<q. Each Si belongs to a closed, convex and bounded set, which ensures column-wise unit
Norm to prevent over-fitting.

Sparse combination learning has two goals. The first goal – effective representation – is to find K basis combinations, which enjoy a small reconstruction error t.The second goal is to make the total number K of combinations small enough based on redundant surveillance video information.

5.2 Pre-Processing

The pre-processing step includes importing the video frames and making it reading for training. It also involves feature extraction which is the input to the training algorithm. Initially the video is converted to frames. Each video will generate a set of frames which approximately denotes the number of seconds. This process of converting video to frame using python libraries is illustrated in Figure 5.1.

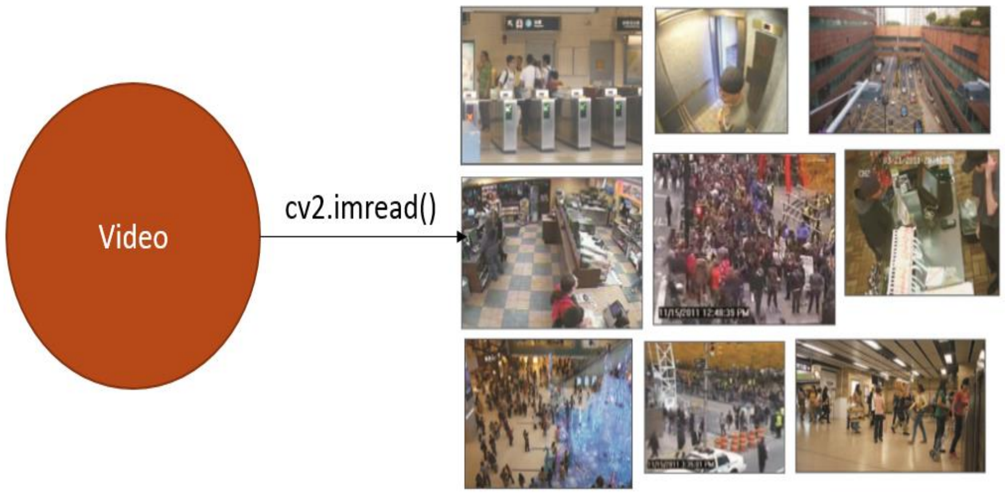


Figure 5.1 Converting Video to Frames

The frames obtained are then analyzed in steps of 5 frames at a time basis. So 5 consecutive frames are taken at a time and each of the frames are resized to different levels of resolution. From experimental result 3 scales are chosen, one with 20x20 resolution, 30x40 resolution and 120x160 resolution. This form an image pyramid as shown in Figure 5.2, which hold lower resolution images at the top and as we go down the pyramid higher resolution images will be encountered.

This process of generating image pyramid by resizing images to different scales helps in analyzing more content at each pixel level. In the top of the pyramid each pixel will hold a bigger portion of the image compared to the lower levels of the pyramid which would hold a zoomed in version of the smaller image covering a smaller portion. This helps the algorithm to detect both local and global abnormal event at each region of the image.

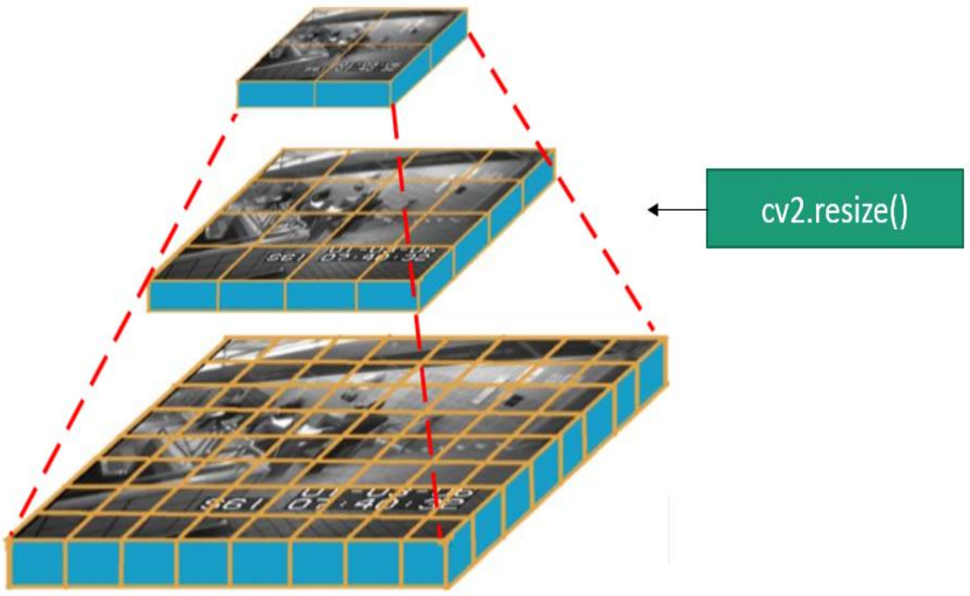
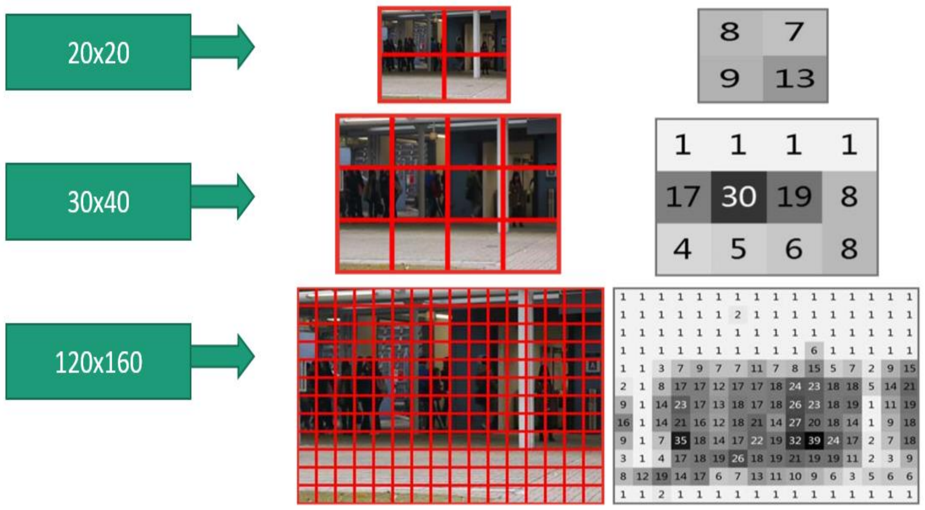


Figure 5.2 Image Pyramid

At each scaled image, the image is divided into non-overlapping region of size 10x10 pixels called patch. So based on the resolution different number of patches will be obtained. An illustration of dividing the images to patches is shown in Figure 5.3. The whole process is repeated for all the 5 frames. With the chosen scales a total of 208 patches for each frame are generated like Figure 5.3, will be used further.



8	7
9	13

1	1	1	1
17	30	19	8
4	5	6	8

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	6	1	1	1	1	1
1	1	3	7	9	7	7	11	7	8	15	5	7	2	9	15
2	1	8	17	17	12	17	17	18	24	23	18	18	5	14	21
9	1	14	23	17	13	18	17	18	26	23	18	19	1	11	19
16	1	14	21	16	12	18	21	14	27	20	18	14	1	9	18
9	1	7	35	18	14	17	22	19	32	39	24	17	2	7	18
3	1	4	17	18	19	26	18	19	21	19	19	11	2	3	9
8	12	19	14	17	6	7	13	11	10	9	6	3	5	6	6
1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 5.3 Scaling of images

Corresponding patches in 5 consecutive frames are collected and stacked as shown in Figure 5.4. So 208 patches will be stacked over 5 consecutive frames thus forming 208 cubes. Since each patch is 10x10. The stacked cube will be 10x10x5 3-D matrix. This cube thus represent a spatio-temporal values of frames in different parts of the frames. These cube will show change in pixel in x, y direction along with time direction.

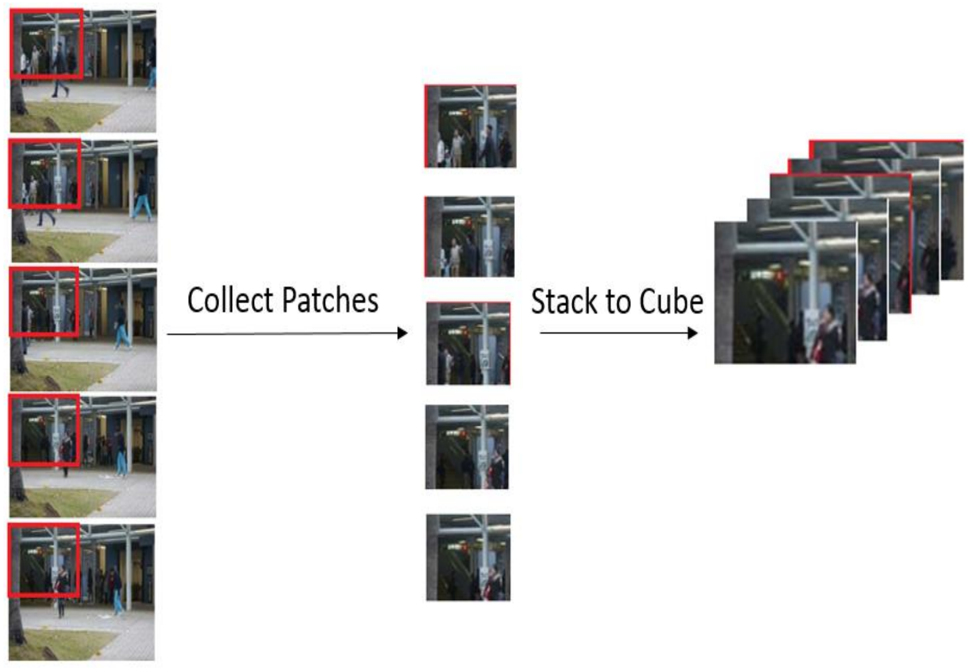


Figure 5.4 Patches to cube conversion

Cubes generated are individually converted to features. A 3-D gradient is applied on the 3-D matrix, which gives 3 values for each pixel location denoting change in horizontal, vertical and time dimension. For this sobel derivative shown in Figure 5.5 is used on all the cubes. Thus each of the 500 values will give 3 values in corresponding direction.

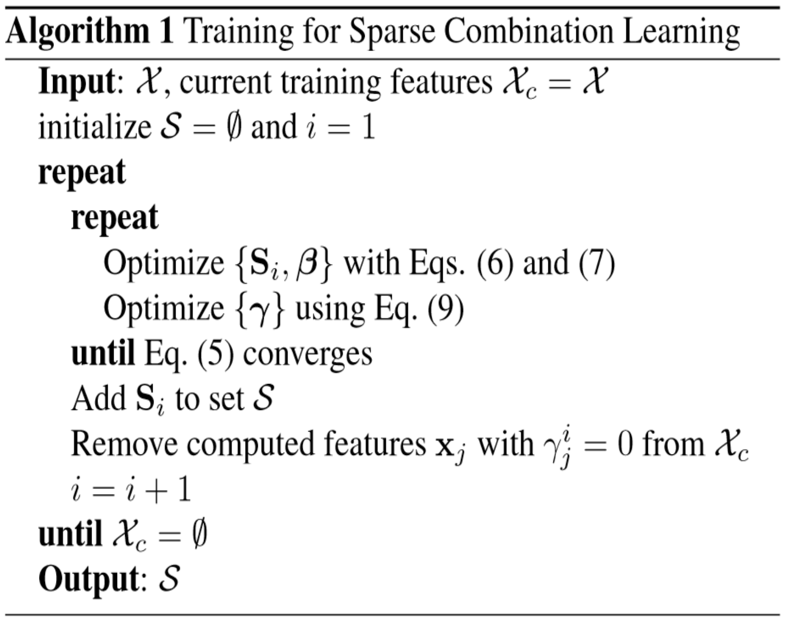
$$\nabla I_i = [I_{i,x}\ I_{i,y}\ I_{i,t}]^T = \left[\frac{\partial I}{\partial x}\ \frac{\partial I}{\partial y}\ \frac{\partial I}{\partial t}\right]^T$$

Figure 5.5 Sobel derivative

The 1500 values obtained from sobel derivative is concatenated to form the feature with a dimension of 1x1500. These vectors are obtained for each of the cubes. Together all of these form the feature set which will be fed to the training algorithm for further processing.

5.3 Training

The training algorithm uses a method called Sparse Combination Learning which will be used to learn a set of combination using the given input features. The pseudo-code for the algorithm is shown in Figure 5.6, it is followed as it is to obtain a set of combination denoted S which will be used in testing phase.



```
**Algorithm 1** Training for Sparse Combination Learning
  **Input**: $\mathcal{X}$, current training features $\mathcal{X}_c = \mathcal{X}$
  initialize $\mathcal{S} = \emptyset$ and $i = 1$
  **repeat**
    **repeat**
      Optimize $\{\mathbf{S}_i, \boldsymbol{\beta}\}$ with Eqs. (6) and (7)
      Optimize $\{\gamma\}$ using Eq. (9)
    **until** Eq. (5) converges
    Add $\mathbf{S}_i$ to set $\mathcal{S}$
    Remove computed features $\mathbf{x}_j$ with $\gamma_j^i = 0$ from $\mathcal{X}_c$
    $i = i + 1$
  **until** $\mathcal{X}_c = \emptyset$
  **Output**: $\mathcal{S}$
```

Figure 5.6 Training Algorithm

5.3.1 Learning Combination on Training Data

The features extracted in data pre-processing step is used as an initial input to the training algorithm which is denoted as Xc. These set of feature are used in the k-means algorithm with 10 centers to obtain the initial Si vector for the iteration as shown in Figure 5.7. The algorithm is run till all the features are used up to generate combinations. Once the feature set is empty the algorithm stops and the generated set of vectors S is used as the training output, the learnt combinations.

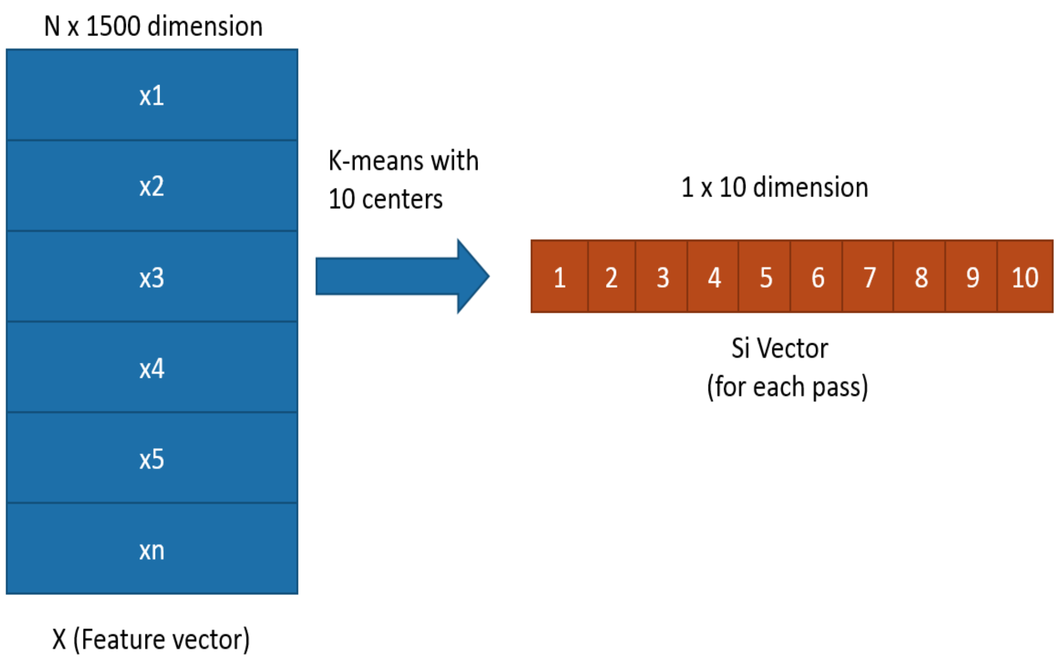


Figure 5.7 Initial Si vector generation

In each learning iteration value of Si, beta and gamma are updated. These updated values are used to calculate energy L. This energy decreases in each iteration leading to convergence. Whenever the energy L starts to increase the Learning iteration is stopped and the Si value at that point is used as the learn vector for that iteration. After each learning iteration the value of Xc is updated by removing all the vectors which were used up in learning the Si vector generated. This is done by inspecting the gamma value which will be either 0 or 1. A 0 at the end of learning iteration implies that the corresponding feature was used up in the learning the Si vector generated and a 1 in gamma at the end of learning iteration implies that the corresponding feature was not used up. At the end of learning iteration appropriate action on Xc is taken as shown in Figure 5.8. So all the feature with gamma value 0 will be removed and the remaining are used in the next iteration for next set of combination learning.

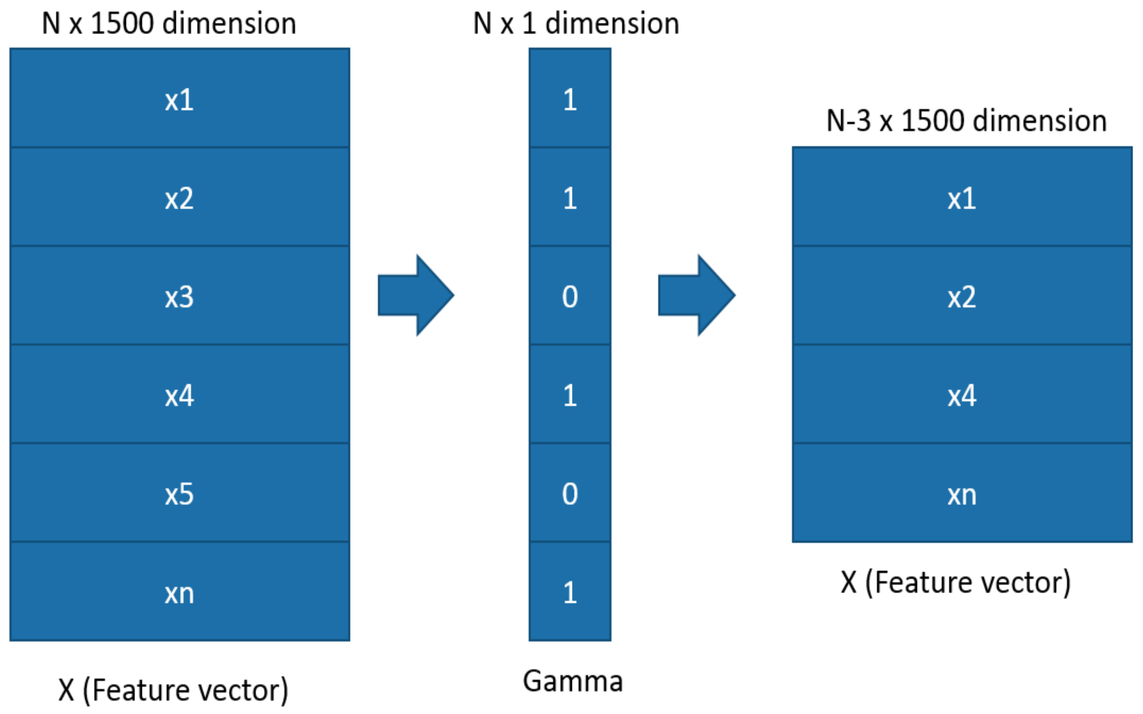


Figure 5.8 Feature updating at end of Learning Iteration

5.3.2 Optimization for Training

The learning iteration calculates Si, beta and Gamma value. Optimization should be performed on each of these calculation to reduce the time for convergence and to make sure that there is proper condition for convergence. At each step first Si value is calculated using the optimized equation for Si shown in Figure 5.9. This value of Si chosen such that a convergence point will be reached and the Si matrix won't reach a point where it becomes a singular matrix, in which case further proceeding with the algorithm will be troublesome.

$$\mathbf{S}_i = \prod[\mathbf{S}_i - \delta_t \nabla_{\mathbf{S}_i} L(\boldsymbol{\beta}, \mathbf{S}_i)]$$

Figure 5.9 Si updating equation

Next the value of beta and gamma are calculated respectively using the equations shown in Figure 5.10 and Figure 5.11 respectively. These values are calculated using the new Si value calculated in the previous step. This is calculate for each feature. So each feature will have its own beta and gamma value.

$$\boldsymbol{\beta}_j^i = (\mathbf{S}_i^T \mathbf{S}_i)^{-1} \mathbf{S}_i^T \mathbf{x}_j .$$

Figure 5.10 Beta Equation

$$\gamma_j^i = \begin{cases} 1 & if \ \ \|\mathbf{x}_j - \mathbf{S}_i \boldsymbol{\beta}_j^i\|_2^2 < \lambda \\ 0 & \text{otherwise} \end{cases}$$

Figure 5.11 Gamma Equation

These new value of Si, beta and gamma will be used to calculate the energy L using equation shown in Figure 5.12. The difference between energy two consecutive iterations are calculated to check for change in energy. If the change in energy is positive, signifying that the energy is increasing, the learning iteration will be stopped.

$$L(\boldsymbol{\beta}, \mathbf{S}_i) = \sum_{j \in \Omega_c} \gamma_j^i \|\mathbf{x}_j - \mathbf{S}_i \boldsymbol{\beta}_j^i\|_2^2 .$$

Figure 5.12 L equation (Energy)

5.3.3 Algorithm Summary

In each pass, learn one Si Repeat this process to obtain a few combinations until the training data set Xc is empty. This scheme reduces information overlap between combinations.

The training algorithm is summarized in Figure 5.6. The initial dictionary Si in each pass is calculated by clustering training data Xc via K-means with s centers. The algorithm is controlled by λ, the upper bound of reconstruction errors. Reducing it could lead to a larger K. The approach is expressive because all training normal event patterns are represented with controllable reconstruction errors under condition

5.4 Testing

5.4.1 Overall Testing Process

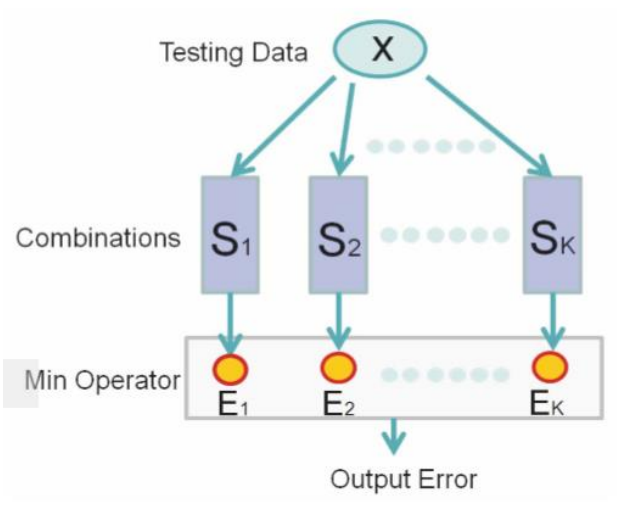


Figure 5.13 Overall Testing Process

5.4.2 Auxiliary Matrix Generation

With the learned sparse combinations S = {S1 ...SK}, in the testing phase with new data x, check if there exists a combination in S fitting the reconstruction error upper bound. It can be quickly achieved by checking the least square error for each Si as shown in Figure 5.14.It is a standard quadratic function with the optimal solution as shown in Figure 5.15.

$$\min_{\boldsymbol{\beta}^i} \|\mathbf{x} - \mathbf{S}_i \boldsymbol{\beta}^i\|_2^2 \quad \forall\, i = 1, \ldots, K$$

Figure 5.14 Equation 10

$$\widehat{\boldsymbol{\beta}^i} = (\mathbf{S}_i^T \mathbf{S}_i)^{-1} \mathbf{S}_i^T \mathbf{x}.$$

Figure 5.15 Equation 11

The reconstruction error in Si is

$$\|\mathbf{x} - \mathbf{S}_i \widehat{\boldsymbol{\beta}^i}\|_2^2 = \|(\mathbf{S}_i (\mathbf{S}_i^T \mathbf{S}_i)^{-1} \mathbf{S}_i^T - \mathbf{I}_p)\mathbf{x}\|_2^2,$$

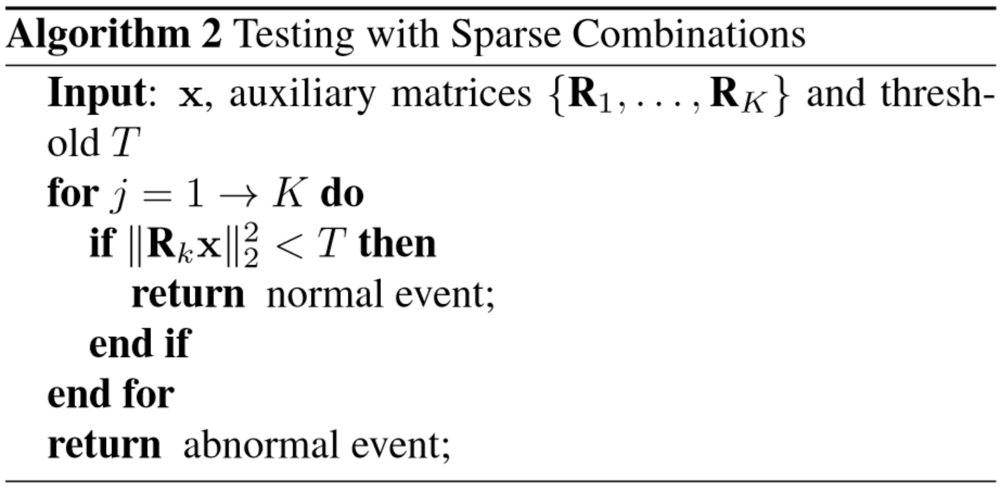
Figure 5.16 Equation 12

Where Ip is a p × p identity matrix. To further simplify computation, an auxiliary matrix Ri is defined for each Si using the equation shown in Figure 5.17

$$\mathbf{R}_i = \mathbf{S}_i (\mathbf{S}_i^T \mathbf{S}_i)^{-1} \mathbf{S}_i^T - \mathbf{I}_p$$

Figure 5.17 Auxiliary matrix equation

5.4.3 Testing Algorithm



```
Algorithm 2 Testing with Sparse Combinations
Input: $\mathbf{x}$, auxiliary matrices $\{\mathbf{R}_1, \ldots, \mathbf{R}_K\}$ and threshold $T$
for $j = 1 \rightarrow K$ do
    if $\|\mathbf{R}_k\mathbf{x}\|_2^2 < T$ then
        return normal event;
    end if
end for
return abnormal event;
```

Figure 5.18 Testing Algorithm

Reconstruction error for Si is accordingly norm of Ri.x . If it is small, x is regarded as a normal event pattern. The final testing scheme is summarized in Figure 5.18.

It is noted that the first a few dominating combinations represent the largest number of normal event features, which enables to determine positive data quickly. Also, the method can be easily accelerated via parallel processing to achieve O(1) complexity although it is generally not necessary.

CHAPTER-6

RESULTS AND DISCUSSIONS

6.1 UCSD Perl Dataset Benchmark

The UCSD Ped1 dataset [13] provides 34 short clips for training, and another 36 clips for testing. All testing clips have frame-level ground truth labels, and 10 clips have pixel-level ground truth labels. There are 200 frames in each clip. Our configuration is similar to that of [13]. That is, the performance is evaluated on frame- and pixel-levels. We show the results via running time comparisons across various methods as referenced.

6.1.1 Running time comparison on UCSD dataset

We compare the running time in Table 6.1. The detection time per frame and working platforms of [13, 5, 2] are obtained from the original papers.

Table 6.1 Running Time comparison on UCSD Dataset

		Second/Frame	Platform	CPU	Memory
[13]		25	-	3.0	2.0
[5]		3.8	-	2.6	2.0
[2]		5~10	MATLAB	-	-
Ours		0.6	Windows	2.0	4.0

6.2 Snapshots

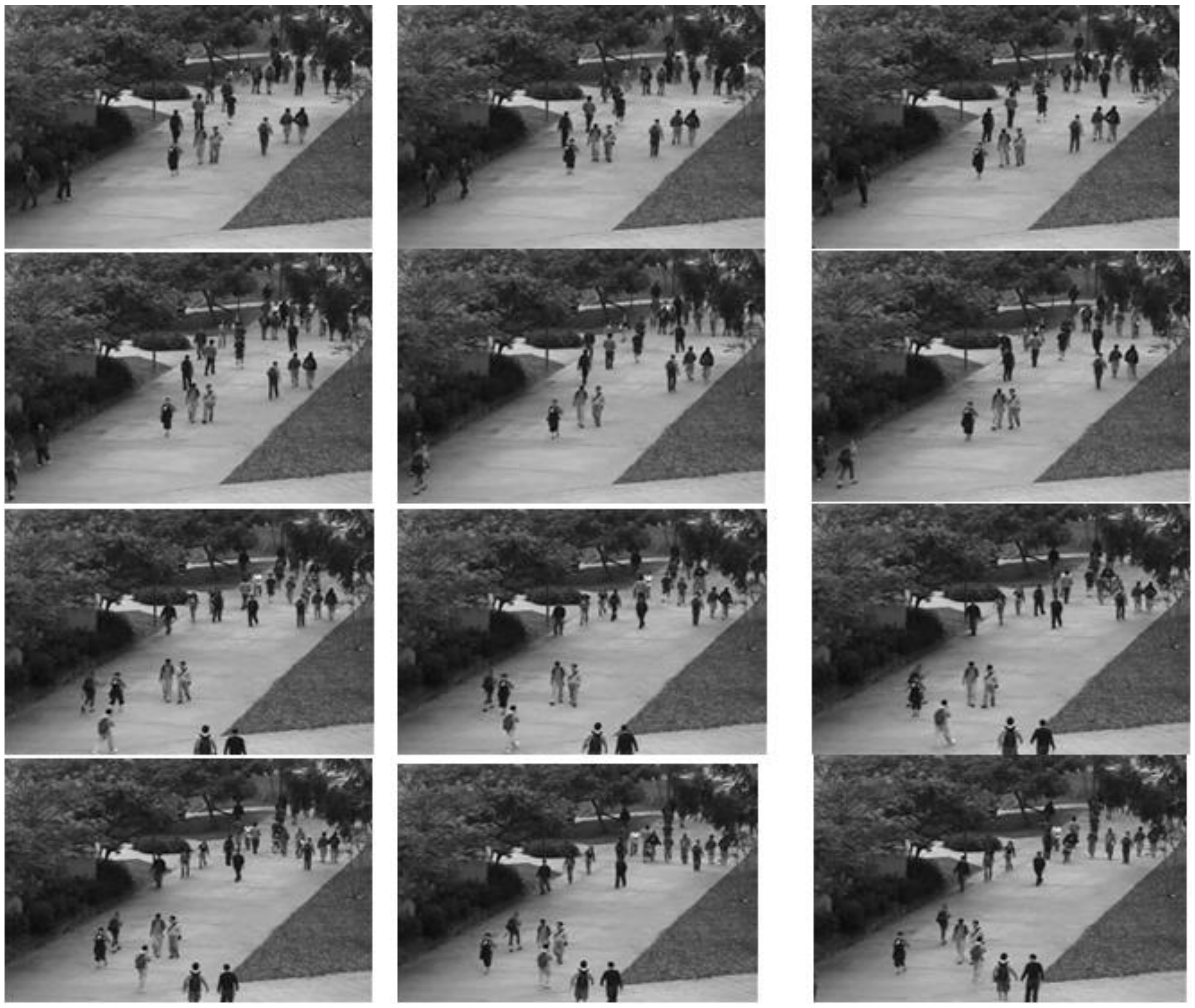


Figure 6.1 - Input test frames

Figure 6.2 – Expected output frames

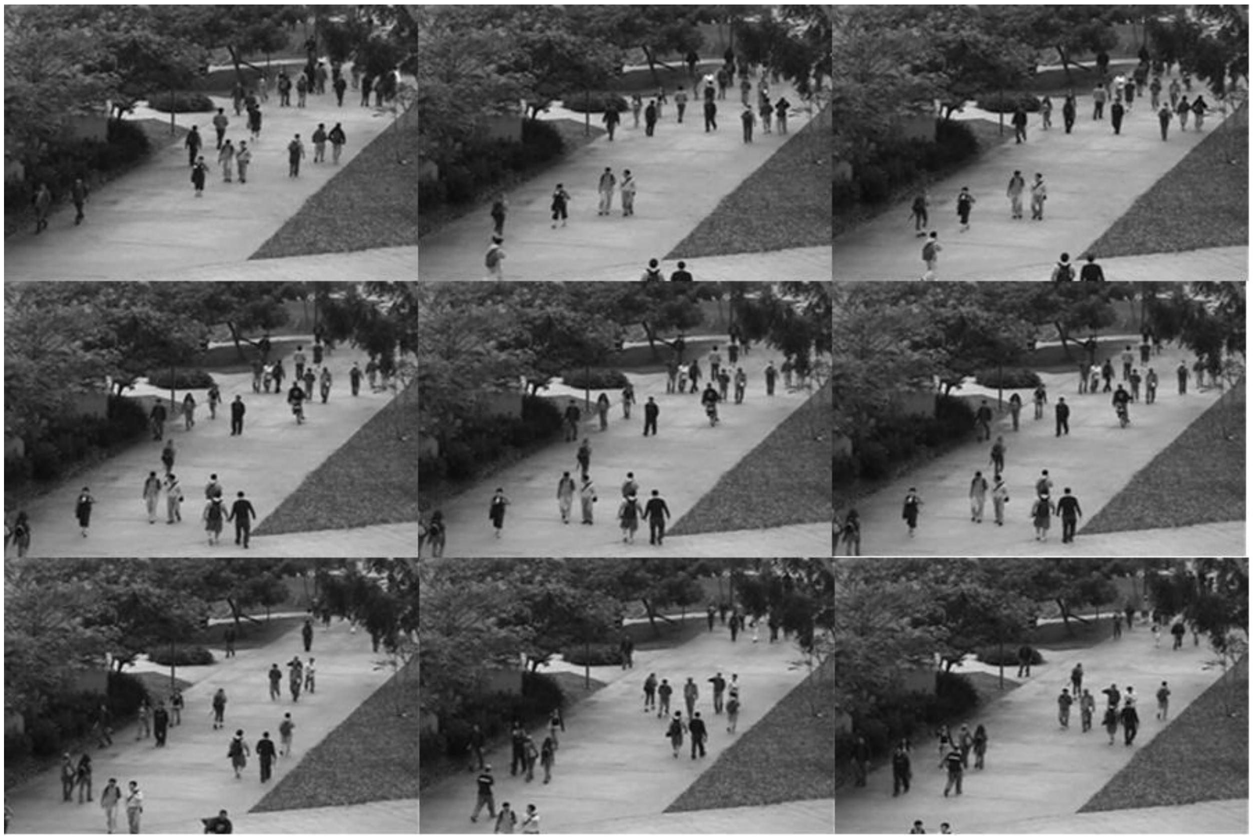
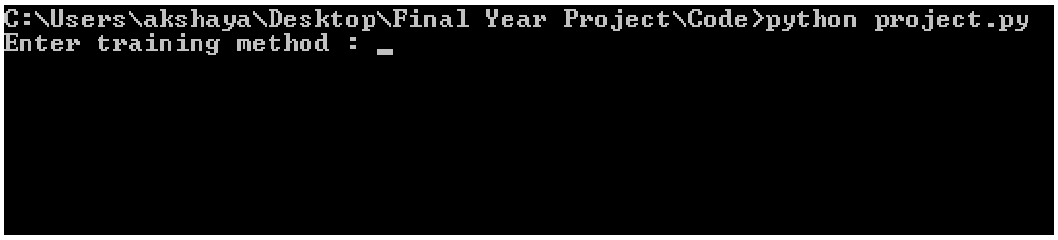
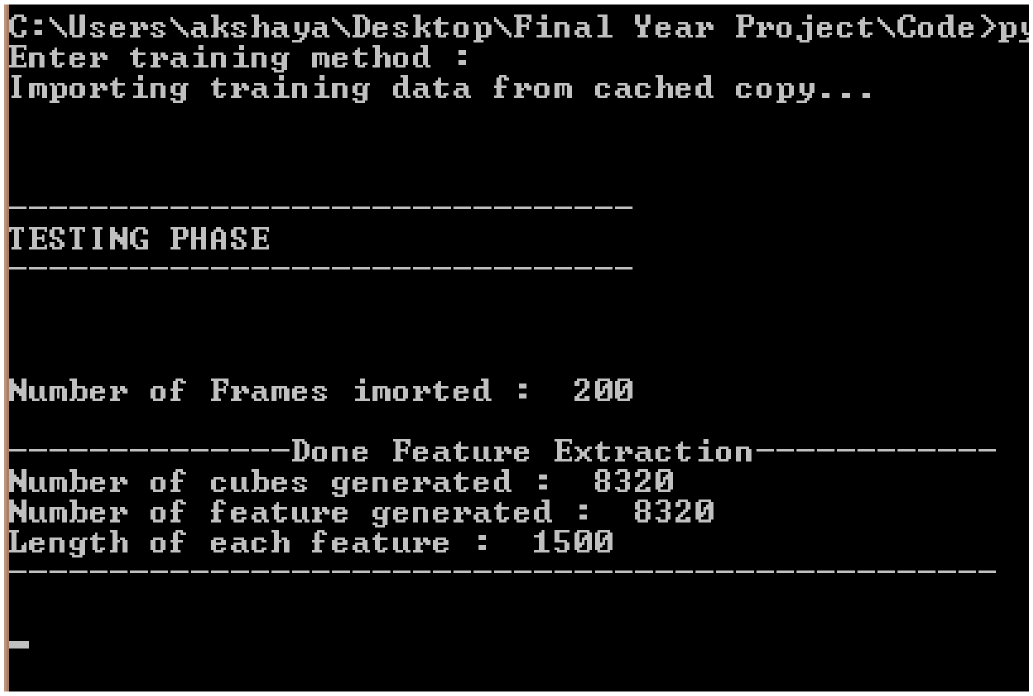


Figure 6.3 - Actual output frames



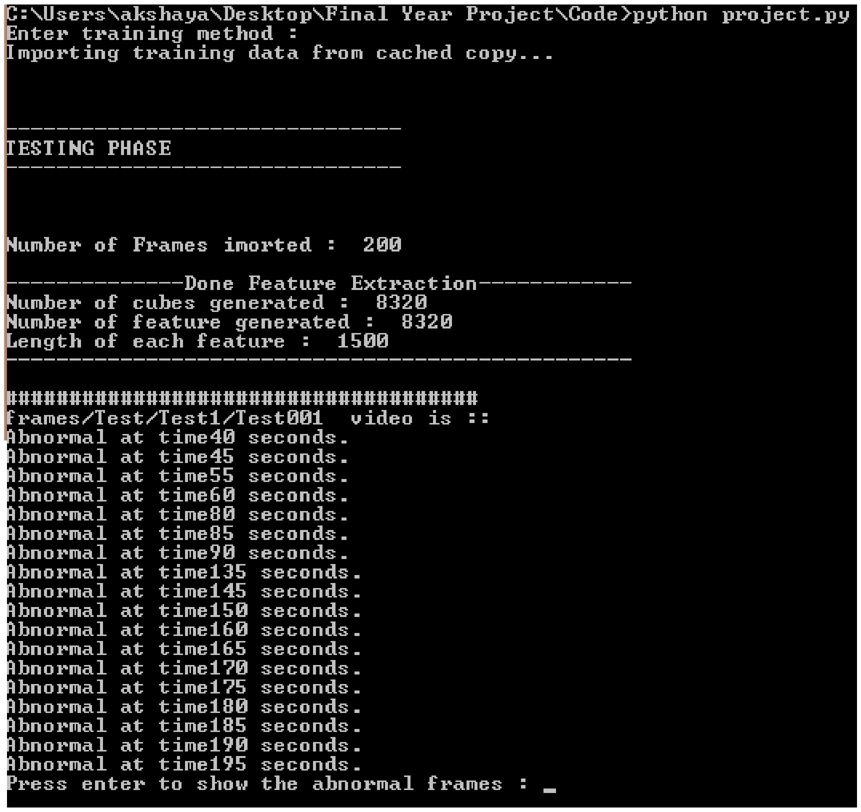
```
C:\Users\akshaya\Desktop\Final Year Project\Code>python project.py
Enter training method : _
```

Figure 6.4 Snapshot 1 (Program Start)



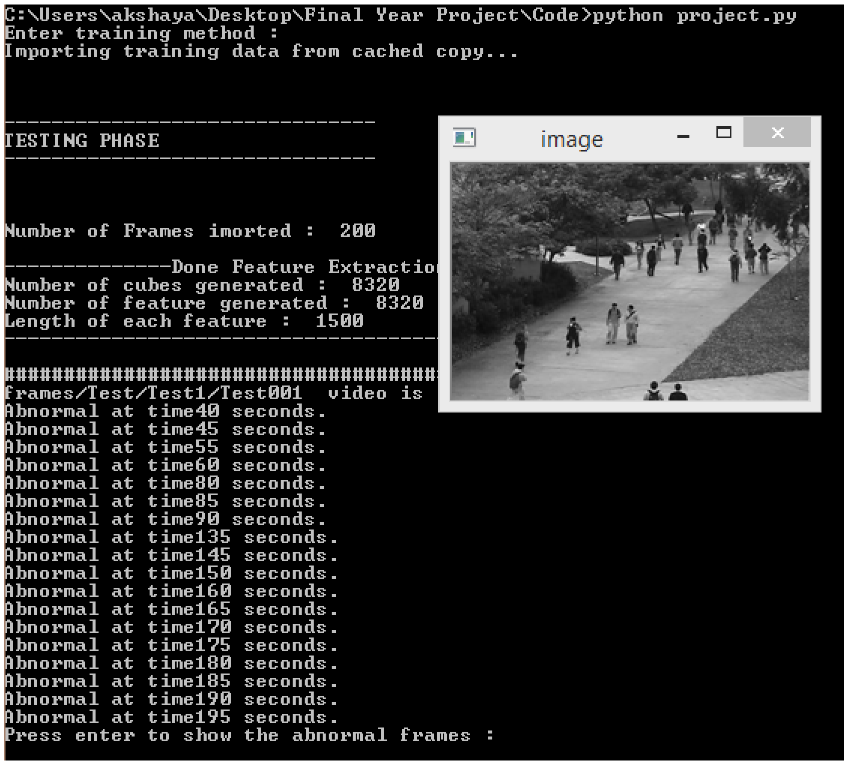
```
C:\Users\akshaya\Desktop\Final Year Project\Code>py
Enter training method :
Importing training data from cached copy...



-------------------------------
TESTING PHASE
-------------------------------



Number of Frames imorted :  200

--------------Done Feature Extraction------------
Number of cubes generated :  8320
Number of feature generated :  8320
Length of each feature :  1500
-------------------------------------------------


_
```

Figure 6.5 Snapshot 2 (Pre-processing and Feature Extraction)



```
C:\Users\akshaya\Desktop\Final Year Project\Code>python project.py
Enter training method :
Importing training data from cached copy...



-------------------------------
TESTING PHASE
-------------------------------



Number of Frames imorted :  200

--------------Done Feature Extraction------------
Number of cubes generated :  8320
Number of feature generated :  8320
Length of each feature :  1500
-------------------------------------------------

#####################################
frames/Test/Test1/Test001  video is ::
Abnormal at time40 seconds.
Abnormal at time45 seconds.
Abnormal at time55 seconds.
Abnormal at time60 seconds.
Abnormal at time80 seconds.
Abnormal at time85 seconds.
Abnormal at time90 seconds.
Abnormal at time135 seconds.
Abnormal at time145 seconds.
Abnormal at time150 seconds.
Abnormal at time160 seconds.
Abnormal at time165 seconds.
Abnormal at time170 seconds.
Abnormal at time175 seconds.
Abnormal at time180 seconds.
Abnormal at time185 seconds.
Abnormal at time190 seconds.
Abnormal at time195 seconds.
Press enter to show the abnormal frames : _
```

Figure 6.6 Snapshot 3 (Testing Output)



```
C:\Users\akshaya\Desktop\Final Year Project\Code>python project.py
Enter training method :
Importing training data from cached copy...



-------------------------------
TESTING PHASE
-------------------------------



Number of Frames imorted :  200

--------------Done Feature Extractio
Number of cubes generated :  8320
Number of feature generated :  8320
Length of each feature :  1500
-------------------------------------

####################################
frames/Test/Test1/Test001  video is
Abnormal at time40 seconds.
Abnormal at time45 seconds.
Abnormal at time55 seconds.
Abnormal at time60 seconds.
Abnormal at time80 seconds.
Abnormal at time85 seconds.
Abnormal at time90 seconds.
Abnormal at time135 seconds.
Abnormal at time145 seconds.
Abnormal at time150 seconds.
Abnormal at time160 seconds.
Abnormal at time165 seconds.
Abnormal at time170 seconds.
Abnormal at time175 seconds.
Abnormal at time180 seconds.
Abnormal at time185 seconds.
Abnormal at time190 seconds.
Abnormal at time195 seconds.
Press enter to show the abnormal frames :
```

Figure 6.7 Snapshot 4 (Display of Result)

CHAPTER-7

CONCLUSION AND FUTURE ENHANCEMENT

We have presented an abnormal event detection method via sparse combination learning. This approach directly learns sparse combinations, which increase the testing speed hundreds of times without compromising effectiveness. Our method achieves state-of-the-art result sin several datasets. It is related to but differ largely from traditional subspace clustering.

In a time where surveillance cameras are being used everywhere, effectively checking it for any abnormal event would be a bottleneck. Thus a fast and intelligent method to check theses surveillance cameras is at most required. It would help in cutting down a lot of work to be done by people struggling to monitor it and would help it taking faster actions during those situation by integrating these with alarms and other important actions like informing the police or calling an ambulance.

Since it achieves a frame rate of 100fpm, frames can be analyzed at a decent rate and thus can be used in surveillance cameras to detect abnormalities automatically. Based on the signal of this system, alarms and other actions can be controlled.

Our future work will be to extend the sparse combination learning framework to other video applications. This can be extended to detect various other kind of abnormal event that are usually encountered. By doing this a system which can detect any abnormality will be build which can deployed in various environments just encouraging portability. The algorithm should be carefully analyzed for areas where parallel processing is possible and suitably the algorithm should be tweaked which can help in achieving better accuracy.

CHAPTER-8

REFERENCES

[1]A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz. Robust real-time unusual event detection using multiple fixed location monitors. IEEE TPAMI, 30(3):555–560, 2008.

[2] B. Antic and B. Ommer. Video parsing for abnormality detection. In ICCV, pages 2415–2422, 2011.

[3] Y. Benezeth, P.-M. Jodoin, V. Saligrama, and C. Rosenberger. Abnormal events detection based on spatio-temporal co occurrences. In CVPR, 2009.

[4] D. Bertsekas. Nonlinear programming. Athena Scientific Belmont, MA, 1999.

[5] Y. Cong, J. Yuan, and J. Liu. Sparse reconstruction costs for abnormal event detection. In CVPR, pages 3449–3456, 2011.

[6] X.Cui, Q.Liu, M.Gao, and D.Metaxas. Abnormal detection using interaction energy potentials. In CVPR, pages 3161– 3167, 2011.

[7] E.Ehsanand R.Vidal. Sparse subspace clustering. In CVPR, 2009.

[8] F. Jianga, J. Yuan, S. A. Tsaftarisa, and A. K. Katsaggelosa. Anomalous video event detection using spatiotemporal context. Computer Vision and Image Understanding, 115(3):323–333, 2011.

[9] K.Jouseokand L.Kyoungmu. A unified frame work for event summarization and rare event detection. In CVPR, 2012.

[10] J. Kim and K. Grauman. Observe locally, infer globally: a space-time mrf for detecting abnormal activities with incremental updates. In CVPR, pages 2921–2928, 2009.

[11] L. Kratz and K. Nishino. Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. In CVPR, pages 1446–1453, 2009.

[12] C. Lu, J. Shi, and J. Jia. Online robust dictionary learning. In CVPR, 2013.

[13] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos. Anomaly detection in crowded scenes. In CVPR, 2010.

[14] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. The Journal of Machine Learning Research, 11:19–60, 2010.

[15] R. Mehran, A. Oyama, and M. Shah. Abnormal crowd behavior detection using social force model. InCVPR,2009.

[16] V. Saligrama and Z. Chen. Video anomaly detection based on local statistical aggregates. In CVPR, pages 2112–2119, 2012.

[17] J.Shi, X.Ren, G.Dai, J.Wang, and Z.Zhang. A non-convex relaxation approach to sparse dictionary learning. In CVPR, pages 1809–1816, 2011.

[18] H. Trevor, T. Robert, and J. H. Friedman. The elements of statistical learning .Springer New York, 2001.

[19]X.Wang,X.Ma, and E. Grimson. Unsupervised activity perception by hierarchical Bayesian models. In CVPR ,pages 1–8, 2007.

[20] S. Wu, B. E. Moore, and M. Shah. Chaotic invariants of lagrangian particle trajectories for anomaly detection in crowded scenes. In CVPR, 2010.

[21] D. Zhang, D. Gatica-Perez, S. Bengio, and I. McCowan. Semi-supervised adapted hmms for unusual event detection. In CVPR, 2005.

[22]B.Zhao ,L.FeiFei ,and E.Xing. Online detection of unusual events in videos via dynamics parse coding. In CVPR,2011.

[23] H.Zhong, J.Shi, and M.Visontai. Detecting unusual activity in video. In CVPR , 2004.

[24] Fan Jiang ; Department of Electrical Engineering and Computer Science, Northwestern University, 2145 Sheridan Rd, Evanston, IL 60208, USA. fji295@eecs.northwestern.edu ; Ying Wu ; Aggelos K. Katsaggelos, Abnormal Event Detection from Surveillance Video by Dynamic Hierarchical Clustering

[25] Uday Kiran Kotikalapudi IISC Bangalore ,Abnormal Event Detection in Video.

[26] Yang Cong ; School of EEE, Nanyang Technological University, Singapore ; Junsong Yuan ; Ji Liu, Sparse reconstruction cost for abnormal event detection.

[27] Sung Chun Lee , Ram Nevatia,Hierarchical abnormal event detection by real time and semi-real time multi-tasking video surveillance system.

[28] Iulian Pruteanu-Malinici ; Duke Univ., Durham ; Lawrence Carin ,Infinite Hidden Markov Models for Unusual-Event Detection in Video.

[29] E. L. Andrade ; IPAB, School of Informatics, University of Edinburgh, UK ; S. Blunsden ; R. B. Fisher, Modeling Crowd Scenes for Event Detection.

[30] Chun-ku Lee ; National Tsing Hua University, Taiwan ; Meng-fen Ho ; Wu-sheng Wen ; Chung-lin Huang, Abnormal Event Detection in Video Using N-cut Clustering.

[31] Zhong Zhang ; Tianjin Normal University, China ; Xing Mei ; Baihua Xiao, Abnormal Event Detection via Compact Low-Rank Sparse Learning.

[32] Shwu-Huey Yen ; Dept. of Compute. Sci. & Inf. Eng., Tamkang Univ., Taipei, Taiwan ; Chun-Hui Wang, Abnormal Event Detection Using HOSF.

[33] P. M. Ashok Kumar ; Department of CSE, Vel Tech University, Chennai ; V. Vaidehi ; E. Chandralekha Shwu-Huey Yen ; Dept. of Comput. Sci. & Inf. Eng., Tamkang Univ., Taipei, Taiwan ; Chun-Hui Wang ,Video traffic analysis for abnormal event detection using frequent item set mining.