Real Time Analysis of Crowd Behaviour for Automatic and Accurate Surveillance

E Padmalatha[1], Karedla Anantha Sashi Sekhar[2], Dasarada Ram Reddy Mudiam[3]
Dept of Computer Science and Engineering
Chaitanya Bharathi Institute of Technology
Gandipet, Hyderabad 500075

***Abstract*—Surveillance in this modern era is a necessity. Creating an alert in case of emergencies and disturbances is of very much importance. As the number of simultaneous camera feeds increase, burden on human supervisor also increases. The proposed system is a way to aid human supervisor in the surveillance job. Creating alerts in real time will help responding quickly to crucial situations. With this in mind, we propose the following things: (1) Generation of ViF (Violent Flow Descriptors) as high-level features in real time. (2) Using generated ViF's of a Video Dataset for training a neural net and testing its accuracy. (3) Developing a system that can detect the signs of disturbance among the crowd in real time and can learn from the decisions it makes.**

***Keywords*—Real time surveillance; violent flow descriptors; neural network**

I. Introduction

Cost of surveillance equipment in this digital era is minimal. In the view of public safety, CCTV cameras are installed in crowded and densely populated areas. The footage from CCTV cameras are continuously monitored by humans in order to respond in case of emergencies. This is a routine and tedious job for a human to continuously pay attention to multiple screens. Surveillance by humans is inefficient as it is limited to human capacity and may not be error free. If computers are replaced by humans to perform surveillance and generate alerts, it may aid the humans to respond quickly to the alerts. If we consider the amount of video footage generated simultaneously, we need a solution which can handle input at this scale.

The research done until now focuses on increasing the accuracy but makes a significant trade off with speed. We here focus on a scalable and efficient algorithm which focuses on both accuracy and generating alerts in real-time.

Generation of ViF [1] is already been experimented previously. So as to generate ViF, there is a detailed process that has to be followed. Starting with the videos, they have standard aspect ratio of 3:4 and are of very low quality. As the crowd behavior is completely random, detecting breakouts in the crowd becomes a real challenge. Also the content of the video is considered to be originated from a CCTV camera hence any other source of information such as subtitles and audio cannot be used. Continuous surveillance system is of much importance and very less attention is given to it.

In this proposed system, we try to implement an algorithm which accurately detects violence in real-time. Through this algorithm we try to obtain safer surroundings and have a quick response time to violent incidents.

II. Previous Work

A. Optical Flow

Optical Flow is the core part of Violence Detection. Optical Flow is the relative motion between two image frames which are taken at times t and t+Δt at every pixel position. Methods for determination of Optical Flow can be listed as Phase correlation [8], Block-based method [9], Differential methods and Discrete Optimization methods. The most commonly used methods are Lucas Kanade and Horn-schunck optical flow methods [6], which come under Differential methods based on solving first order derivative. We used C Liu's [2] optical flow algorithm for our task which will be used to further obtain Flow Vector Magnitude. Suppose V_x and V_y are the velocities of a pixel along x and y axis obtained through optical flow algorithm, then the flow vector magnitude can be obtained as, $m_t = \sqrt{V_{x,t}^2 + V_{y,t}^2}$. C Liu's Optical flow algorithm was originally written in C language and mex files were written for compatibility with MATLAB. We used bob package [3] for using that particular algorithm in Python.

B. Violent Flow Descriptors

ViF (Violent Flow Descriptors) [1] have been used previously to obtain global level features of a video. After obtaining the flow vector magnitude (m), we calculate the binary vector. This binary vector is calculated for each pixel which reflects the change in magnitude.

$$b_{x,y,t} = \begin{cases} 1, & if \quad |m_{x,y,t+1} - m_{x,y,t}| \geq \theta \\ 0 & otherwise \end{cases} \tag{1}$$

After obtaining the binary vector for each frame, we add binary vectors which are obtained for all the frames and normalize the value with the number of frames taken under consideration.

$$\bar{b}_{x,y} = \frac{1}{T} \sum_t b_{x,y,t} \tag{2}$$

This $\bar{b}$ generated is divided into M*N non-overlapping cells and collecting magnitude changes of each cell separately. These magnitude changes are then represented by a fixed size histogram. These M*N histograms are further concatenated to obtain a single descriptor vector which is known as the ViF.

III. Methodology

Doing real time automatic surveillance on CCTV footage has many challenges and limitations. There are two assumptions which we make, the first assumption is to keep the camera away from the area under surveillance, there has to be standard distance between the CCTV camera and the area to be monitored. The main challenge we face is to keep the processing in terms of real-time, which means all the processing has to be done in less than 1/25th of a second. Our system should have the ability to handle multiple video sources at a time. This has been achieved by continuously accepting frames through multiprocessing using threads. First part is we calculate the Optical flow which is the most time consuming of all the processes. Then we use the calculated optical flow to obtain flow vector magnitude. Next we generate ViF which are further used for training and classification.

A. Algorithm

The main part of building the system is to have a well built algorithm. First the video is preprocessed, then we calculate the optical flow which is most time consuming. Feature Extraction is done next which is used for training a multilayer perceptron [5]. So as to make the system scalable for multiple video sources, we have embedded threading and multi-processing. The built algorithm is robust and can handle faulty video sources.

B. Global Feature Extraction

1) Video Preprocessing: Video coming from source is preprocessed. Considering the video aspect ratio as 3:4. The surveillance footage is consider to be standard definition (scale = 240:320). The input frames coming in are resized to 75:100 size and then converted to gray-scale. The length and breadth of the videos are almost reduced by one-third. For video processing OpenCV [7] package has been used.

2) Optical Flow: Ce. Liu [2] optical flow algorithm has been used to calculate the optical flow. This algorithm has been used particularly since it is highly efficient and robust. It returns three values, v_x(velocity vector along x-axis), v_y(velocity vector along y-axis) and w(wrap). The vectors are in the same shape of the resized frame width and height.

3) Violent Flow Descriptors: Violent Flow Descriptor [1](Global Features) Algorithm is being used which has been already been implemented previously in MATLAB. So as to increase the scalability of the algorithm, system has been implemented using basic multiprocessing and threads. Violent Flow Descriptors use flow vector magnitude's output. Histogram equalization on normalized binary vector for the whole set of frames under consideration is performed to obtain a single feature vector. The ViF's obtained are then used for training and classification purposes. If the number of bins are fixed as 21(0.0 to 1.0, interval of 0.05) and consider both M, N as 4, then for a standard definition video of scale 240:320, 336 features are obtained exactly(21 * 4 * 4). Values of these 336 features range from 0.0 to 1.0.

C. Neural Network

1) Structure: For the given dataset once violent flow descriptors are generated, neural net training using these features is done. Built four layered neural net, one input layer, two dense layers and one output layer. Input layer accepts 350 inputs and gives 336 outputs. Middle dense layers accept 336 inputs and give 336 outputs. Output layer accepts 336 inputs and gives 1 output. For input a nd dense layers ReLU (Rectifier Linear Unit) activation function is used and for the output layer Sigmoid function is being used.

2) Training: For building neural net Keras [10] and Tensorflow are being used. Initially the data needs to be formatted so that it fits into input layer of Neural Net. The Violent Flow Features generated for each video will be in the format of numpy array of dimensions 336*1. Array is reshaped into 1*336 dimensions, so that the features of single video fit into the input layer of neural net as one data tuple. Further, concatenation of reshaped feature array of each video into a single array is done so that the final input to the neural net will be an array of dimensions 246*336 as there are 246 videos (violent and non-violent) in the dataset. On the other hand, pre known outputs (0(non-violent) or 1(violent)) of each video are stored in an array of dimensions 246*1 to train neural net and to calculate accuracy. After data is ready we can proceed to build the neural net into the structure mentioned above. Using Keras Sequential neural net with dense layers can be built.

A model is compiled for which we must specify a logarithmic loss function which evaluates a set of weights and also an optimizer to set learning rate. Keras has a logarithmic loss function for binary classification problem defined as binary-crossentropy and Adam Optimizer which is an efficient optimizer of choice. Number of epochs for which the training must be carried out, batch size(number of instances evaluated to perform weight changes) and the input data for training the model are provided as parameters. Trained model is stored into a file of hd5 format using hdpy python package.

3) Violence Detection: This phase involves detection of disturbance or violence in live crowd surveillance videos in real time. The input surveillance video is preprocessed and Violent Flow Descriptors are generated dynamically in Real Time. For each second of video, features are extracted and are given as input to the trained model for classification and violence detection. If some disturbance or violence is detected, it will be reported as an alert stating that it is violence along with the time it has occurred within a second of occurrence.

4) Feedback: In the Violence Detection phase, as the real time surveillance takes place, the features generated for every second are tested against trained model for classification and violence detection. Those features, along with their actual output(provided by human) generated by the trained model are given as feedback to the model. This allows continuous training of neural net model which helps to increase the accuracy of classification and also faster detection of violence.

D. Extraction of Interesting Features

AdaBoost is an ensemble of weak classifiers. AdaBoost is an algorithm which could tell us the important set of features that help us classify our features. For this the Feature Selection Algorithm through AdaBoost [4] is used. Once the features are arranged in increasing order of the error rates, we can obtain the features among total 336 features which are highly

efficient in classifying videos. The weak classifiers used here are decision stumps (decision tree of height 1).

IV. Implementation

In the below subsections we provide the implementation details and the outputs analysis. Clear analysis of the system will be done in the next section.

A. Continuous Surveillance

We used two sets of configurations for calculating processing speeds of the system. First Configuration consists of 4GB RAM, Intel Centrino Processor with Ubuntu OS. Second Configuration consists of 8GB RAM, Intel i5 Processor with Debian OS.

For a video which is not initially violent but later on becomes violent, the proposed system is able to detect the exact instance where the video frames go from violent to non-violent. Considering Real-Time CCTV feed, within a second of occurrence of violence our system in able detect the violence and raise an alert.

```
loaded model from disk
FPS is : 30.0
violent  ---   5 seconds , processing time : 5.74629306793
violent  ---   9 seconds , processing time : 9.68529391289
violent  ---   11 seconds , processing time : 11.6434879303
violent  ---   29 seconds , processing time : 28.7351050377
violent  ---   33 seconds , processing time : 32.4793879986
violent  ---   39 seconds , processing time : 38.1013178825
violent  ---   41 seconds , processing time : 40.0718650818
violent  ---   45 seconds , processing time : 43.8183250427
violent  ---   76 seconds , processing time : 70.0854079723
violent  ---   80 seconds , processing time : 73.7055718899
violent  ---   104 seconds , processing time : 94.7628529072
violent  ---   110 seconds , processing time : 100.153498888
violent  ---   124 seconds , processing time : 112.719841957
violent  ---   130 seconds , processing time : 118.084775925
violent  ---   132 seconds , processing time : 119.904102087
violent  ---   134 seconds , processing time : 121.691713095
violent  ---   136 seconds , processing time : 123.467772007
violent  ---   138 seconds , processing time : 125.274302006
violent  ---   140 seconds , processing time : 127.064712048
violent  ---   162 seconds , processing time : 146.576081991
violent  ---   172 seconds , processing time : 155.607837915
violent  ---   174 seconds , processing time : 157.406991005
violent  ---   182 seconds , processing time : 164.669116974
violent  ---   184 seconds , processing time : 166.488634109
violent  ---   186 seconds , processing time : 168.348366022
violent  ---   188 seconds , processing time : 170.149859905
Done
```

Fig. 1. Processing Speeds with Intel Centrino Processor

```
loaded model from disk
FPS is : 30.0
violent  ---   5 seconds , processing time : 4.34860682487
violent  ---   9 seconds , processing time : 7.25317382812
violent  ---   11 seconds , processing time : 8.71798682213
violent  ---   29 seconds , processing time : 21.9189689159
violent  ---   33 seconds , processing time : 24.8225078583
violent  ---   39 seconds , processing time : 29.1507589817
violent  ---   41 seconds , processing time : 30.5916497707
violent  ---   45 seconds , processing time : 33.4507169724
violent  ---   76 seconds , processing time : 55.4075779915
violent  ---   80 seconds , processing time : 58.2983269691
violent  ---   104 seconds , processing time : 75.7260508537
violent  ---   110 seconds , processing time : 80.1277518272
violent  ---   124 seconds , processing time : 90.3564748764
violent  ---   130 seconds , processing time : 94.698748827
violent  ---   132 seconds , processing time : 96.1417148113
violent  ---   134 seconds , processing time : 97.5626308918
violent  ---   136 seconds , processing time : 98.9930028915
violent  ---   138 seconds , processing time : 100.432123899
violent  ---   140 seconds , processing time : 101.866064787
violent  ---   162 seconds , processing time : 117.839699984
violent  ---   172 seconds , processing time : 125.074666977
violent  ---   174 seconds , processing time : 126.504262924
violent  ---   182 seconds , processing time : 132.335903883
violent  ---   184 seconds , processing time : 133.779084921
violent  ---   186 seconds , processing time : 135.228977919
violent  ---   188 seconds , processing time : 136.651611805
Done
```

Fig. 2. Processing Speeds with Intel i5 Processor

Above are two figures showing running time of the system. The total length of video taken under consideration is nearly 200 seconds. Proposed system is able to detect the exact second of violence occurrence i.e where the frames go from non-violent to violent. With Intel centrino processor Fig. 1, the processing is being completed nearly in 180 seconds (20 seconds faster than run time of video). With Intel i5 processor Fig. 2, the processing of entire video is being done nearly in 140 seconds (1 minute faster than run time of video). With each detection of violence outbreak, the corresponding time taken by the system to detect that is being shown.

B. Accuracy

The accuracy obtained by ViF's [1] as global features using a linear SVM is 81.30% for existing system. Proposed system has an accuracy of nearly 85%.

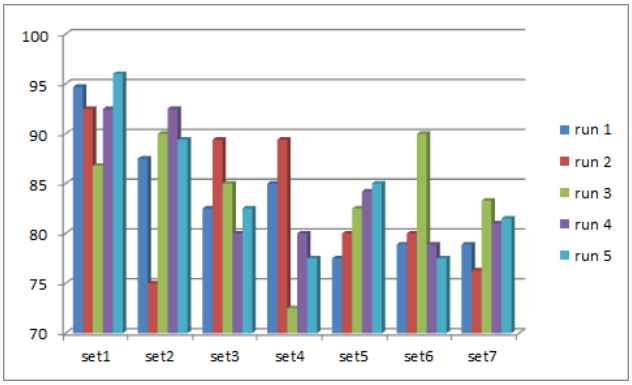

Fig. 3. Bar Plot of Obtained Accuracy Values in N Folds

The bar graph in Fig. 3 shows the result of N-folds cross verification with N=7. There are total 5 runs(execution of n-folds once in a run). In each run we consider 7 heaps in total.

Each heap containing equal number of videos. Among these videos violent and non-violent videos are distributed evenly. Violent and Non Violent videos are placed randomly in heaps. This gives us an idea how robust the proposed system is. The minimum accuracy we obtain for a heap in any run is greater than 70%.

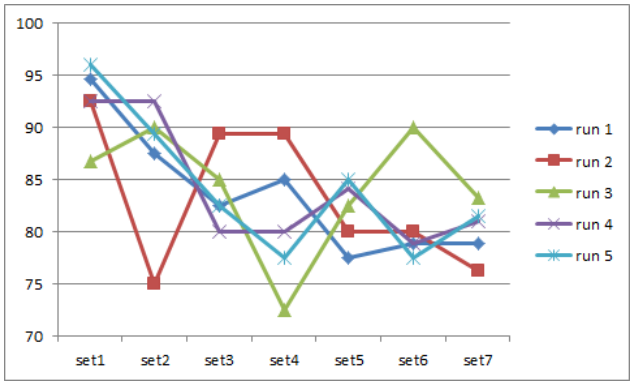

Fig. 4. Line Plot of Obtained Accuracy Values in N Folds

Fig. 4 shows the line graph, it is same as Fig. 4, but gives us the clear picture of accuracies of each set in its corresponding run. Each run has been assigned a different color. From this we can clearly identify minimum and maximum accuracy. The maximum obtained accuracy is nearly 96% and minimum accuracy is of 73%.

```
(bob_py2) dasarada@dasarada:~/Desktop/ASAGS/keras (copy)$ python keras_neural_ne
tworks_training_70_30_random.py
Using TensorFlow backend.
2018-04-13 13:23:35.658603: I tensorflow/core/platform/cpu_feature_guard.cc:137]
 Your CPU supports instructions that this TensorFlow binary was not compiled to
use: SSE4.1 SSE4.2 AVX AVX2 FMA
accuracy is : 0.902777777778
--- 58.6380441189 seconds ---
[[33  2]
 [ 5 32]]
0.902777777778
[76, 8, 111, 75, 50, 74, 25, 56, 55, 83, 102, 129, 9, 17, 85, 101, 35, 117, 107,
 64, 104, 97, 95, 38, 14, 96, 81, 93, 67, 48, 112, 62, 87, 115, 61, 109, 11, 119
, 36, 4, 22, 30, 89, 66, 71, 33, 114, 60, 21, 37, 24, 100, 34, 121, 16, 10, 51,
5, 18, 6, 128, 77, 110, 125, 53, 43, 68, 103, 108, 65, 58, 19, 39, 2, 3, 28, 122
, 127, 84, 59, 13, 90, 105, 54, 98, 29, 94, 124, 26, 1, 46, 42, 76, 8, 111, 75,
50, 74, 25, 56, 55, 83, 102, 129, 9, 17, 85, 101, 35, 117, 107, 64, 104, 97, 95,
 38, 14, 96, 81, 93, 67, 48, 112, 62, 123, 87, 115, 61, 109, 11, 119, 36, 4, 22,
 30, 89, 66, 71, 33, 114, 60, 21, 37, 24, 100, 34, 121, 16, 10, 51, 5, 18, 6, 12
8, 77, 110, 125, 53, 43, 68, 103, 108, 65, 58, 19, 39, 2, 3, 28, 122, 127, 84, 5
9, 13, 90, 105, 54, 98, 29, 94, 124, 26, 1, 46]
[70, 113, 12, 82, 78, 92, 23, 45, 80, 73, 120, 40, 27, 118, 69, 52, 88, 86, 32,
7, 126, 47, 41, 15, 57, 31, 63, 91, 49, 116, 72, 106, 20, 44, 99, 42, 70, 113, 1
2, 82, 78, 92, 23, 45, 80, 73, 120, 79, 40, 27, 118, 69, 52, 88, 86, 32, 7, 126,
 47, 41, 15, 57, 31, 63, 91, 49, 116, 72, 106, 20, 44, 99]
```

Fig. 5. Accuracy Obtained by training with 70% of data and testing with 30% of data

TABLE I. CONFUSION MATRIX FOR 70:30 DATASET

	p	n
P	33	2
N	5	32

The dataset which contains 246 videos is divided in the ratio of 70:30. 70% of the data is used to train the neural net, 30% of the data is used to test the accuracy of generated model. Output in Fig. 5 shows that the accuracy obtained is 90.27% .

As we can see the confusion matrix in Table I, the number of False Negatives are just 2, that means there are only 2 cases in the test set which are actually violent but our system was not able to detect it. Whereas there were 5 cases in which videos were not violent but our system detected some violence. Following are the results obtained:

- Accuracy = TP/(total) = 0.9027
- True Positive Rate = TP/(Positives)= 0.9428
- Precision = TP/(Predicted yes) = 0.8684
- Specificity = TN/(Actual no) = 0.8648
- Misclassification Rate = (FP + FN)/(total) = 0.097

The above accuracy tests were done on a dataset [1] containing 246 videos. Shortest Video is of 1 second and Longest Video is of 6 seconds. These collections of videos have equal number of violent and non-violent videos. This kind of dataset is known as “in the wild” dataset. Videos present in the dataset are of standard CCTV resolution (scale = 240:320) and of similar aspect ratio (3:4).

V. RESULTS AND DISCUSSION

Consider the following scenes obtained through surveillance footage:

Fig. 6. Violence Not started

Fig. 7. Violence about to start

Fig. 8. Violence slightly started

Fig. 9. Violence furiously started

Above are four figures which show four different phases of surveillance video. Initially in Fig. 6, Violence has not yet started. In Fig. 7 Violence is about to start, people are slightly pushing each other. Fig. 8 shows the start of violence and in Fig. 9 Violence has started furiously.

```
(bob_py2) sekhar@ubuntu:~/Desktop/ASAGS/continous_system$ python test_violence_v
ideo.py
Using TensorFlow backend.
2018-03-19 01:19:11.227351: I tensorflow/core/platform/cpu_feature_guard.cc:140]
 Your CPU supports instructions that this TensorFlow binary was not compiled to
use: AVX2 FMA
[[  2.56899511e-07]]
[[ 0.14527404]]
[[ 0.2733964]]
[[ 0.99957734]]
[[ 0.99810362]]
[[ 0.66406441]]
[[ 0.06192874]]
```

Fig. 10. System's output for the above video

In Fig. 10, it shows the output of the system for the particular video shown in previous figures. As we can see for the initial frame, output value is very less, as the scene gets tense in Fig. 7, the system output value increases. When the violence starts in Fig. 9, output value increases to 0.999 indicating violence. Later on in the video violence decreases gradually and hence the output value falls down to 0.06.

VI. Conclusion and Further Work

Timely detection of violence in real time is of much importance. System is able to accurately detect violent scenarios in real time. It is scalable enough to process three to four parallel video streams at a time with a household PC setup. Whatever work has been done is to give attention and importance to accurate real time surveillance.

Further the accuracy can be greatly improved by using results of Feature Selection algorithm of AdaBoost. As explained above, once we arrange the features in increasing order of their error rate, we get order of importance of 336 features. In that particular order, we can associate the weights of input layer in neural net. The features which are highly important can have a higher weight at their input node and as the importance decreases, weights can also be decreased. This may make a difference in increasing the accuracy of neural net.

Acknowledgment

We would like to thank Dr. T Hassner [1] for his contributions towards real time surveillance, this project would not have been possible without the generation of Violent Flow Descriptors (VIFs).

References

[1] T. Hassner, Y. Itcher, O. Kliper Gross. *Violent Flows: Real-Time Detection of Violent Crowd Behavior*,3rd IEEE International Workshop on Socially Intelligent Surveillance and Monitoring (SISM) at the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR),June 2012

[2] Ce. Liu. *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*,Massachusetts Institute of Technology,Ph.D. Thesis,2009

[3] Anjos, André AND El Shafey, Laurent AND Wallace, Roy AND Günther, Manuel AND McCool, Christopher AND Marcel, Sébastien. *Bob: a free signal processing and machine learning toolbox for researchers*,20th ACM Conference on Multimedia Systems (ACMMM), Nara Japan, 2012

[4] Ruihu Wang. *AdaBoost for Feature Selection, Classification and Its Relation with SVM, A Review*, Department of Science and Technology Chongqing University of Arts and Sciences Yongchuan, Chongqing 402160, CHINA

[5] Zhen Zhang, Weimin Shao and Hong Zhang. *A learning algorithm for multilayer perceptron as classifier*. IJCNN'99. International Joint Conference on Neural Networks. Proceedings.

[6] Ibrahim Kajo, Aamir Saeed Malik and Nidal Kamel. *An evaluation of optical flow algorithms for crowd analytics in surveillance system*. 2016 6th International Conference on Intelligent and Advanced Systems (ICIAS)

[7] Ivan Culjak, David Abram, Tomislav Pribanic, Hrvoje Dzapo and Mario Cifrek. *A brief introduction to OpenCV*. 2012 Proceedings of the 35th International Convention MIPRO.

[8] Marius Drulea and Sergiu Nedevschi. *Motion Estimation Using the Correlation Transform*. IEEE Transactions on Image Processing, 2013.

[9] Jobin T Philip, Binoshi Samuvel, K. Pradeesh and N. K. Nimmi. *A comparative study of block matching and optical flow motion estimation algorithms*. 2014 Annual International Conference on Emerging Research Areas: Magnetics, Machines and Drives.

[10] Petra Vidnerova and Roman Neruda. *Evolving keras architectures for sensor data analysis*. 2017 Federated Conference on Computer Science and Information Systems.