

COM S 575 – Computational Perception

Spring 2015

THE EYE

Abhineet Sharma

Masters Student

Department of Computer Science

Iowa State University

abhineet@iastate.edu


Sayantani Ghosh

PhD Student

Department of Computer Science

Iowa State University

ghosh@iastate.edu



ABSTRACT

‘The Eye’ is an attempt at detecting anomalies in a scenario, by means of a video or a live camera feed, based on the system deciding based on learning from the history of the input device. A camera has a fixed viewing angle and thereby sees routine activities of various objects in its view every day. A camera can, then, learn from that routine and be ‘intelligent’ enough to judge any activity which is off the routine. Since the camera is designed to detect anomalies in real time, we cannot model it using a very high dimensional feature space. Also, all the calculations must be fast enough and they must not compromise a lot on accuracy. The idea is to extract few important features out of the objects movements and based on those features, classify their activity as normal or abnormal. This report presents a basic idea, the implementation and possibility of further exploration into the same in order to detect anomalies in live video feed.

1. INTRODUCTION

‘The Eye’ is a vision based application that would detect anomalies in a live video feed. The idea of this application is to assist in video surveillance systems. Consider a day in life of a surveillance team in charge for monitoring highways. It is not possible for all different cameras to be monitored at the same time. Moreover, if some detail misses the human eye, the application will be able to detect it. For example, a car pulled over to the side on a deserted highway in need of assistance, and the team in charge of surveillance somehow missed the catching the detail, then the camera could be of assistance. In other words, we made an attempt to build “intelligence” into cameras that can help assisting the teams and, subsequently, increase the productivity.

With this idea in mind, we made an attempt to build an application which would learn an environment and its constituents and then detect any anomalies in that environment. Now, let us define some of the terms we have used as follows:

1. Environment:

Environment implies the field of view of the camera i.e. the image recorded in each frame of the camera.

2. Normal Behavior:

Normal behavior refers to routine events. For example, people walking on the streets or cars moving on road.

3. Anomaly or Abnormal Behavior:

Anomalous or abnormal behavior refers to off-routine events. For example, people crowded in the middle of the street or a car skidding off the path.

The thing to be noted here is that not all anomalous behavior has to be a bad event, which might demand action at the immediate moment. The Eye simply serves as a means for alerting the

surveillance team, who can then decide whether an action needs to be taken or not. For example, people gathered in an area does not necessarily mean that an urgent action is necessary, unless somebody gets hurt or somebody picks up a fight. Unfortunately, the cameras are not intelligent enough to understand these subtle differences or differentiate right from wrong without human intervention. The Eye would, therefore, simply highlight any event it finds different from an everyday routine.

We have also attempted to further describe the ‘routine’, since a routine may be different for different places, different times of the day and so on. For example, if on every Monday afternoon, a group of people gather in an area on the street, it will be ‘abnormal’ on the first day or two, but eventually, it would become a part of the routine, and so the application would adapt itself to treat the gathering as normal.

For all intents and purposes in this project, we will strictly define our environment, normal and abnormal conditions.

1. Environment:

The environment we use are the two views from the Iowa State University webcam – Memorial Union and West Lawns. Also, since the camera is movable, we fix our view to the default viewing frame that has been setup in the webpage.

2. Normal Behavior:

Normal behaviors, in our case, will be people walking and riding bikes or skateboards on the pathway in the West Lawns or on the pathway outside Memorial Union and cars, buses and motorcycles driving on the street outside Memorial Union.

3. Abnormal Behavior:

Abnormal Behaviors will be cases such as gathering of people on the pathways or on the grass in the West Lawn or outside Memorial Union. Also, since cars are not allowed in the West Lawns area, a car driving on a walkway will be an abnormal behavior.

However, if a gathering as such occurs on a routine basis, it will eventually be categorized as normal.

Following are some of the images signifying normal behavior in our environment:



Figure 1: Iowa State University Memorial Union webcam



Figure 2: Iowa State University West Lawns webcam

Following image shows an anomaly which may not refer to any emergency situation, but since it is different from the normal routine, we count it as an abnormality:



Figure 3: Abnormal activity detected in the Iowa State University West Lawns webcam

2. NEED FOR APPLICATION

Imagine somebody meeting with an accident on a deserted highway. Now, if there is no one around to alert, it might take really long for help to arrive. Moreover, if somebody missed taking notice of the event at the time it was happening, nobody would be able to know if at all something happened. How about when you saw that there were a bunch of people gathered in the middle of the road, possibly due to an accident? Wouldn't it be better if there was someone to attend to the situation as soon as possible? The situation would turn bad if there aren't enough personnel to observe the live feed so that all the cameras can be analyzed at the same time. Also, a handful of guards patrolling the roads or observing a surveillance camera may not always be as quick to notice and respond to situations like these.

Now, imagine if the surveillance camera was 'intelligent' enough to raise an alert if such a situation occurs. Furthermore, it would be able to alert the authorities over a possibility of an emergency situation without waiting on someone to dial 911.

Our aim, therefore, is to develop an application that will be able to provide assistance in case of 'abnormal' situations on the road.

3. TARGET AUDIENCE

The target audience for this project involves all the registered participants along with our professor for COMS 575. The idea is to convince them of the possibility of intelligent camera surveillance systems.

We hope to work on it further and put it to use on an everyday environment.

4. RELATED WORK

1. STAUFFER C., GRIMSON W. Adaptive Background Mixture Models for real-time tracking. Artificial Intelligence Laboratory, MIT.

Highlights:

1. Model each pixel in image
2. Compare incoming images with the modelled image
3. Resulting Tracker has following properties :
 - 3.1. Real-time,
 - 3.2. Robust – deals with lighting changes,
 - 3.3. Repetitive Motion from Clutter,
 - 3.4. Long term scene changes

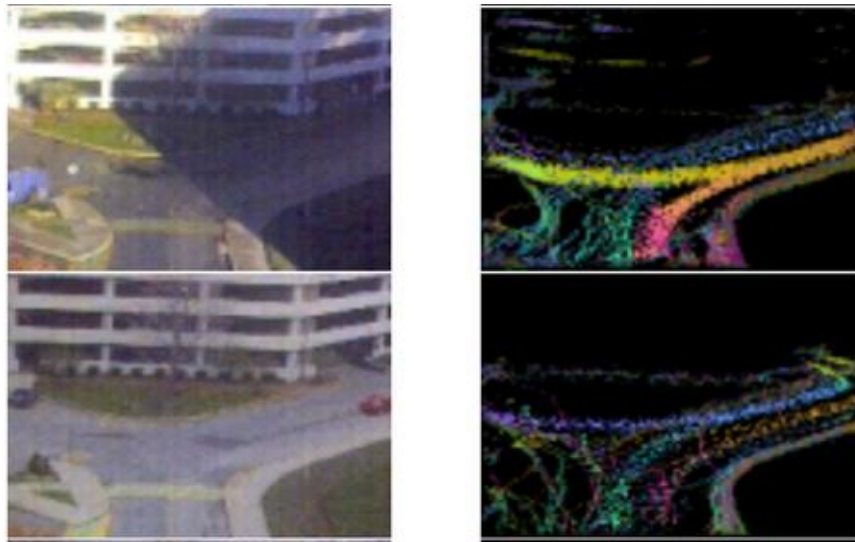


Figure 4: This figure shows consecutive hours of tracking from 6am to 9am and 3pm to 7pm. (a) shows the image at the time the template was stored and (b) shows the accumulated tracks of the objects over that time. Color encodes direction and intensity encodes size.

2. CHELLAPPA R., VEERAGHAVAN A., AGGARWAL G. Pattern Recognition in Video. University of Maryland.

Highlights:

1. Discusses the application of pattern recognition in video.
2. Activity modeling and anomaly detection.
3. Activity Model used is Hidden Markov Model (HMM).
4. Talks about two types of changes :
 - 4.1. Drastic
 - 4.2. Slow

APPROACH

Our approach includes detecting moving objects in a video and extract certain features about objects activity and train a model using that data. Then, we can use this model to classify objects detected in an environment of the same camera based on objects activity. From here on, we would use the terms, video and live feed, interchangeably because there is not much of a technical difference between the two.

To begin with the approach, let us first define the parameters used for training the data model.

1. Life

Life of an object is measured in terms of the number of frames the object existed in the video.

2. Speed

The speed at which object moves is measured in terms of Euclidean distance between the start point and end point divided the number of frames. For efficiency reasons, we are not taking the square root of to get the actual distance, just measuring the distance squared divided by life of object.

$$Speed = \frac{(|endPoint - startPoint|)^2}{life}$$

3. Slope & Intercept

Based on the object's movement, we fit a linear line and calculate the slope and intercept of the same using the following equation:

$$y = mx + c$$

Here, m is the slope and c is the intercept.

4. Average X & Average Y

Along with the other parameters, we are also using average X and average Y of the objects location. This is to determine the center of objects activity, which can be used to classify it as normal or abnormal.

So, we broke the entire task into various sub-tasks, which involve collection of data, classification followed by training of a data model and then a test phase and a model update phase. Following are the sub-tasks identified along with the details:

Before, we could extract the identified parameters from the input video, we had a preliminary task of extracting the live camera feed from the public cameras of Iowa State University. We selected two such cameras:

1. West Lawns Camera
2. Memorial Union Camera

Preliminary Task: Extract Live Camera Feed

Due to time limitations, we wrote a Java Program to read the Motion - JPEG (MJPEG) feed over http and connect the extracted images into a mov file as follows:

1. Read the camera feed over HTTP
2. Initialize mov file writer
3. For i from 1 to 200
 - 3.1. Extract image from the http data stream
 - 3.2. Append to the mov file
4. Finalize mov & save the file.

Here's how the MJPG format looks like:

--myboundary

Content-Type: image/jpeg
Content-Length: 24421

<IMAGE1 in binary>

--myboundary

Content-Type: image/jpeg
Content-Length: 24268

<IMAGE 2 in binary>

And so on.

Once, a certain number of images are extracted, they are joined together as a .mov video file, which is used for further processing. We have used 500 image threshold for collecting and processing West Lawns video feed and a 200 image threshold for Memorial Union video feed.

After completion of preliminary task, following are the sub-tasks for the successful completion of project. Following gives a brief overview of the sub-tasks. Details about the algorithms used in the various tasks are provided in the algorithms section.

1. Data Gather Mode:

We have used basic algorithms to detect objects and collect data about objects activity. The data gathered is noisy but still the algorithms work with significant accuracy. Because of the limitations of time and expertise, we focused more towards getting the concept working because, there are various methods available which can help us reduce the noise in the data. Once the data is

collected, we require human supervision, to classify the data so that data model can be trained and tuned using available machine learning algorithms.

2. Training Mode

Once the data is classified, the program can be run in training mode, where it trains data model using two different approaches: K-Nearest Neighbors (KNN) & Support Vector Machines. We have used, 10-fold cross validation in order to tune the models. Finally, best performing model is selected by evaluating both the models against test set and selecting the one with the best performance.

3. Test Mode:

After finalizing the data model, we used test videos to check the performance of our program. Again the output was evaluated manually, to see how the code was performing.

4. Model Update Mode:

We have also provided a provision for model update, wherein, the program saves the history of all the objects detected in a comma separated file (during runs in test mode). When the program is run in model update mode, it retrieves the data and expects an input from a supervisor (human) so that it can retrain and select a model in the same fashion as in training mode.

5. DATA FLOW

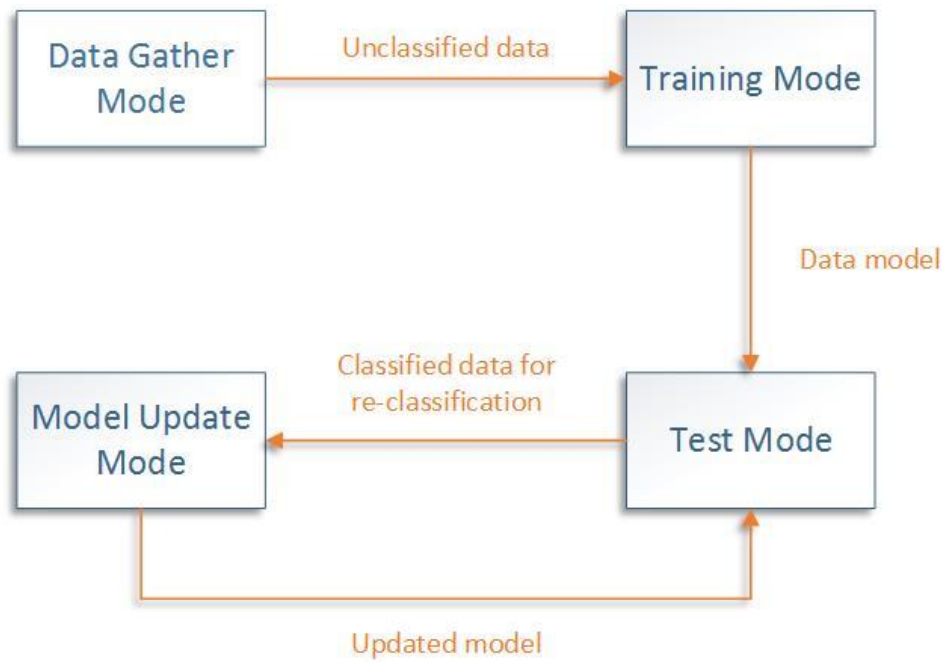


Figure 5: Phases of implementation



Figure 6: Data Gather Phase

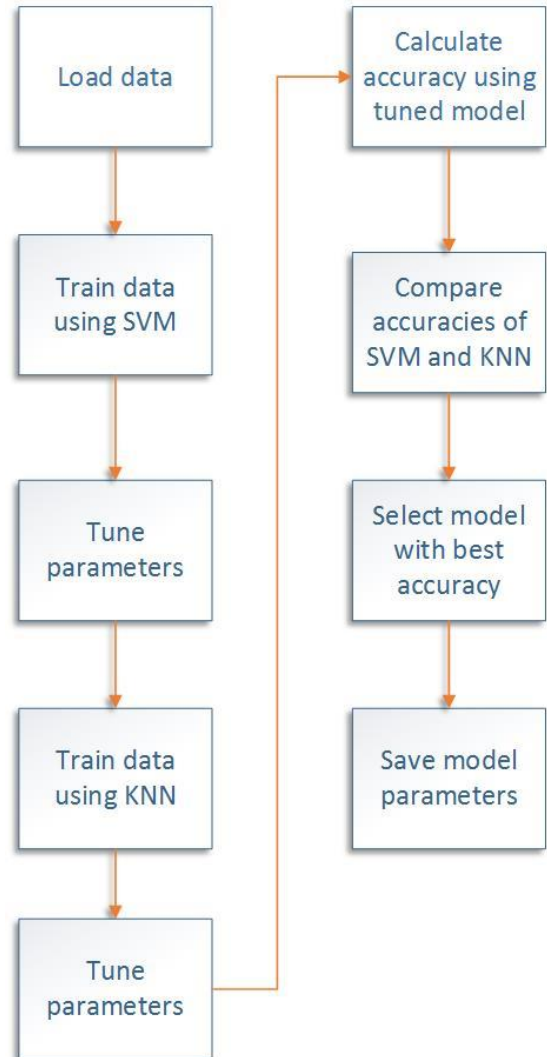


Figure 7: Training Phase

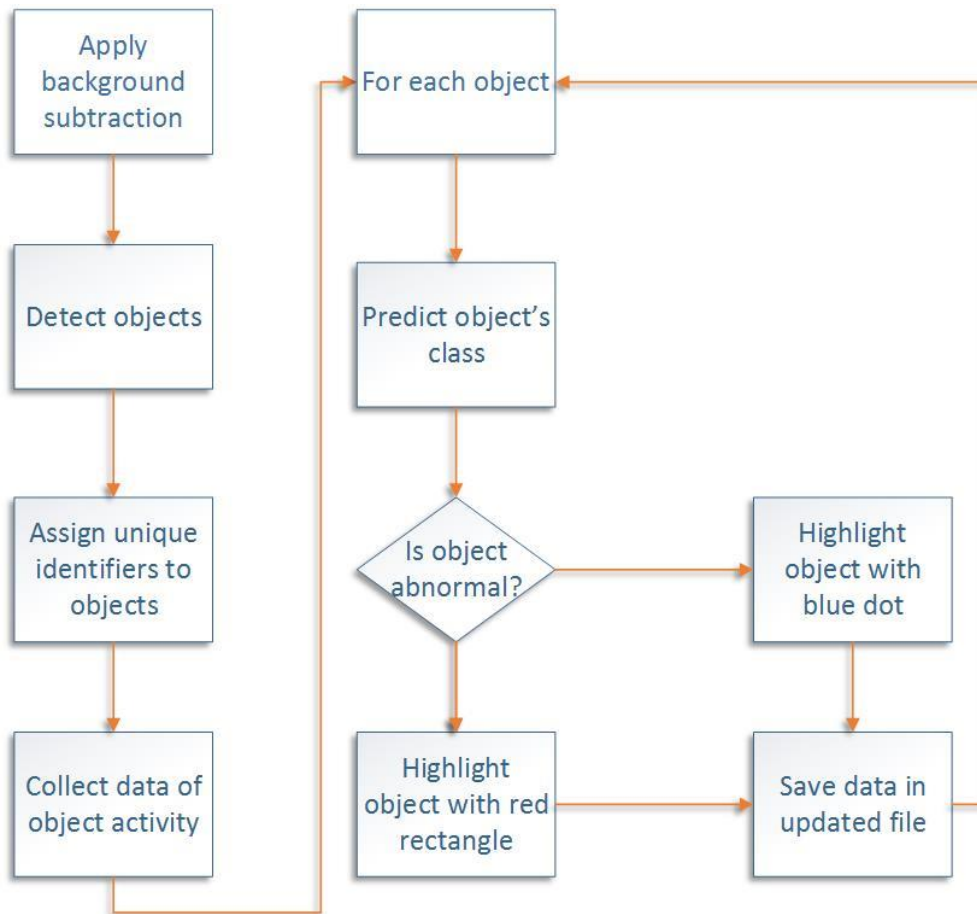


Figure 8: Test Phase

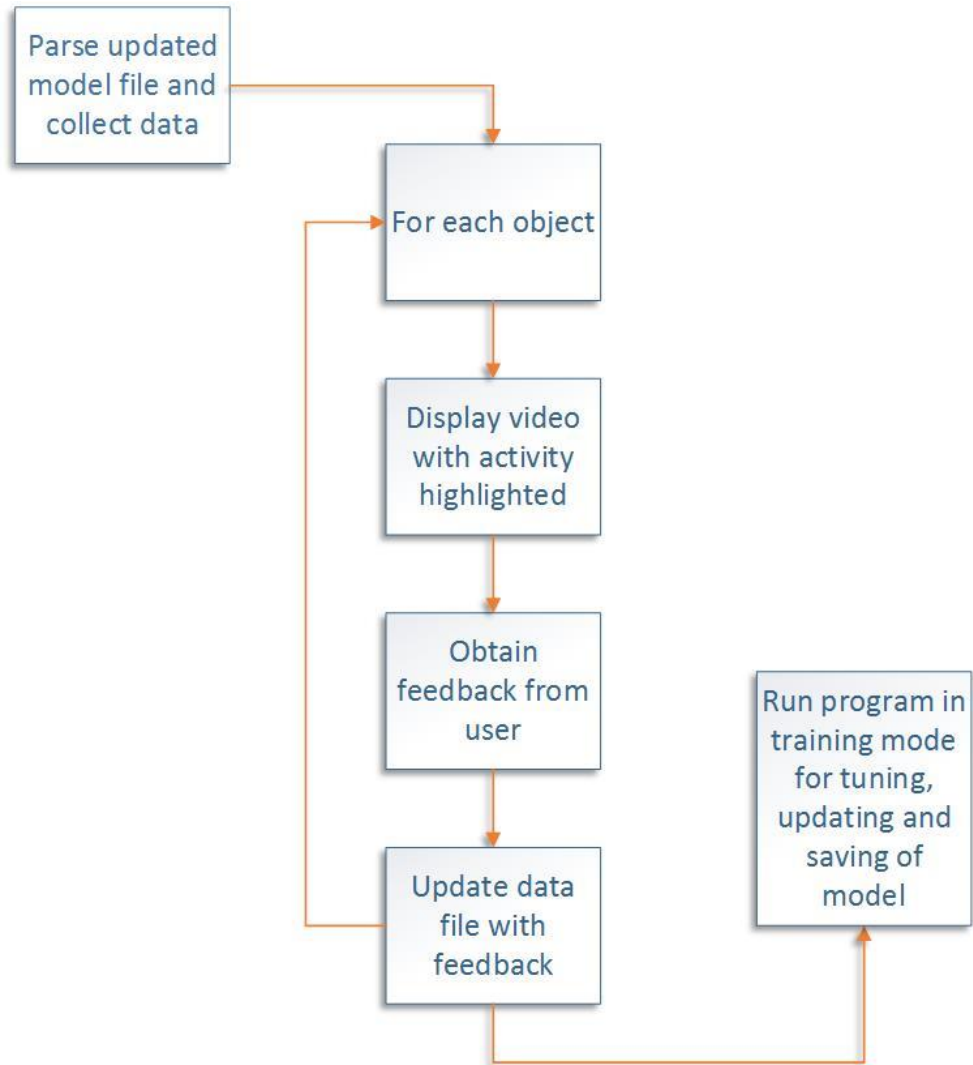


Figure 9: Model Update Phase

6. FEATURES

Following is the feature list of 'The Eye':

1. Object detection:

The application subtracts the background from the foreground, and then identifies the various moving objects on the foreground.

2. Object activity classification:

The application then detects a pattern among the movements based on certain parameters and accordingly, classifies new objects as normal or abnormal.

3. Data Model Update:

The user is able to provide updates on the system whether an object has been classified correctly and if not, input the correct classification and accordingly, the model updates itself.

7. EQUIPMENT

We have identified two live feeds at Iowa State University campus:

1. Memorial Union Web Cam
2. West Lawns Web Cam [Parks Library]

We have used Eclipse to design a program that extracts the feed from the webcam frame by frame and then strings them together to make it into a video file.

For our development environment, we have used RStudio for a basic analysis of data and for preliminary test of the machine learning algorithms at our disposal. Since, the implementation of the program is in C++, we had to stick to OpenCV as a basic platform to evaluate all the machine learning algorithms. We used Microsoft Visual Studio 2013 on a Windows platform (64-bit) with Open CV as the video and image processing library for video processing.

8. ALGORITHMS

We have divided up our algorithms based on 3 phases –Preliminary, Setup and Run. The last two phases are then sub-divided into modes where the Setup phase consists of Data Gather mode and Training mode, and the Run phase consists of Test mode and Model Update mode.

Preliminary: Video Extraction

Once you have the URL of the public web cam feeds of Iowa State University, do

1. Open an HTTP connection
2. Read the Data using data stream
3. Parse the data according the MJPG format specified in the previous section
4. Extract a certain number of images from the live feed.
5. Connect them all and save as .mov file.

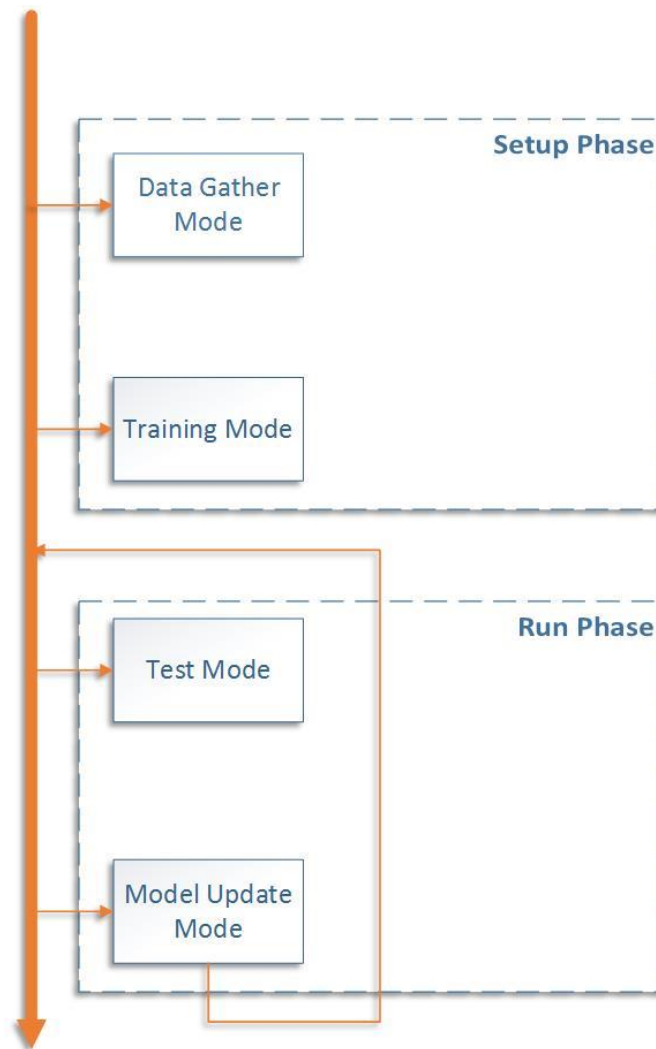


Figure 10: Phases and Modes

Data Gather mode:

For each file in training directory, do

1. Use background subtraction to separate background and foreground.
2. Detect objects & assign them unique identifiers.
3. Collect data about objects activity. The values of data include: Speed, Slope, Intercept, Average X, Average Y and Life.
4. Save them to file.

Training mode:

Before, running the program in Training mode, there is a manual step involved, which includes manually classifying the objects detected as normal/abnormal. Once, the data is classified, then the program can be run in training mode.

1. Load Training data as well as Test data
2. Train data model using Support Vector Machines (SVM).
3. Tune parameters using 10-fold cross validation.
4. Calculate accuracy using the tuned model on test set.
5. Train data model using K-Nearest Neighbors (KNN).
6. Tune parameters using 10-fold cross validation to select optimum K.
7. Calculate accuracy using the tuned model.
8. Compare accuracies of SVM and K-NN models.
9. Select the model with best accuracy and save the model parameters.

Test mode:

For input file, do

1. Use background subtraction to separate background from foreground.
2. Detect objects & assign unique identifiers to identify them.
3. Collect data about objects activity.
4. For each object detected in each frame, do
 - 4.1. Use the model to predict object's class based on the data collected about object's activity so far.
 - 4.2. If object is categorized as abnormal,
 - 4.2.1. Highlight the object with red dot and a rectangle
 - 4.3. If object is categorized as normal,
 - 4.3.1. Highlight it using a blue circle, just to indicate object detection.
5. Save the data in updated file, for re-classification (manual step) and model update.

Model Update mode:

1. Parse the updated model file to collect data about various objects detected during test run.
2. For each object, do
 - 2.1. Display the video to user highlighting object's activity
 - 2.2. Obtain feedback from the user for re-classification.
 - 2.3. Update the data file with user's feedback.
3. Run the program in Training mode, which would tune, update and save the model.

9. EXECUTION

Based on the 4 modes that we have designed, the outputs and observations obtained at each stage are as follows:

1. Data Gathering Mode:

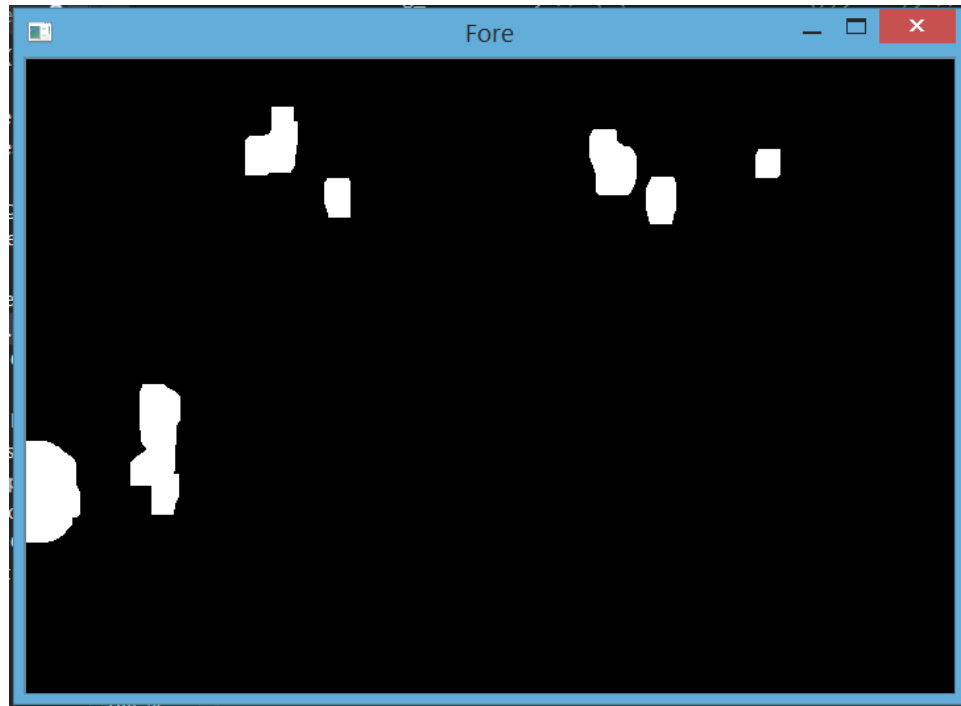


Figure 11: Thresholded image of moving objects after background subtraction



Figure 12: Moving objects are detected and tracked

2. Training Mode:

```

C:\Users\Abhineet\OneDrive\ProjectWork\C++\TheEye\x64\Debug\TheEye.exe
Support Vector Count : 244
No Of Rows in Test Data : 138
SUM Confusion Matrix
      A      N
A      2      46
N      8      82
Accuracy : 60.8696
  Best Accuracy : 0.485507
  Optimal K : 3
KNN Confusion Matrix
Accuracy : 48.5507
      A      N
A      2      46
N     25     65
ACCURACY of Data Model : 0.608696
SUMMARY :
C : 62.5
SUM Type :      C-Support Vector Classification
Kernel Type :  Radial Basis Function (Gaussian)
Gamma : 1e-005
Support Vector Count : 244

Saving Model at : D:/TheEyeTrainingdata/datamodelInfo.txt
Press any key to continue . . .

```

Figure 13: Results of the KNN and SVM Models

3. Test Mode:



Figure 14: Thresholded image of moving objects after background subtraction



Figure 15: Moving objects are detected, tracked and classified into normal and abnormal. The blue dots represent normal objects and the red squares represent abnormal objects.

4. Model Update Mode:



Figure 16: Displays tracked object activity

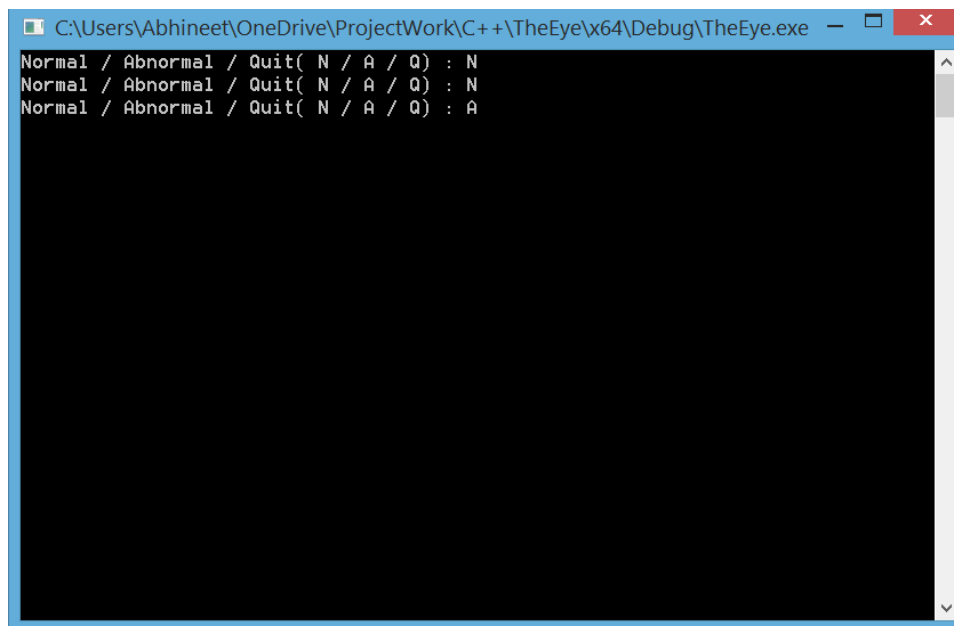
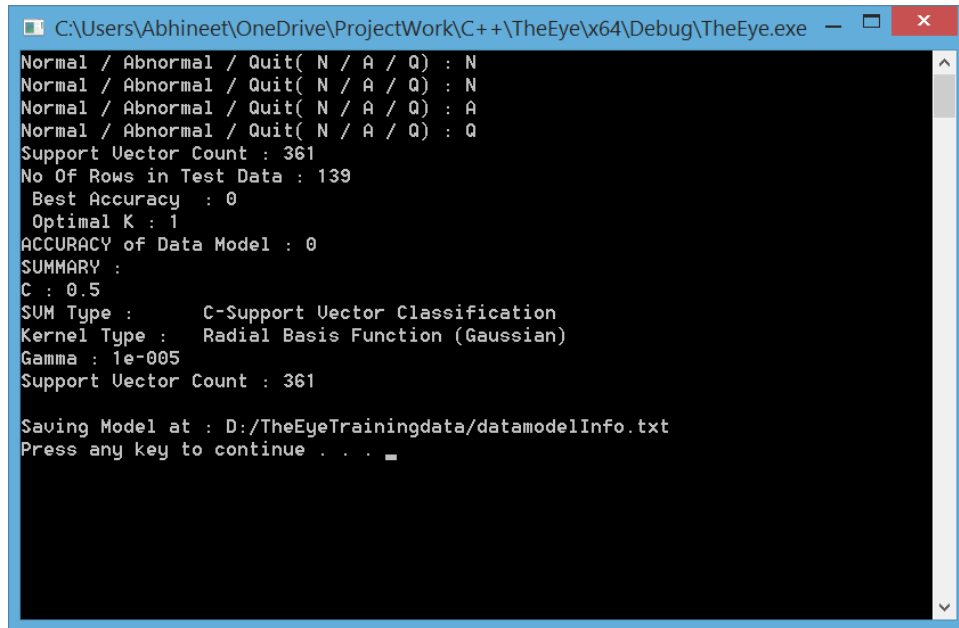


Figure 17: Asks user to classify the object tracked



```
C:\Users\Abhineet\OneDrive\ProjectWork\C++\TheEye\x64\Debug\TheEye.exe
Normal / Abnormal / Quit( N / A / Q) : N
Normal / Abnormal / Quit( N / A / Q) : N
Normal / Abnormal / Quit( N / A / Q) : A
Normal / Abnormal / Quit( N / A / Q) : Q
Support Vector Count : 361
No Of Rows in Test Data : 139
Best Accuracy : 0
Optimal K : 1
ACCURACY of Data Model : 0
SUMMARY :
C : 0.5
SUM Type : C-Support Vector Classification
Kernel Type : Radial Basis Function (Gaussian)
Gamma : 1e-005
Support Vector Count : 361

Saving Model at : D:/TheEyeTrainingdata/datamodelInfo.txt
Press any key to continue . . .
```

Figure 18: Displays model update summary after user input on all the objects

10. EVALUATION METHODOLOGY

We have divided our testing into 2 different environments for evaluation:

1. Test Environment:

Our test environment consists of sample videos recorded from the Iowa State webcam feeds, keeping the point of view same for each video so that the background is the same. In these videos, we already know the anomalies and so, we have tested the system against false positives and false negatives. A false positive implies an anomaly which has falsely been categorized as a normal behavior and false negative implies a normal behavior which has been falsely categorized as an abnormal behavior.

We have used 5 such videos in each of the 2 different views (Memorial Union and West Lawn), resulting in a total of approximately 10 videos on which the application was tested.

In this environment, our application has provided us with an 80% success rate.

2. Real Life Environment:

Testing in a real life environment includes testing for anomalies from the live feeds from Iowa State University webcams at Memorial Union and West Lawns. The approach was to extract the live feed from the web camera and detect anomalies on the go. Since we are previously not aware of what the anomalies will be, or if there will be at all, it is a greater challenge to achieve a high accuracy in this case.

Our application has provided us with a 60% accuracy in this case.

11. RESULTS AND ACCURACY

1. Support Vector Model

	Abnormal	Normal
Abnormal	2	46
Normal	8	82

Accuracy: 60.87%

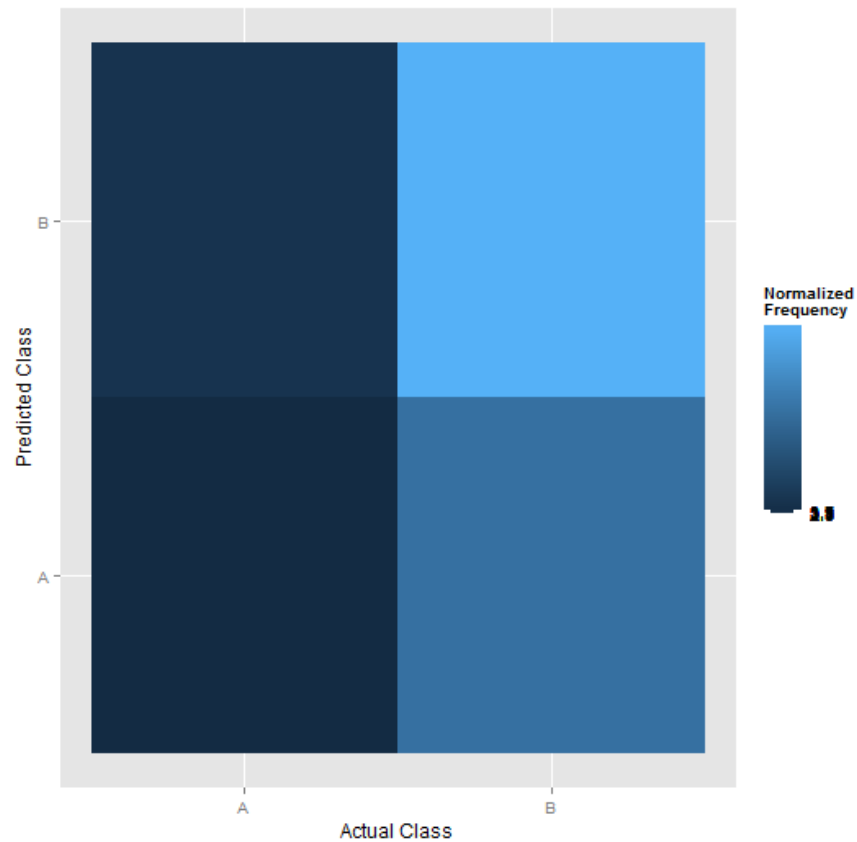


Figure 19: Confusion Matrix for SVM

2. K Nearest Neighbors

	Abnormal	Normal
Abnormal	2	46
Normal	25	25

Accuracy: 48.55%

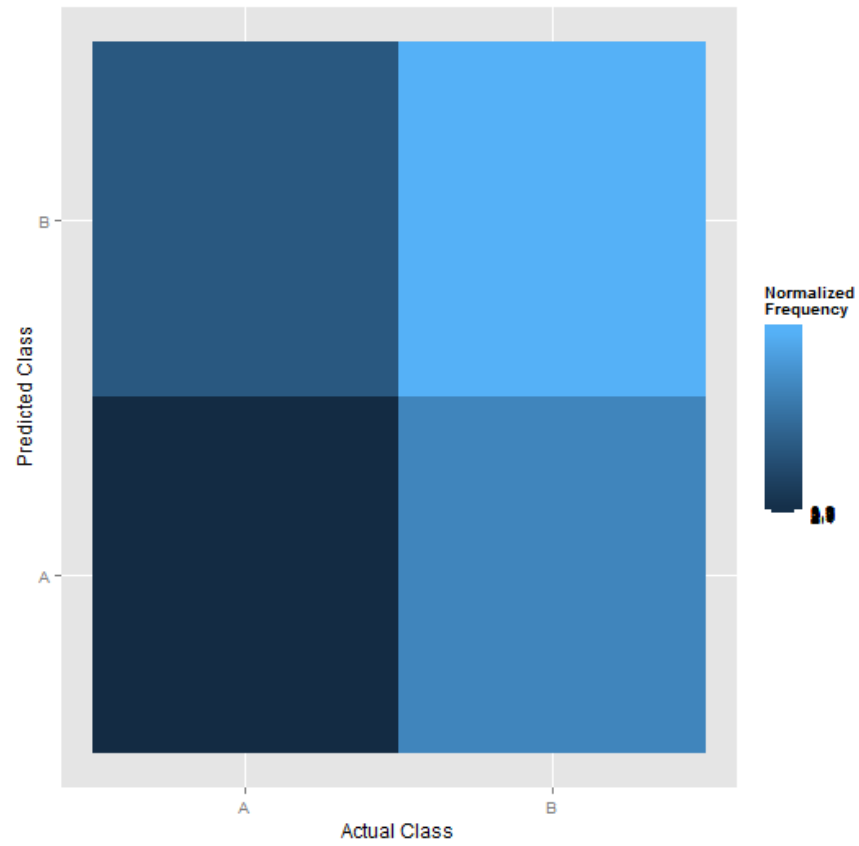


Figure 20: Confusion Matrix for KNN

12. CHALLENGES

We faced various challenges during the growth and development of the idea as follows:

1. Extracting the live video feed:

One of the biggest challenges was the extraction of live video feed. The idea was to directly run software on the live video feed of one of the Iowa State University's public cameras. But due to limited expertise in C++ with respect to socket programming, we had to switch over to Java. It took a while to understand the underlying protocol and extract the feed working at raw data level.

2. Selecting the Underlying Model:

Once we had the data, there was another challenge, which was rather unforeseen in the beginning.

2.1. Model Each Pixel :

Initially we started with the idea of modelling each and every pixel in the image and classify each and every one of the incoming image. This was ruled out in the very early stages due to the following limitations:

2.1.1. It was prone to lighting changes

2.1.2. There was a possibility that if every pixel was modelled as Gaussian then the standard deviation might go very large over the course of time and may not result in accurate predictions.

2.2. Temporal Modelling of Each Pixel :

Due to the limitations with the approach of modelling each pixel, we came with a time based modelling of every pixel. For example, West Lawns camera has got a routine which is similar for 5 days of weeks and then similar for 2 days. Again, it varies when university is in session and otherwise. But this seems to be a clumsy model and the odds of running into implementation issues are high. Moreover, it requires lot of data to be gathered over a large period of time. Hence, this idea was dropped.

3. C++ Libraries for Machine Learning:

We did put in a lot of struggle looking for libraries for implementation of Support Vector Machines and K-Nearest Neighbors in C++. Finally, we came across OpenCV's implementation of these algorithms. It made our task of using machine learning algorithms fairly easy.

13. SCOPE OF IMPROVEMENT

There are quite a few areas with a scope of improvement for the current project. Following are the areas that can be improved:

1. Minimizing human error:

Because of the limitations imposed by time and expertise, we had to manually classify the objects based on the IDs directly from the video files, which added to noise because of human errors in misclassifying some of them. A better method can be employed in order to assist the human supervisor or completely remove the need by using unsupervised learning methodologies to cluster different activities and then get a human feedback for the entire cluster.

2. Reducing noise in the gathered data:

Since, we have employed very basic algorithms to collect data, there is inherent noise in the data set. This noise comprises of the following:

2.1. If two objects are close, they are merged into one and some data is lost in the process. This results in noisy data.

2.2. Also, there is inherent noise caused by shaking of camera due to wind or movement of leaves.

Better algorithms can be devised that reduce the noise in the gathered data so that the trained models are better at classifying.

3. Providing more anomalies:

The data gathered has more data points corresponding to normal behavior and less corresponding to the anomalous behavior. Since, we are more concerned at detecting anomalous activities, it might help to gather more data on anomalous behavior.

4. Selecting a better model:

Due to limitations imposed by time, we could only try and evaluate two models: K-Nearest Neighbors & Support Vector Machines. Other methods for classification can also be tried for to choose a better model for classification.

5. Code Optimization:

The code written can be optimized further to enhance the software design as well as performance. Although, there were no performance issues while testing the code, but there is a lot of scope for improvement in this area.

14. ACKNOWLEDGMENTS

We would like to thank our teacher and mentor Dr. Alexander Stoytchev for his help and support and for constantly motivating us to do our best in this project. Also, we would like to thank Nate Kent for all the help and encouragement that he gave us.

15. REFERENCES

- [1] STAUFFER C., GRIMSON W. Adaptive Background Mixture Models for real-time tracking. *Artificial Intelligence Laboratory, MIT*.
- [2] CHELLAPPA R., VEERAGHAVAN A., AGGARWAL G. Pattern Recognition in Video. *University of Maryland*
- [3] LEE D.S. Effective Gaussian Mixture Learning for Video Background Subtraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, No. 5, May 2005
- [4] GREGGIO N., BERNARDINO A., LASCHI C., DARIO P., SANTOS-VICTOR J., Self-Adaptive Gaussian Mixture Models for Real-Time Video Segmentation and Background Subtraction. *10th International Conference on Intelligent Systems Design and Applications 2010*.
- [5] POWER P.W., SCHOONEES J.A., Understanding Background Mixture Models for Foreground Segmentation. *Proceedings Image and Vision Computing New Zealand 2002*.
- [6] KAEWTRAKULPONG P., BOWDEN R., An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection. *2nd European Workshop+ on Advanced Video Based Surveillance Systems, AVBS01. Sept 2001*.
- [7] ANTANI S., KASTURI R., JAIN R., A survey on the use of pattern recognition methods for abstraction, indexing and retrieval of images and video. *The Journal of the Pattern Recognition Society*, 35 (2002) 945-965. March 2001.
- [8] JOG A., HALBE S., Multiple Objects Tracking using CAMshift Algorithm in Open CV. *IOSR Journal of VLSI and Signal Processing*, Vol 1, Issue 2. Sep-Oct 2012.