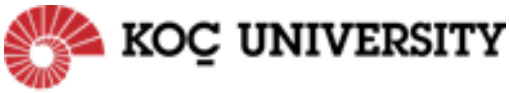


College of Engineering

COMP 408 Fall 2017

Term Project Final Report

Anomaly Detection in Crowded Scenes

Arda Kırkağaç, 49799, akirkagac14@ku.edu.tr

Gökberk Özsoy, 49721, gozsoy14@ku.edu.tr

09/01/2018

TABLE OF CONTENTS

Abstract

The aim of the project is detecting abnormalities in the crowd behavior. 'Abnormality' is defined as the situation that is not expected. In terms of crowd behavior, while people walks in slow pace, if an alarm situation happens, they start to run. Thus, an alarm situation is an abnormality for crowd behavior. In this project, we aim to find that 'alarm' situations in different crowd datasets. For this purpose, in parallel with the source paper [1], we first detect the optical flow between frames by using Farneback method. The magnitude of the difference of two consecutive frames will give us the activity map. The activity map shows how much motion is estimated between consecutive frames. Then, by using the differences between activity maps, we obtain Temporal Occupancy Variation (TOV) level and Entropy level as proposed in the source paper. By using specific thresholds to TOV and Entropy on each different dataset, we can detect the alarm situations by looking if TOV or Entropy levels exceeded their thresholds.

1. Introduction

In computer science, anomaly can be defined as the situations or the data points that are outside of the observed pattern, outside of the clusters. Anomaly detection is quite important for many applications areas in computer science. For example; monetary transactions or medical analyses should be carefully inspected for outliers (for fraud detection or for cancer detection, respectively).

In our context, anomaly -abnormal crowd behavior- can be considered as the abnormal movement of the people, objects, and so on. This abnormality can happen both over time, or over space. These are called spatial and temporal abnormality, respectively. Spatial abnormality is basically abnormalities happening in a region of an image, which is normally not supposed to happen. For example, a truck that is moving on the pedestrian way is causing spatial abnormality. Temporal abnormality stands for abnormal changes in a scene over time. For example, if people started to sprint suddenly after a bomb exploded, we can say that abnormal behavior was observed through time.

In the last decades, many researchers published solutions for this challenges, addressing both spatial abnormality and temporal abnormality. M. Shah et al. tried to capture crowd behavior using force model, considering social attractive and repulsive forces between people when moving [2]. R. Mehran et al. used spatio-temporal motion pattern models and

tried to model the behaviors using Hidden Markov Models [3]. S.Wu et al. used Lagrangian Particle Trajectories to model crowd flow and detected the chaotic invariants that captures abnormal model behavior [4]. Other works include employing dynamic textures, using convolutional neural networks to model the crowd behavior, and calculating dense trajectories and extending social force models [5][6].

In our project, in parallel with our source paper we try to implement an online solution to detect crowd abnormalities, which can be caused by panic. After calculating optical flow and calculating magnitudes using Farneback method, we create and activity map using multiple frames to see the flow persistency over time. The activity map is then used to create two metrics, TOV (Temporal Occupancy Variation) and entropy, respectively. We employ these metrics along with a threshold to classify each frame as normal or abnormal in a given sequence. Details will be discussed in the following sections.

2. Project Design and Implementation

The implementation of our project heavily based on our source paper. We write the code in MATLAB environment with using Computer Vision toolbox in order to calculate the optical flow easily. We first create a VideoReader object in order to read the video. Then we defined an opticalFlowFarneback object that is inherited from Computer Vision toolbox. We use this toolbox for estimating optical flow between consecutive frames. This estimation is the result of the theory of Farneback Optical Flow method. Farneback method is based on the paper 'Two-Frame Motion Estimation Based on Polynomial Expansion' written by Gunnar Farneback. It presents a two-frame motion estimation algorithm. In the first step, it uses polynomial expansion transform to approximate each neighborhood of both frames by quadratic polynomials. After a series of refinements, he estimates displacement fields form polynomial expansion coefficients.

Then, for each frame of the video, we converted that frame into gray image and applied a Gaussian filter to smooth. These steps are done also in the source paper. Next, we calculate the optical flow with the method of estimateFlow(opticalFlowFarneback, currentFrame) that is present in the Computer Vision toolbox. This method is the same method with `cv2.calcOpticalFlowFarneback()` that is present in OpenCV. The resulting flow object has two properties: angle and magnitude. The important property for us is the magnitude property since we care how much the optical flow has changed between frames in order to detect the alarm situation. The more magnitude is the more possibility of detecting

alarm situation. Less magnitude means actually the crowd is walking in slow pace. After getting the magnitude part of the flow object, we threshold the magnitude with >0.3 in order to get rid of noise. This step is important since there may some areas on the frame in which flow change estimated but there is not abnormality in the crowd.

Then, we create a 3D matrix whose z-axis has length of 30 and represents each frame's magnitude image. We use the matrix that has 30 frame in its z-axis because in the source paper it is said that the dataset is captured by cameras that are able to take 30 frames per second. This means that we hold 1 second of the video sequence in our 3D matrix.

Having the magnitude matrix, we then get the average of the frames to get the activity map. The activity map will have color space from black to white and white color means 'there is motion in that part of the mean image'. The black means no change happened at that pixel. There are also gray pixels shows less amount of movement. In addition, to get rid of salt and pepper noise in the activity map, we apply median filtering on it. Having the sequence of activity maps, we can now detect the excess amount of motion between them.

For this step, as proposed in the source paper also, we subtract each current activity map from the last one to get a difference matrix. This matrix is called 'Temporal Occupancy Variation (TOV)' in our source paper. TOV frame only shows the different pixels between the consecutive activity maps which means we can now 'see the motion'. This frame is called as dif in our code.At that point, the source paper suggests two ways for detecting the 'alarm' situation. If one of them shows that there is an alarm situation then no matter if the other parameters shows the alarm or not, the abnormality becomes detected. One of these two ways is counting the white pixels in the difference frame. This will show us how great our difference is. We call this TOVDifference.

Secondly, entropy can also help us to estimate how big our motion in our difference frame. This is also declared in the source paper; thus, we use entropy(dif) function of MATLAB to get the entropy information of the frame. This variable is called as imgEntropy in our code. You can see how the entropy of an image is calculated below, where Pi is equal to the frequency of pixels having a specific intensity value, which varies from 0 to 255. In our activity map, increasing motion increases entropy, since normally the points all would be black when no motion happens.

$$H = -\sum_{i} p_i(\log_2 p_i)$$

Lastly, we applied exponential smoothing on both TOVDifference and imgEntropy to get more reliable results that are supported by the previous values of these two values. This step will prevent our code to detect any alarm in frames in which no alarm situation occurs but the instantaneous values of TOVDifference and Entropy exceeds the threshold somehow. The outputs of the exponential smoothing are called as TOVLevel and EntropyLevel in our code. As last step, we check if they exceed their particular threshold or not. If one of them exceeds the threshold, then it means that there is an alarm situation. In other words, the crowd suddenly starts to run. The threshold regions, and hence the optimal thresholds are unique to each dataset and they do not work for the other datasets since the environment conditions such as illumination and the camera distances are different for each dataset. Since our code works online, we can immediately insert 'ALARM' string into the video that is playing. At the end of each video, we can see the instantaneous changes on TOVLevel and EntropyLevel. The discussion on these graphs will be given in the next section. Below, you can see snapshots from our pipeline:

For the input dataset, we used UMN dataset [7]. The dataset includes three different scenes, named Lawn, Indoor and Plaza, and several videos for each type of scene. Below, you can see snapshots from our pipeline for each scene. The discussion on resulting graphs will be given in the next section.

UMN Dataset Lawn Sequence

Normal Crowd Behavior

Abnormal Crowd Behavior

Figure 1: RGB Images

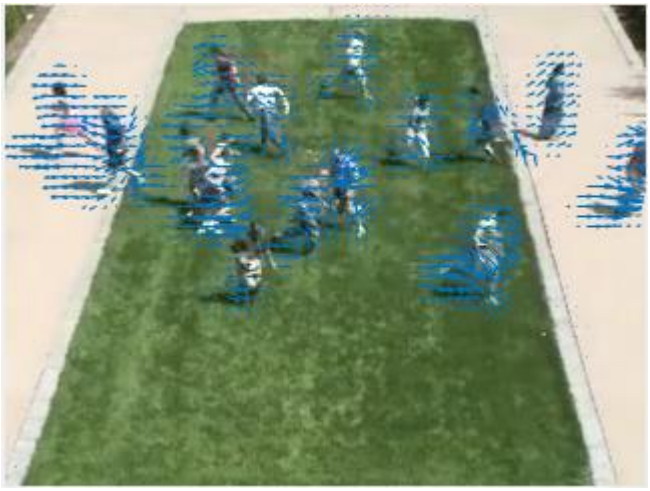
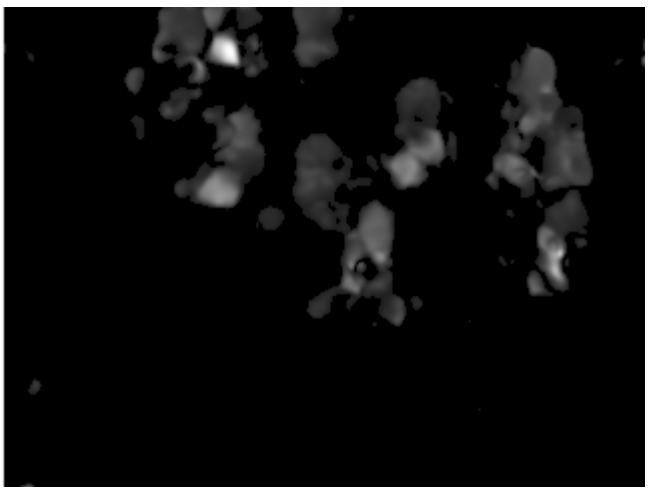


Figure 2: Farneback Optical Flow output



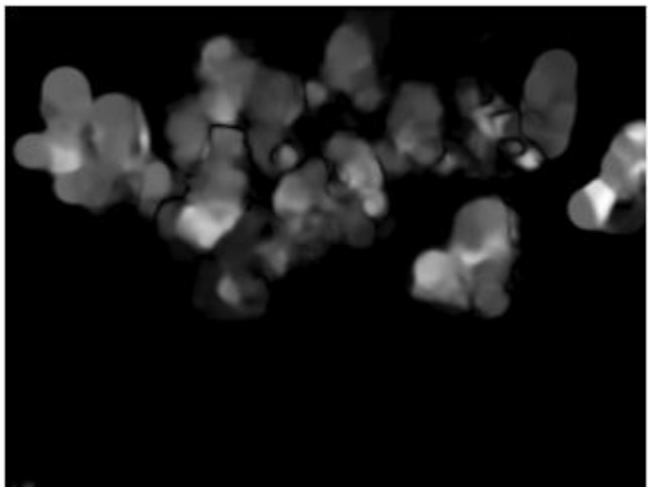


Figure 3 : Magnitude of the Optical Flow

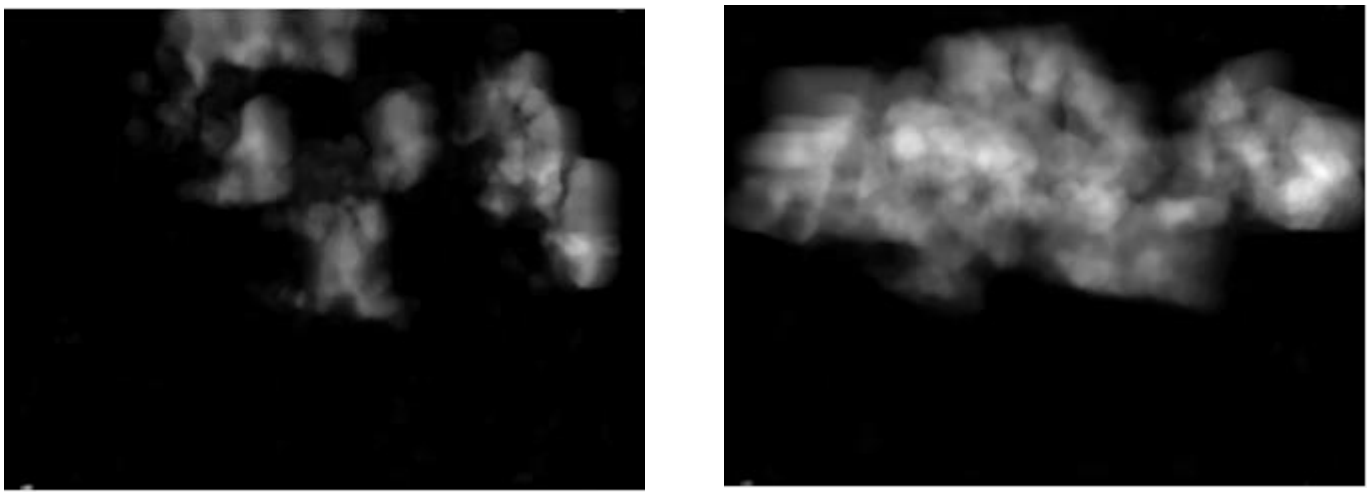


Figure 4: Activity Maps

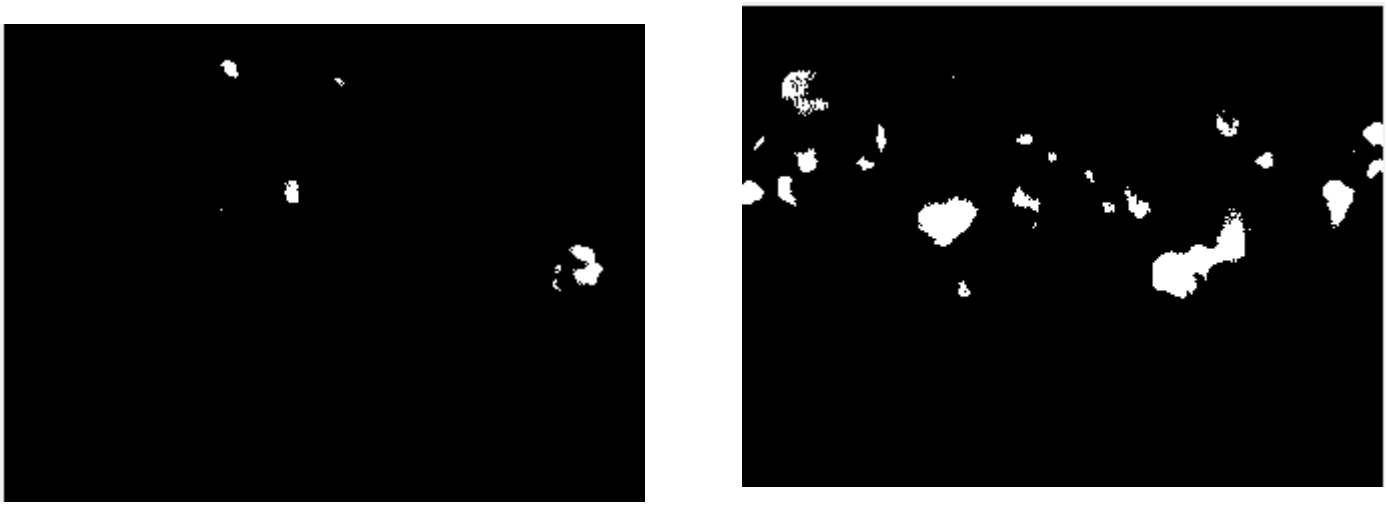


Figure 5: Difference of Consecutive Activity Maps

Figure 6: Output of our code showing Alarm detection

UMN Dataset Indoor Sequence

Normal Crowd Behavior

Abnormal Crowd Behavior

Figure 7: RGB Images

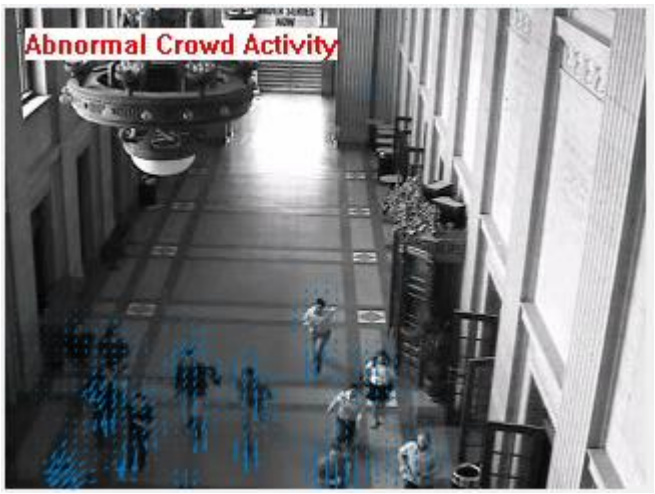


Figure 8: Farneback Optical Flow output

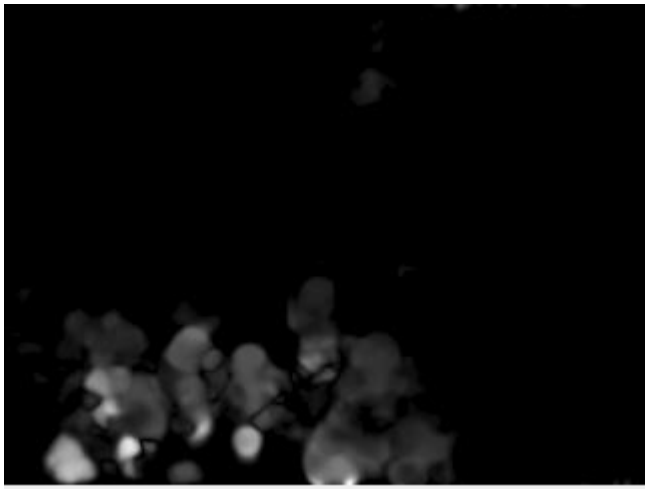


Figure 9: Magnitude of the Optical Flow

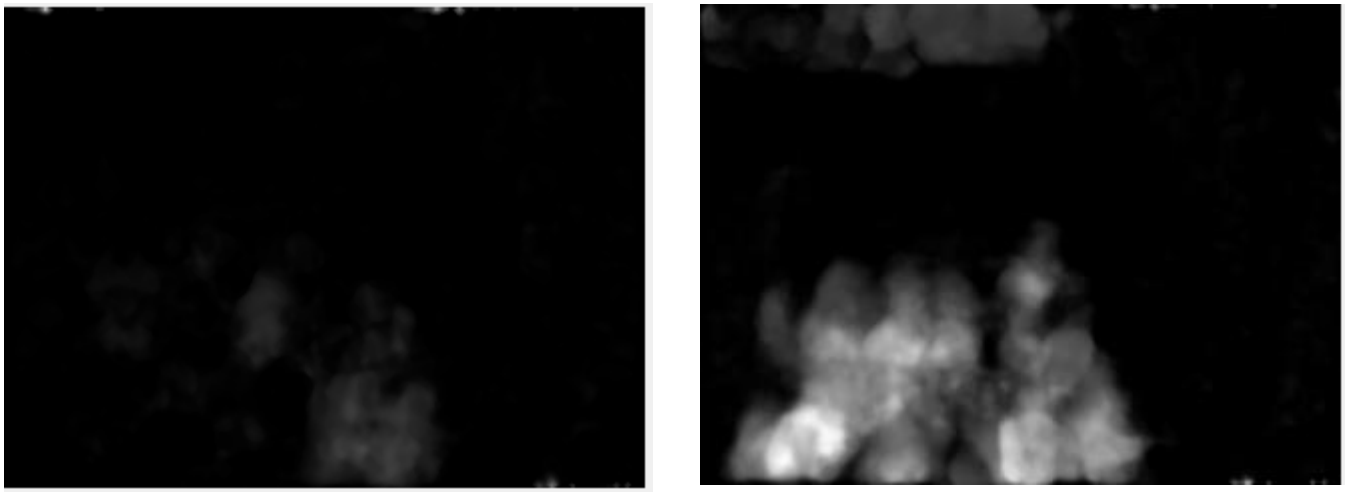


Figure 10: Activity Maps

Figure 11: Difference of Consecutive Activity Maps

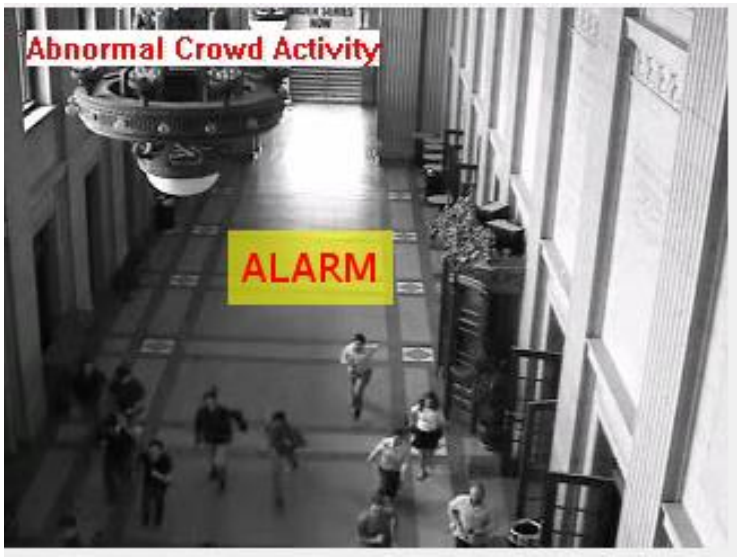


Figure 12: Output of our code showing Alarm detection

UMN Dataset Plaza Sequence

Normal Crowd Behavior

Abnormal Crowd Behavior

Figure 13: RGB Images

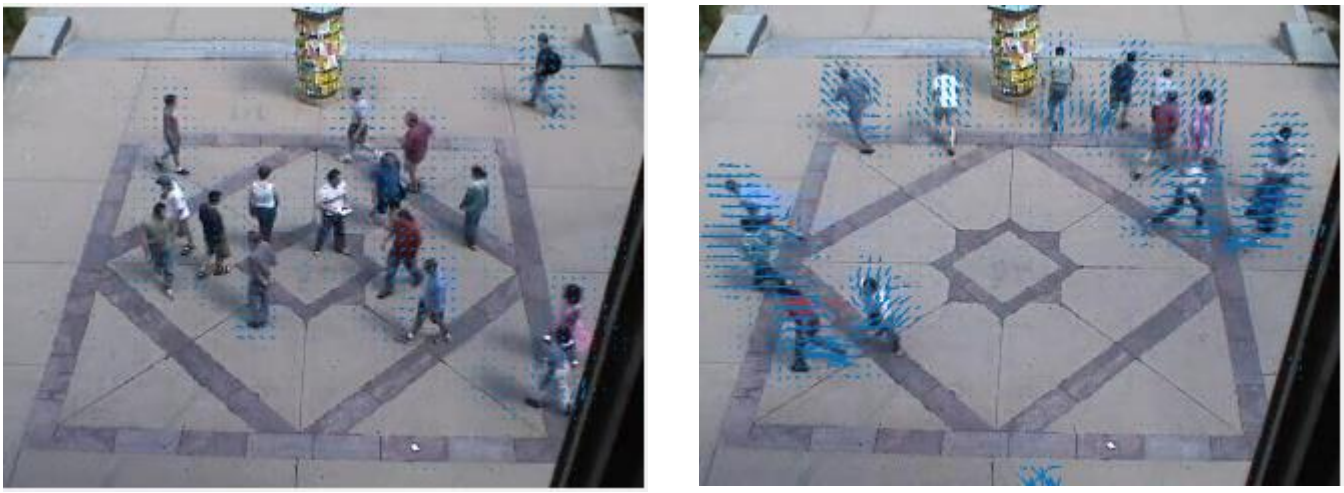


Figure 14: Farneback Optical Flow output

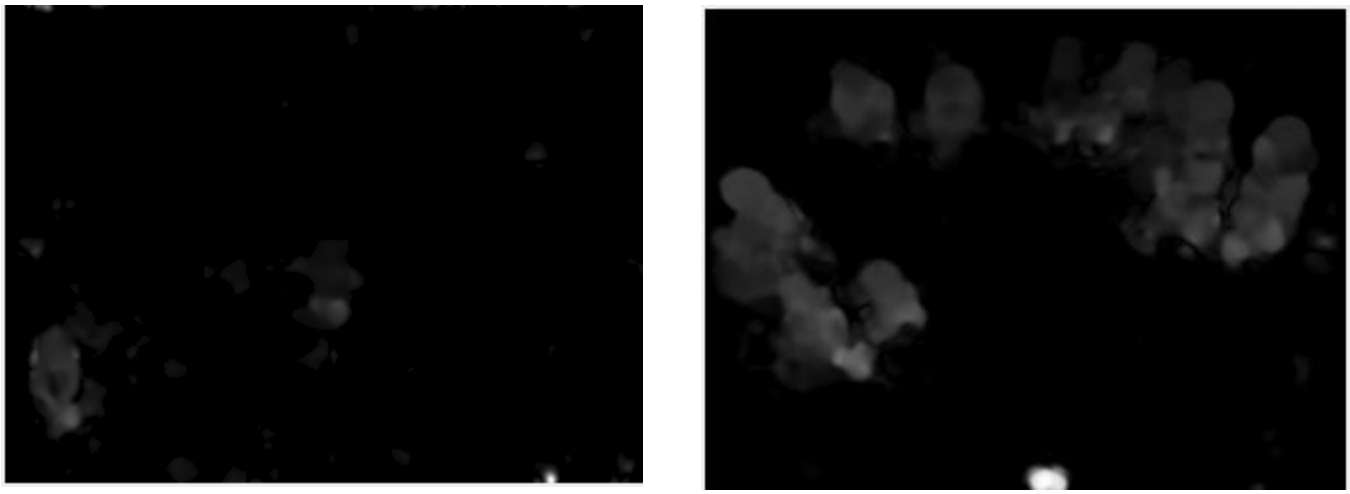


Figure 15: Magnitude of the Optical Flow

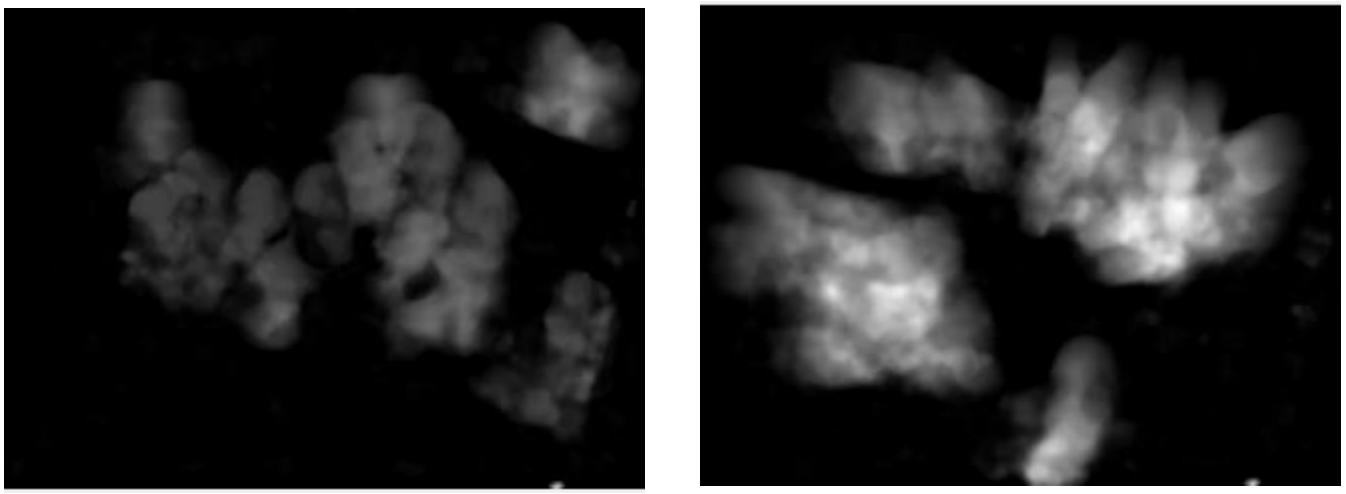


Figure 16: Activity Maps

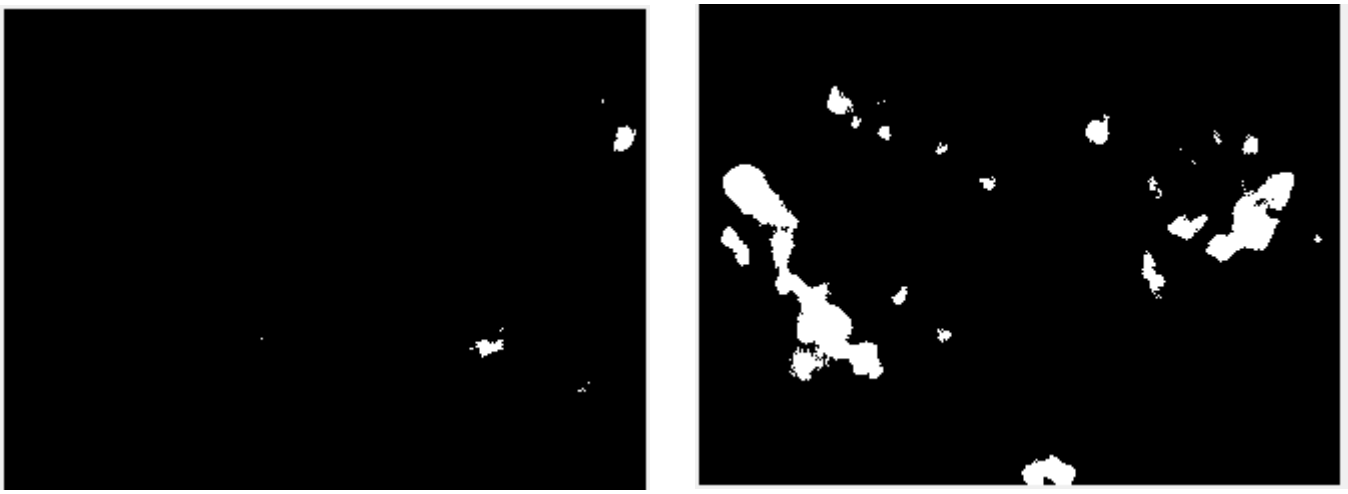


Figure 17: Difference of Consecutive Activity Maps

Figure 18: Output of our code showing Alarm detection

Overall comment for the results of the code attached above: The first row (Figure 13) of each sequence shows the raw image frames for both normal and abnormal scenes. The second row (for example Figure 14) shows the optical flow between the frames on the first row and the former frames of them. As you can see, the blue arrow shows the magnitude and the angle of the optical flow. The longer blue arrow is higher amount of motion. Thus, in the alarm scenes all the blue arrows appear to be big in magnitude while in normal scenes they seem to be really small in magnitude. The third row of the output (for example Figure 15) shows only the magnitude of flow, which is our starting point of estimating alarm situation as explained in the code description. Fourth row (for example Figure 16) shows the activity maps for 30 frame sequence. As could be seen, there is huge amount of white region in the abnormal one in compare to the normal one since there is more motion difference in the abnormal event. If we get the difference between activity maps, then there will be alarm situation on the exact frame in which the crowd stop having slow pace and start running. This will create huge amount of white pixel frame since normal pace has not many whites and the abnormal one has a lot of white pixels. Then it is enough for us to estimate alarm situation after some thresholding. The output of the code could be seen in the Figure 18.

3. Analysis and Discussion of Results

Below are the samples of TOV and entropy metrics, calculated once for each scene type (Lawn-Indoor-Plaza). The x-axis corresponds to the frame number, while y-axis means TOV between two consecutive activity maps, and the entropy of the difference between two activity maps, respectively. Since ground truth data is not available, it is observed by us on each video, and used in our calculations. The red lines indicate where alarming situation happened, meaning when people started to escape. Alarming situation lasts until the video ends for each video. In the following images, one can find that both metrics yield very similar outputs, since their meaning and calculation are very similar as well. They are both indicators of change in flows happening over time. One can easily notice that after the anomaly starts, both metrics show rapid increase, creating a pulse that lasts until the situation ends. This shows that both our metrics show very promising behaviors in order to classify frames as normal or abnormal.

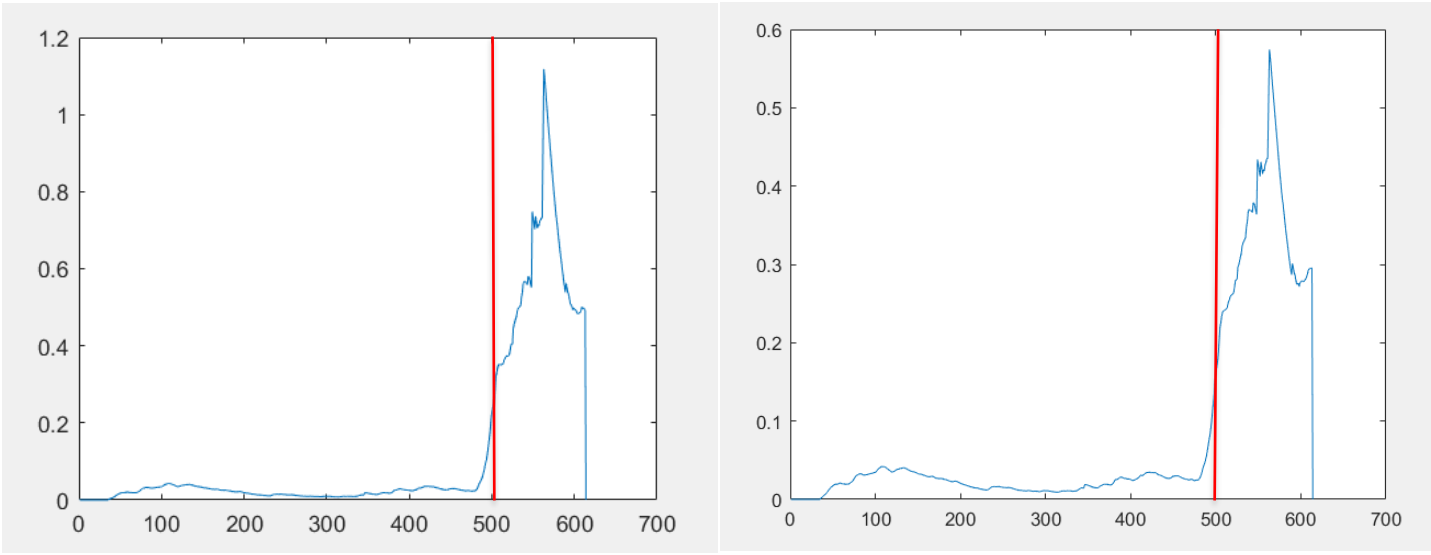


Figure 19: TOV and Entropy Plots for “Lawn” scene, respectively.

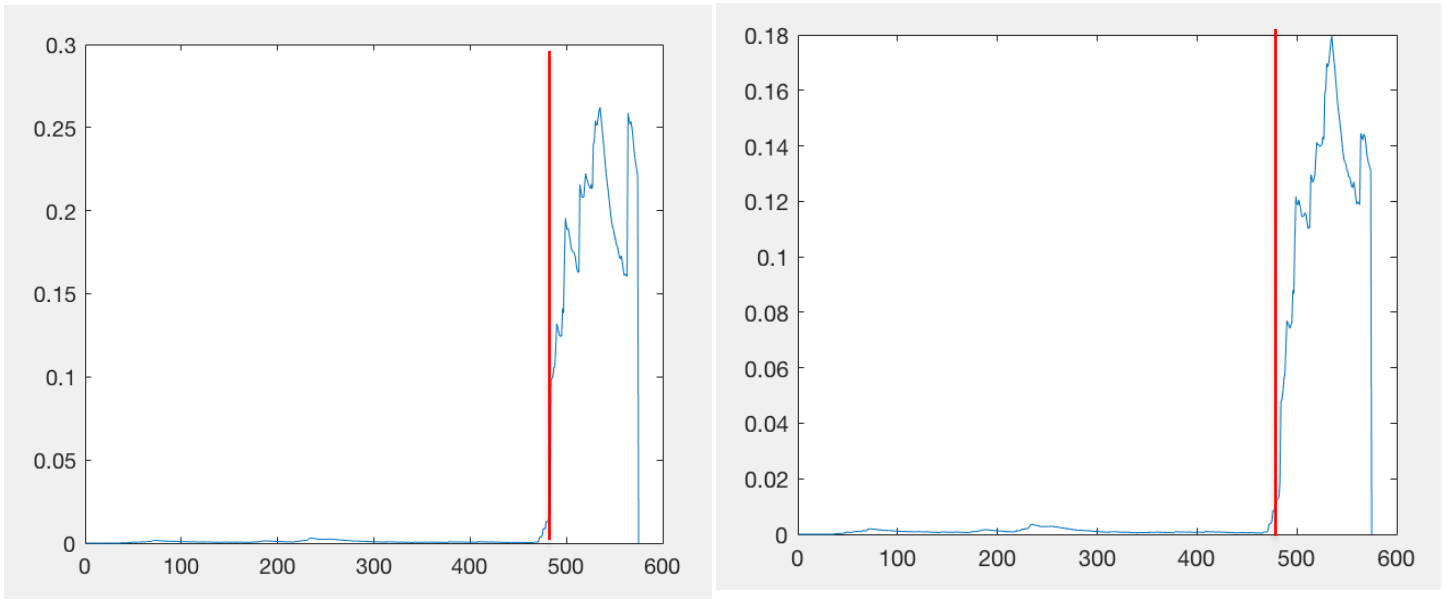


Figure 20: TOV and Entropy Plots for “Indoor” scene, respectively.

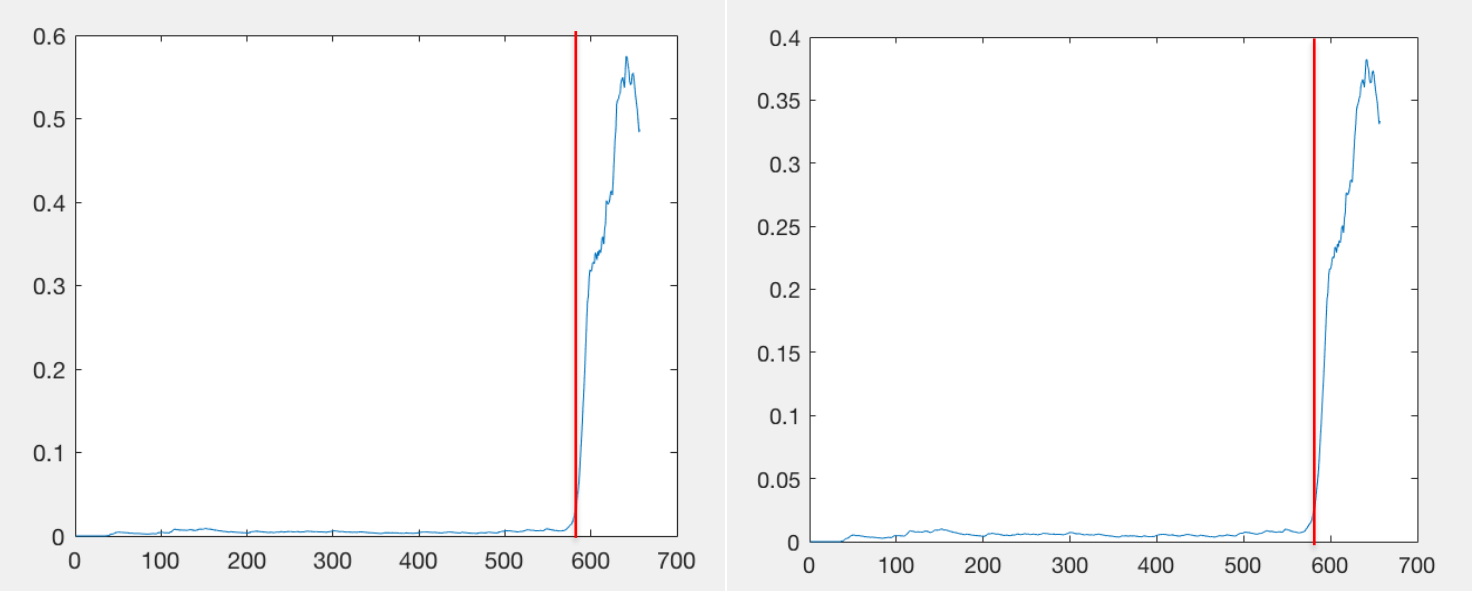


Figure 21: TOV and Entropy Plots for “Plaza” scene, respectively

With human eye, it is quite easy to see where anomaly occurs according to these metrics. However, the important thing here is to choose the threshold values to classify each frame as normal or abnormal. Firstly, since the elevations, angles and distances for the surveillance camera in each scene differs from one another, the feasible regions for thresholds will not be the same. This means that best threshold values will differ for each scene group. To overcome this issue, the minimum and maximum values for both metrics can be learned as a function of elevation of viewing angle, and then this learner can be automatically introduced to new surveillances cameras that are being installed.

Even though these parameters are found with cross-validation techniques, it is not enough for accurate evaluation. One of the most accurate performance indicators for such classifiers is Receiver Operating Characteristics (ROC) curves. ROC curves show the relation between sensitivity and specificity, which are good measures for overall performance. Hence, we worked on three different threshold intervals for each scene type to plot Receiver Operating Characteristics (ROC) curves. The thresholds for TOV and Entropy metrics are changed simultaneously, considering the scale between them. You can find the ROC plots for each type of scene (**averaged** over video sequences for the same types) below:

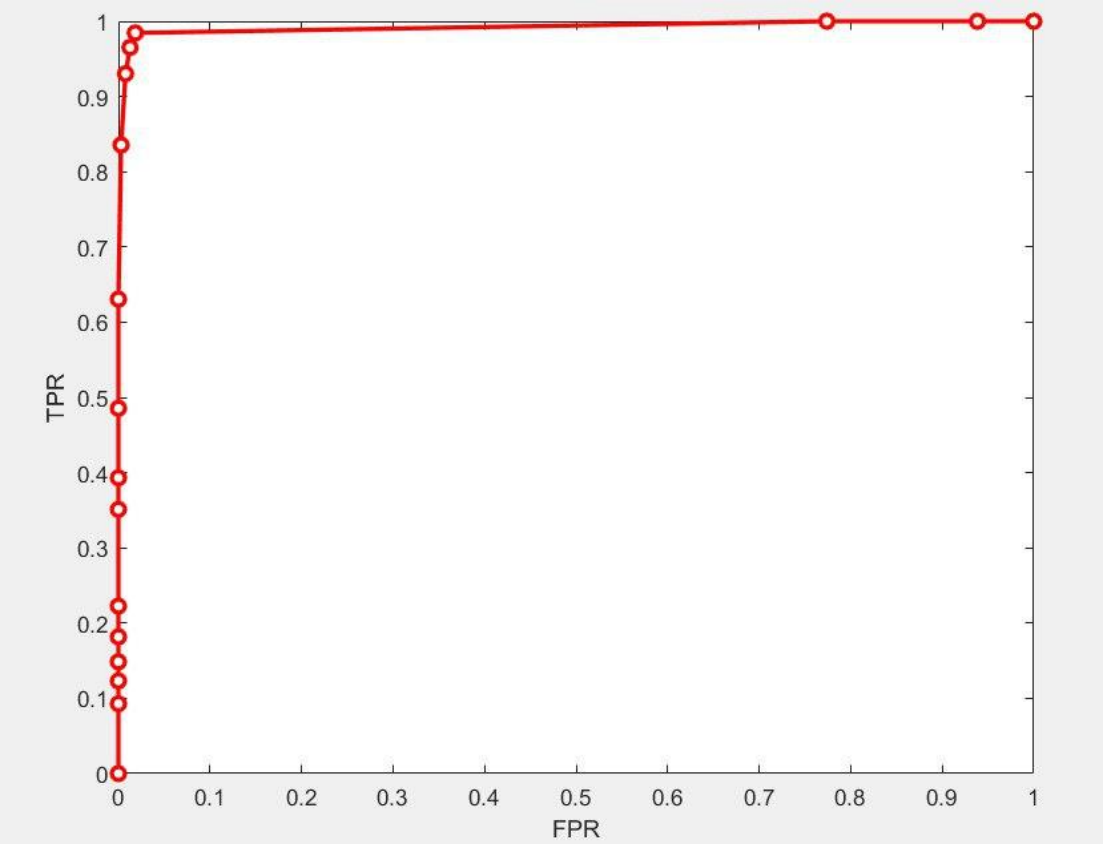


Figure 22: ROC Plot for “Lawn” Scene.

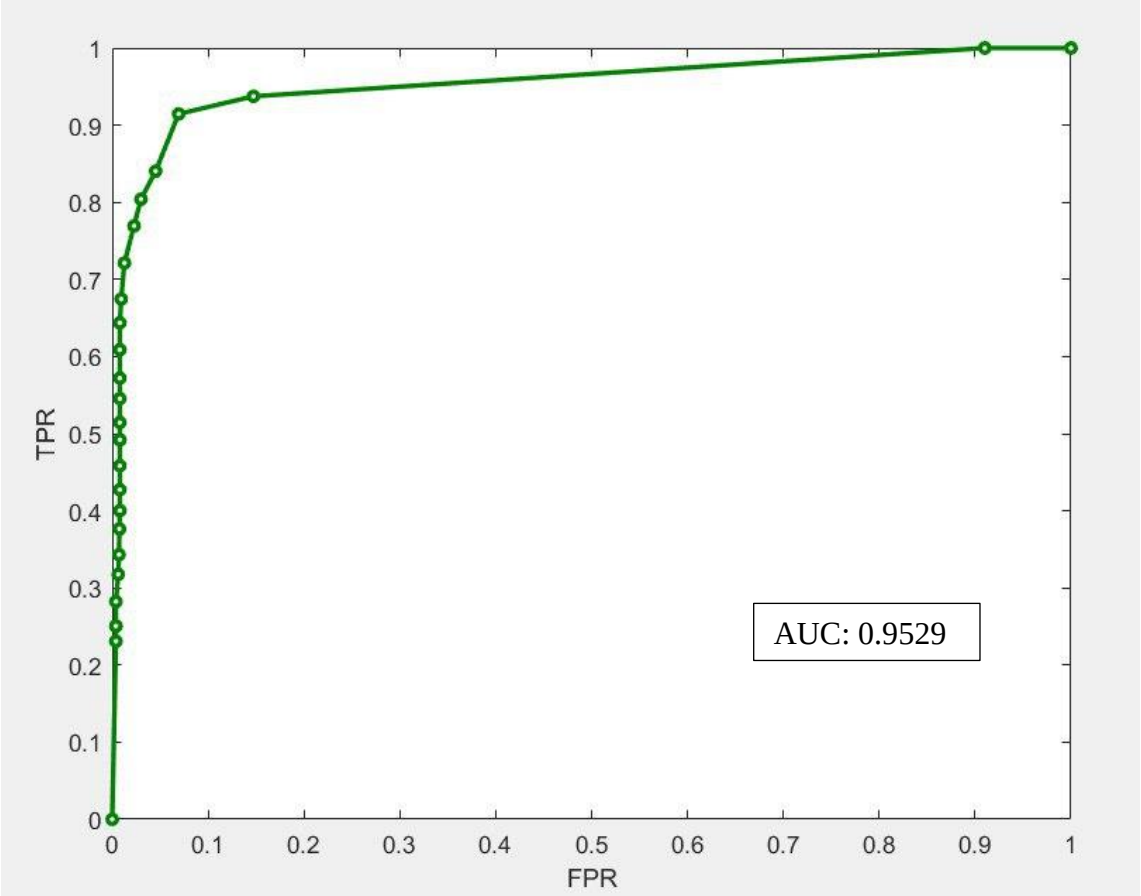


Figure 23: ROC Plot for “Indoor” Scene.

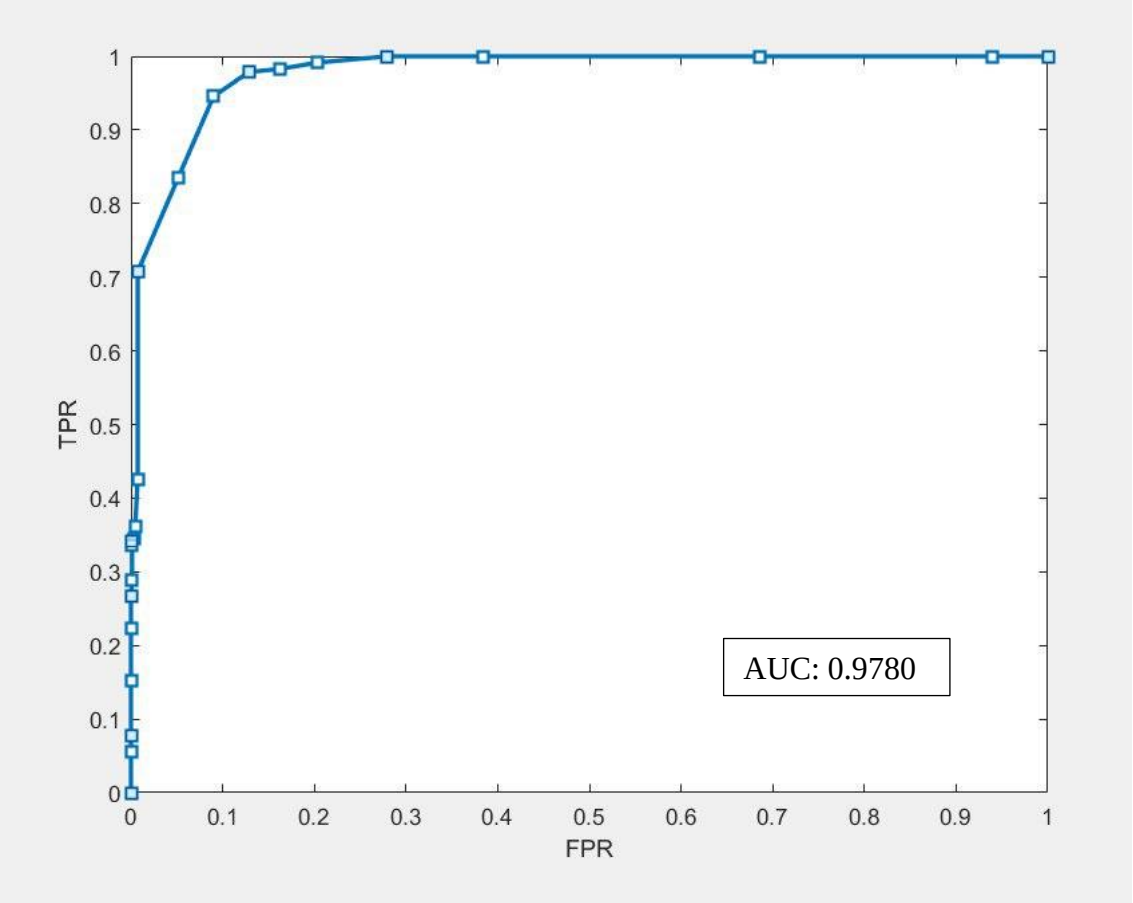


Figure 24: ROC Plot for “Plaza” Scene.

The area under curve (AUC) is a performance indicator for roc plots, as shown on the figures. All performances are above 95% which is promising for anomaly detection. Different threshold values will provide different specificity and sensitivity. If it is more important to detect every abnormal event, then sensitivity (TPR) should be very high. If it is more of use to not detect much false alarms, then we need to set a relatively higher threshold value. Then the specificity should be increased, which is equivalent to (1 - FPR).

3.1 Limitations and Challenges

The most important property of our pipeline is that it needs to be working in real-time since it is meant to be implemented on surveillance cameras, which captures real-time visuals. Then, the processing rate of the algorithm should be faster than the input sequence itself. Our algorithm performs at a processing rate over 30 frames per second, which makes it online given the input as 30 FPS as well. A frame rate of 30 is quite satisfying for human eye. This performance is possible since we have two features for each frame -TOV and Entropy- and they can be updated in real time.

Another property of our algorithm is that it has no training data, and it is not possible to grab the movement behavior of different scenes with a training data. The movements of the people across scenes, across cities will be unique, and this is a limitation in the sense that the algorithm should only work on current input. It also might have a limited memory to determine the crowd behavior. In conclusion, the optical flow data is not feasible to introduce in any type of learner; and given the nature of our problem, it could be extremely challenging to employ training data. There is some research which uses neural networks as the classifiers for the detection; however, this eliminates the real-time property of the detector.

One of the possible problems about our algorithm is whether our code should work on the places where huge amount of motion change is perceived as normal. One of the possible examples for this is a football match. If we put a camera on the field, nearly in all the frames, the algorithm should give alarm since every player runs. But there is no alarm situation. Thus, the algorithm should understand from the behavior of the players whether they are running for the ball or because of an alarm situation. The explanation is as follows:

Our algorithm works based on the optical flow magnitudes. This means that wherever a huge amount of motion between consecutive frames estimated, the alarm situation should appear. In case of the football match, the proposed question was whether we can understand the players are running for the ball or running because of an alarm situation. Both situations include huge amount of motion change, that means our code will detect the alarm situation even if the players would run for the ball. Separating these two concepts require further and more complex work. To get more information from images, first, we should detect more feature points than we detect now. The feature space of our algorithm is just based on TOV and Entropy, thus it allows our code to run online. Since for online calculation, we should process the input frames fast. However, if the feature space increases for detecting more information to grab the behavior, the speed will reduce, and it will not be online anymore. This is a situation we do not seek since we want to detect alarm situations as fast as possible. In addition to that, to have more elaborate features, our cameras should have high resolution and since surveillance cameras are lack of high resolution in general, we cannot inspect the behavior of the people from the frames we get from surveillance cameras easily.

The other important issue is threshold value decision for different cameras that are located in different places. Threshold values are the ones that we completely rely on for detecting alarm situations. Picking them carefully will give us best results for a specific camera. However, each camera is located on different elevations with different viewing angles. In addition, characteristics of each place is unique and picking threshold values could

be done based on the context. Last of all, the illumination of the place should not be very high or very low to detect features points. Let's explain all these different parameters for detecting threshold values.

High elevation will result small amount of changes in the pixels since the area covered is very wide. Thus, even if an alarm situation, the flow of the feature points may be that much high in compare to the low elevation cameras. Thus, for high elevation we should pick threshold values smaller. On the other hand, in low elevations, algorithm can detect that a feature point has high amount of motion change since the scope is narrow. Thus, in low elevation cameras, the threshold values should be higher.

Viewing angle is important because for our code to detect optical flow the body of the person should be appearing mostly whole in order to estimate optical flow. Thus, for example from a bird's eye camera, the camera can only capture the heads of the people and this might be misleading since in the frames we do not track the body but instead small black balls. This is error prone since we get away from the idea of getting optical flow according the body movement of the people. On the other hand, if our camera is set so that it is parallel to ground, the code will not capture all the bodies since some bodies will appear in front of the other ones. This results in error also. The healthiest way of estimating the optical flow is having a normal elevation with 45 degree viewing angle. Thus, we can separate all the bodies separately and code could generate optical flow out of them. This is the case for our datasets.

Another parameter is the context of the place where we set camera. If we expect very slow amount of motion from that place, we should lower the threshold values. An example for this could be a calm street where a few people appear on the camera in every frame. That means, the algorithm will generate very small amount of motion change. Even if an alarm situation happens, the motion change in the frames could be as high as a normal situation in a crowded square. Thus, we should select the parameters low for this calm street. On the other hand, if we put this camera on a crowd square, we expect that every frame will consist high amount of motion even in the normal situations. Thus, our threshold values should be high enough to prevent giving alarm in normal situations.

Overall, these parameters are very important for our code to work. Thus, we should preset our threshold values for each unique camera to get best results in online.

4. Concluding Remarks

In this project, we aimed to calculate metrics to detect abnormalities in given image sequences, taken from surveillance cameras. This is importance since safety concerns increase in today's world day by day. What this algorithm tries to achieve its to automate the event observing process for the surveillance cameras as accurate as possible. Our input are some surveillance camera records, which is in parallel with the aim of this project. We followed our source paper for the implementation. After calculating optical flow, the activity map is created to be able to capture the changes in optical flows in a given time window. Using this activity map, the entropy -hence the surprise factor- of the situation and temporal occupancy variation is calculated. Both metrics show very similar values for normal and abnormal scenes. Using the changes in entropy and temporal occupancy variation, each video frame is classified as normal or abnormal. ROC plots show us that the classification performance is very pleasing.

5. References

[1] A. Pennisi, D. Bloisi, L. Iocchi, “Online real-time crowd behavior detection in video sequences, In Computer Vision and Image Understanding”, Volume 144, 2016, Pages 166-176, ISSN 1077-3142, https://doi.org/10.1016/j.cviu.2015.09.010.

[2] R. Mehran, A. Oyama, and M. Shah, “Abnormal Crowd Behavior Detection Using Social Force Model,” Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2009.

[3] L. Kratz and K. Nishino, “Anomaly Detection in Extremely Crowded Scenes Using Spatio-Temporal Motion Pattern Models,” Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2009.

[4] S. Wu, B. Moore, and M. Shah, “Chaotic Invariants of Lagrangian Particle Trajectories for Anomaly Detection in Crowded Scenes,” Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2010.

[5] W. Li, V. Mahadevan, and N. Vasconcelos. Anomaly detection and localization in crowded scenes. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 36(1):18–32, 2014.

[6] Shifu Zhou, Wei Shen, Dan Zeng, Mei Fang, Yuanwang Wei, Zhijiang Zhang, Spatial–temporal convolutional neural networks for anomaly detection and localization in crowded scenes, In Signal Processing: Image Communication, Volume 47, 2016, Pages 358-368, ISSN 0923-5965, https://doi.org/10.1016/j.image.2016.06.007.

[7] "Monitoring Human Activity - Detection of Events", Mha.cs.umn.edu, 2018. [Online]. Available: http://mha.cs.umn.edu/proj_events.shtml. [Accessed: 08- Jan- 2018].