University of Stirling

Faculty of Natural Sciences

Division of Computing Science and Mathematics

Classification of crowd motion using Computer Vision and ML

Author:
Chaitanya Kumar Reddy Alva

Supervisor:
Dr. Deepayan Bhowmik

Dissertation submitted in partial fulfillment for the degree of Master of Science in *BigData*

September 2020

Abstract

Summary of the dissertation ***within one page***.

This template starts the page numbering at the foot of this page. While you are printing drafts, you might find it useful to add the printing date and time into the footer – to help you, and your supervisor, tell which version is most current.

It is suggested that the abstract be structured as follows:

- Problem: What you tackled, and why this needed a solution
- Objectives: What you set out to achieve, and how this addressed the problem
- Methodology: How you went about solving the problem
- Achievements: What you managed to achieve, and how far it meets your objectives.

Attestation

I understand the nature of plagiarism, and I am aware of the University's policy on this.

I certify that this dissertation reports original work by me during my University project except for the following (*adjust according to the circumstances*):

- The technology review in Section 2.5 was largely taken from [17].
- The code discussed in Section 3.1 was created by Acme Corporation (`www.acme-corp.com/JavaExpert`) and was used in accordance with the licence supplied.
- The code discussed in Section 3.5 was written by my supervisor.
- The code discussed in Section 4.2 was developed by me during a vacation placement with the collaborating company. In addition, this used ideas I had already developed in my own time.

Signature: *(you must delete this, then sign and date this page)* **Date**

Acknowledgements

Acknowledge anyone that you wish to thank who has helped you in your work or supported you in any way: such as your supervisor, technical support staff, fellow students, external organisations or family. Acknowledge the source of any work that is not your own.

Contents

List of Figures

List of Tables

Listings

1 Introduction

Evolution of technology, artificial intelligence and robotics helped the world to achieve new targets in the field of security and surveillance. The combination of machine learning and surveillance emerged as a powerful tool to tackle crime, illegal activities and violent protests. In the recent years, we experienced many such activities that made us to understand the importance and necessity of automated video surveillance. With the help of computer vision, detecting people in the frame, counting the people in a dense scene, abnormal behaviour detection and motion analysis in surveillance videos is done without any manual intervention. Crowd motion analysis and abnormal behaviour detection have always been a challenging task in this field. Reason being the number of independent factors that define the motion of the individual. Analysing the motion of the crowd can avoid many voluntary or involuntary violence, riots, traffic jams and stampede.

1.1 General Context and motivation

As mentioned in [1], the main objectives of automated surveillance video analysis are continuous monitoring, reduction in laborious human task, object identification or action recognition and crowd analysis. This paper talks about detecting different types of crowd motion and abnormal behaviour tracking using CNN. Most of the study on analysing the crowd is done on the following areas.

- Counting the crowd.
- Types of the crowd based on density.
- Detecting the motion in the frame.
- Identifying the types of motion.

1.1.1 Counting the crowd.

Counting the crowd is a very important in order to maintain the safety and security. It helps to plan the events, traffic and the capacity of any situation. But, counting dense crowd is a difficult task. As mentioned in [2], more than 17% of the total papers written on crowd analysis are published on crowd counting. For example, [3] generalise different types of crowd counting and different algorithms used in the past while proposing a new approach of using the statistics of thespatio-temporal wavelet sub-bands. [4] uses a multi source (identifying different parts of the body in the frames from different algorithms) and Markov Random Field to count the people in the dense crowd.

1.1.2 Types of the crowd based on density

It is important to categorise the type of the crowd to understand the dynamics of the motion. Moore [5] suggests, the crowd can be treated as particles in fluid dynamics and the crowd is of 3 types, microscopic, mesoscopic and macroscopic based on the density. Microscopic view of crowd through a hydrodynamic lens implies understanding the flow of every individual in crowd and this is specific to limited number of individuals in the frame. Mesoscopic view implies more number of people in a frame. Macroscopic view implies the frame filled with people. The personal and interaction forces in each case are different which in turn drive the motion of the crowd. To further explain, the interaction force is very less in a microscopic view but very high in macroscopic view.

1.1.3 Detecting the motion in the frame

Detecting the motion in the frame can be done either by training a model which involves feeding the motion images into a CNN architecture or without training by just tracking every point in the frame using optical flow. Santoro [6] did optical flow computation with the help of Shi-Tomasi Corner Detection and Lucas–Kanade algorithm to detect the motion of the crowd. Where as [7] uses motion information Images (MII) to train a CNN model for the motion and abnormality detection

1.1.4 Identifying the types of motion

Identifying the types of the motion can be a very useful in order to understand the crowd behaviour, planning an event, avoiding traffic jams and predicting the abnormal motion. Wei [8] trained 2 VGG16 CNN architecture models to detect the type of the crowd whether it is homogenous, heterogeneous or violent crowd. [9] s tudies the stability with the help of Tylor's theorem and Jacobean matrix and identifies the crowd motion to be of 5 generic types i.e. Lanes, arc/circle, fountainheads, bottlenecks and blocks.

1.2 Aim and Objectives

This project aims to classify different types of crowd motion into 4 classes: Arcs, Lanes, Converging/Diverging and Blocks/Random. The project focuses on the drone footages by exploring state of the art CNN and machine learning techniques. This project also explore different optical flow techniques and compare the advantages and disadvantages of the existing techniques. The proposed model also helps to understand the motion in the scene and can be used to train multiple anomaly detection techniques. To achieve these aims the following objectives were setup:

1. Identifying the interesting features in the frame to track the motion with the help of different corner detection techniques.
2. Exploring multiple options for noise reduction and improving the quality of the tracking through density based clustering algorithms.
3. Developing a new approach to track the points which improves the quality of the features and reduce the computation time.
4. Use of this approach to understand the motion in the scene and create the data for anomaly detection.
5. Creating the Motion information image dataset from multiple videos and labelling the data for training CNN model.
6. Creating a Spacio Temporal dataset with block wise information of magnitude and direction for training the machine learning models.
7. Comparing the best approach visually and statistically to identify the best approach in this context.
8. Improving the model to perform similar operation from a fixed lens security cameras to a dynamic drone footages.

1.3 Achievements

Summarise what this project has achieved. Avoid terms like I achieved this or that.

1.4 Overview of Dissertation

Briefly overview the contents of what follows in the dissertation.

2 Background

Computer vision evolved from many complex theories, algorithms and models. This paper mainly talks about the video surveillance. This section helps to understand the required technical details confined to this area.

2.1 Optical Flow

Optical flow can possibly be one of the most important concepts of computer vision. Optical flow is used to find the pattern in the movement of the objects from one frame to another. This is widely used in the fields like robotics, image processing, motion detection, object segmentation etc. Videos are the series of images. These images can be independent from one another. But, in the real time, a video captures consecutive change in the pixels in certain duration of time. There are many algorithms which discuss the relation between these pixels in two different frames. [10] discuss various types of optical flow algorithms and evaluates them. This paper concludes that Lucas Kanade Algorithm is best among the other 8 optical flow algorithms.

Optical flow diagrams are usually denoted by the vectors pointing the change from frame F1 to frame F2. But in real time, it is easy to concentrate on only those points which provide more insights. For example movement of the hand from F1 to F2 changes hundreds of pixels and can be redundant. Rather it is simple and more appropriate to see the flow of only those pixels at the corner of the hand. Thus, Corner detection algorithms are used to reduce the complexity and improve the performance of the algorithms.

2.1.1 Corner Detection

This paper trails 2 types of corner detection techniques to check the best possibility for the model.

- Shi-Tomasi Corner detection.
- FAST Corner detection.

Shi Tomasi Corner detection algorithm is similar to Harris Corner Detector. it is widely used in detecting the interest points and feature descriptors. Interest points can be corners edges and blobs and are invariant to rotation, translation, intensity and scale changes. Only difference in harris corner detection and Shi Tomasi corner detection is the computed R value (used to detect the corner). FAST (Features from Accelerated Segment Test) on the other hand uses a different technique to predict not only the corners but also the edges based on the colour intensity and the threshold.

2.1.2 Lucas Kanade Algorithm

In the conclusion of [10], we can see that Lucas Kanade Algorithm is the one of the best algorithm to detect the optical flow. The assumption of Lucas Kanade algorithm is the flow of the local neighbourhood of the pixel is constant. It combines all the information from the surrounding pixels and often solves the inherent ambiguity of the optical flow equation. It is also considered to be less sensitive to the noise.

2.2 Density based clustering

Clustering in general is combining a group of similar objects based on their similarities like shape, angle, magnitude and position. In order to reduce the memory consumption of the CPU/GPU it is important to consider those points which are critical to the analysis. Thus, clustering the points and vectors based on the position and direction helps to combine the similar points and vectors to predict the movement of the crowd. In this paper we have considered using 2 types of density based clustering.

- DBSCAN.
- OPTICS.

2.3 Convolutional Neural networks

Convolutional neural networks are the advanced concept of neural networks which gives computers the ability to understand the images and videos. CNNs currently are being used in a wide range of application like Robotics, Face detection, Crowd detection, Weather study, Advertising, Environmental studies etc. Every neurone in the CNN has the learnable weights. They are initialised with random weights and can be trained to develop a model. CNN are comprised of below 3 topics.

- Convolution Networks (ConvNets).
- Pooling.
- Fully Connected Layers.

2.3.1 Convolution Networks (ConvNets)

Convolution Networks also called as ConvNets is the process of changing the pixels of the image using filters. The image is a matrix of pixels and a filter/kernel is used to alter the pixel value with matrix multiplication. This filter is applied on the whole image by striding through the image. There are different filters for different types of results.

2.3.2 Pooling

Pooling is the process of reducing the size of the image with the help of a filter. The pooling is usually of 2 types, Average pooling and Max pooling. Image is reduced to a desired size by the filter by taking the average of the pixels or Max pixel depending on the pooling technique.

2.3.3 Fully Connected Layers

Fully connected layers are the neural networks which has the 1D array of the ConvNets as the inputs and a series of different hidden layers which are fully connected. The output of these networks are the classification nodes which can either be integers or One Hot encoded values predicting the classification. The prediction of the classification is usually done by SoftMax (Picks the highest probability node).

2.3.4 Notable CNN Architectures

There are few CNN architectures which are available as the modules. These modules are pre-trained and can be directly implements provided the inputs and outputs are exactly same as expected by the module. PyTorch library can be explored to find the list and implementation of these networks. This project implements below 3 CNN architectures.

- AlexNet
- VGG
- ResNet

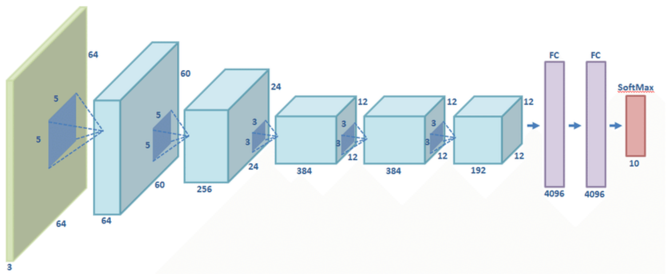

Figure 2.1: AlexNet Architecture

2.3.5 AlexNet

AlexNet CNN architecture was first proposed in [11] by Alex Krizhevsky in the year 2012. The architecture diagram 2.1 of the Alexnet gives the insight of the depth and the processing of the CNN. AlexNet contained 8 layers, among them 5 were convolution layers followed by Max pooling and the last 3 layers were the fully connected layers. Alex net was considered to be one of the most famous architectures at that time. It uses non saturating ReLu function which can be changed to tanh and sigmoid functions to improve the performance.

2.3.6 VGG

VGG architecture was first proposed in [12] by Karen Simonyan in the year 2015. As per this paper there are 4 variations (vgg11, vgg13, vgg16 & vgg19) in the architecture depending on the depth of the networks. Vgg architecture takes an input of 224*224*3 image and pass it through different convolution layers followed by Max pooling. This paper implements vgg11 architecture and this architecture is different from the AlexNet as described in 2.2.

AlexNet architecture has max pool layer after every convolutional layer, but in VGG architecture there is a difference in the placement of the Max pool layer some times after a series of 2 or 3 convolutional layers. This helps to retain the feature information before reducing the size through max pooling. towards the end of the architecture there are 3 layers of fully connected networks to classify different outputs.

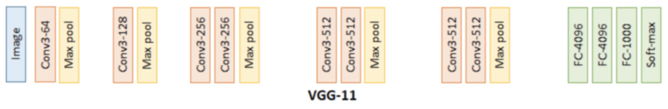

Figure 2.2: VGG 11 Architecture

2.3.7 ResNet

ResNet is residual neural network which is build on constructs known from pyramidal cells in the cerebral cortex. In order to achieve this property it uses skip connections over some layers. The model is implemented with double or triple skips using ReLu and batch normalisation. ResNets are built on the idea that the added more layers to the CNN architectures will not reduce the error rather just increase the computation cost. This is the reason, this model proposes to use the skip connects to reach to the lower layers faster when required by preserving the feature characteristics. The skip connections are described in the figure 2.3

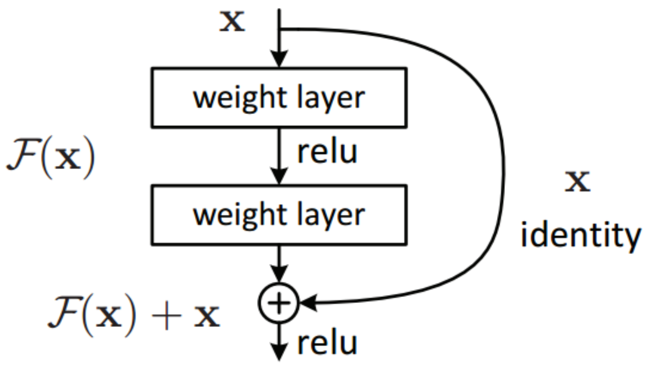

Figure 2.3: ResNet Skip connections

2.4 Machine Learning Models

There are 3 types of machine learning algorithms: Supervised learning, Unsupervised learning and reinforcement learning. Supervised learning works on the concept of learning from the previous experiences. In this case, the models are trained with the help of labelled data. The model trains from the inputs and is implemented on the fresh data to check the model accuracy. Unsupervised learning is the concept of leaning based on the input information with out any previous experience. In this case of learning the models are fed with large amounts of data and the model plots the data to find the similarities in the placement, pattern recognition and prediction. The third type of machine learning approach is the reinforcement learning where the machine is exposed to an environment where it trains itself continually using trial and error.

This project deals with the classification models and implements the classification ML models under supervised learning. The data is trained using the following models:

- Logistic Regression
- Support Vector Machines
- K nearest neighbours
- Gaussian Naive Bayes
- Perceptron
- Stochastic Gradient Descent
- Decision Tree
- Random Forest

2.4.1 Logistic Regression

Logistic Regression is used to estimate the discrete values based on the inputs provided. Logistic regression can be implemented in multiple ways depending on the requirement. in Simple logistic regression, the classification is binary. where as in the multi-nominal logistic regression the output can be multiple classes. The idea of the logistic regression is developed from the linear regression where the model defines a linear function to predict the value. But in case of classification all the values are either 0 or 1. In this case the best fit cannot be a linear function, rather it uses the sigmoid function to predict the values as shown in 2.4. The derivation of the Logistic regression likelihood function is as follows.

$$y = mx + c \tag{2.1}$$

$$\sigma(y) = \frac{1}{1 + e^{-(mx+c)}} \tag{2.2}$$

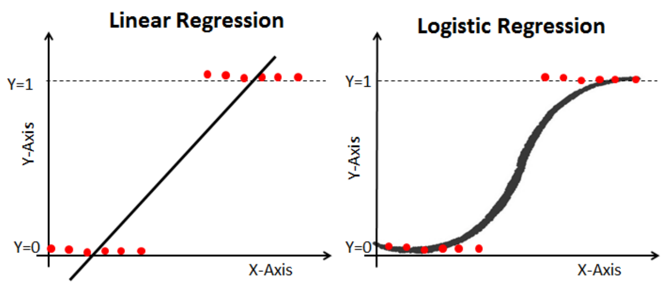

Figure 2.4: Linear Regression Vs Logistic Regression

2.4.2 Support Vector Machines

Support vector machines is used to classify the data into groups and generally works on the supervised dataset. It plots every point of the data in the n dimensional plot. Where n being number of features. As all the data is plotted in n dimensions, the algorithm works to split the data into the different classes based on the plot. The SVMs are considered to be one of the best classification algorithms.

2.4.3 K nearest neighbours

K nearest neighbours algorithm is one of the simple algorithm which takes the parameter k. The value k defines the clustering the data based on the distance function. There are many types of the distance functions namely: Manhattan, Euclidean, Minkowski and Hamming distance. Among these distance functions, hamming is used for the classification problems. KNN algorithm is considered to be computationally expensive, specially to work on a larger dataset. Basic idea is to calculate the distance matrix of all the points and classify them based on the number of classes with a threshold value.

2.4.4 Gaussian Naive Bayes

Gaussian Naive Bayes technique is based on the Bayes theorem which is built on the assumption that the predictors are completely independent. Naive Bayes theorem is simple to calculate and is known for its simplicity yet very efficient model. The theorem is based on calculating the posterior probability P(c|x) from P(c), P(x) and P(x|c) as mentioned in the equation 2.3

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{2.3}$$

2.4.5 Perceptron

Perceptron algorithm is considered to be the best algorithm to be used on the linearly separable data. Perceptrons are the 3 layer neural networks where the inputs are the features and the middle layer takes the weighted sum of the inputs and applies a function and gives an output. This output is compared with the actual value and the loss is back propagated to change the weights. After multiple calculations, the algorithm returns the values with less error and thus nearly fit to the classification problem.

2.4.6 Stochastic Gradient Descent

Stochastic Gradient Descent works on the concept of Gradient Descent algorithm. This can be called as the modified Gradient Descent which tackles the problem of computation logic. In gradient descent, every instance is used to calculate the best fit of the model. This can be cost consuming in case of many samples. In order to reduce this computation problem, SGD uses the batch selection process where it uses one record from each batch for the computation of the best fit. SGD is a considered to be a good model on large number of data with redundant values.

2.4.7 Decision Tree

Decision tree model is one of the most frequently used classification model. It can be used both on the categorical and continuous target variables. This model works on the bases of splitting the data based on a category. The main components of decision tree are as follows

- Root Node: It is the base node of the tree and generally represents the total population.
- Splitting: It is a process of dividing a node into two or more sub-nodes.
- Decision Node: The node which is split into sub nodes
- Leaf Node: The node which do not split into further nodes.
- Pruning: Reducing the sub nodes of the decision tree.
- Branch: It is the sub tree of the decision tree.

There are different types of splitting. But in general the splitting is done on the bases of 2 algorithms Gini and Information Gain. Gini is the probabilistic way of splitting the node into binary nodes by calculating the square of probability of success and failure of the sub nodes. It then uses the weighted Gini score on the each node to split the nodes. On the other hand the Information gain works on calculating the entropy on each class and further calculate the information gain on each class. Depending on the information gain, it splits the node into sub nodes.

2.4.8 Random Forest

Random forest is the collection of multiple decision trees algorithm. It classifies the class with most outputs from the different decision trees. The basic algorithm works by row sampling and feature sampling the dataset to multiple decision trees. This is bootstrap of the data to multiple decision trees and get the output from all the trees. Now the aggregation works on the outputs of different decision trees and picks the class with more predictions from the decision trees.

3 State-of-the-Art

Crowd motion analysis deals with a combination of computer vision, Image processing and machine learning techniques. This section provides in depth knowledge of related work and state of the art techniques by explaining the recent and ground braking scientific papers in this field. More specifically the motivation introduction and the techniques used in these papers.

3.1 Crowd analysis using optical flow

The papers [6] [13] gives a generic and very efficient way of tracking the crowd motion in the videos. As shown in the figure 3.1 below, the process of crowd tracking is done in 4 steps. KLT feature tracker is used to do the optical flow estimation.KLT feature tracker is the combination of Shi- Tomasi corner detection and the famous Lucas Kanade optical flow algorithm. Shi-Tomasi corner detection technique is used to identify the interesting corners of the frame. A section of surrounding pixels of these points are also added to the tracking. This step is followed by tracking the points in the consequent frames. The tracking is done with the help of Lucas Kanade algorithm. In addition to the change in the points, magnitude and the angle of the vector is also calculated at this point of time.

Now that all the vectors are derived from frame fk and fk+1, block partitioning is done on the whole frame. The frame is divided in to multiple blocks and the vectors in the specific blocks are clustered based on the angle and magnitude. This clustering is done with the help of DBSCAN algorithm. All the points which are considered to be one group are marked with single vector. Any person leaving the crowd or joining the crowd is considered as separate or single block accordingly. The result of this paper are shown in the figure 3.2. The person behind the crowd is considered as separate group and the tracking is done multiple times to get the flow.

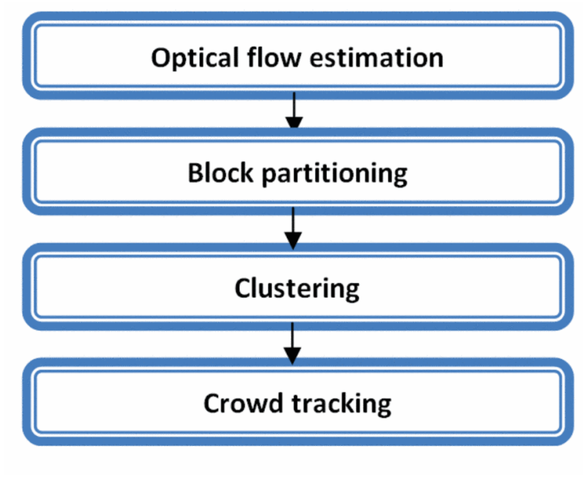

Figure 3.1: Data Flow Diagram

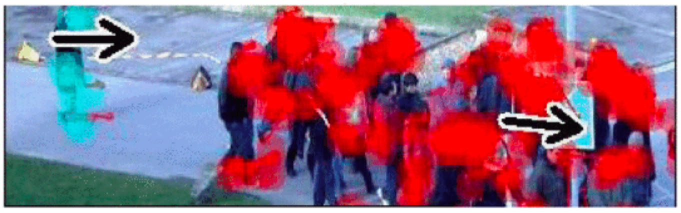

Figure 3.2: Density based partitioning and crowd tracking

The paper [14] is an interesting use of optical flow to detect the dominant motion in the crowded scenes. This paper suggests a combination of both Shi Tomasi corner detection algorithm and FAST corner detection to identify the interesting points in the frame. Keeping track on interesting points in multiple frames, the trajectory is captured. New feature points are added in every 5 frames to handle the load. The new feature points which are close to the old points are discarded. By getting the trajectories of all the points, a new clustering framework Longest Common Subsequences (LCSS) is introduced. With the help of this framework, multiple trajectories are compared for the matching points and the dominant path is captured by clustering trajectories.

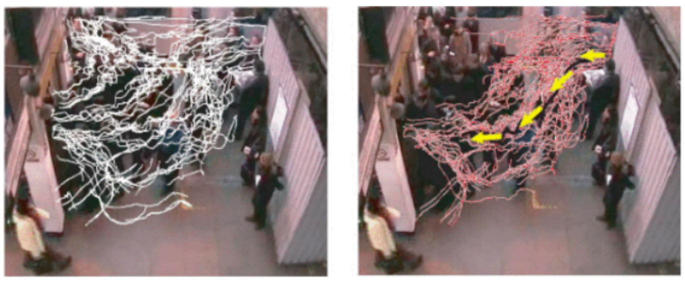

Figure 3.3: Dominant motion detection results

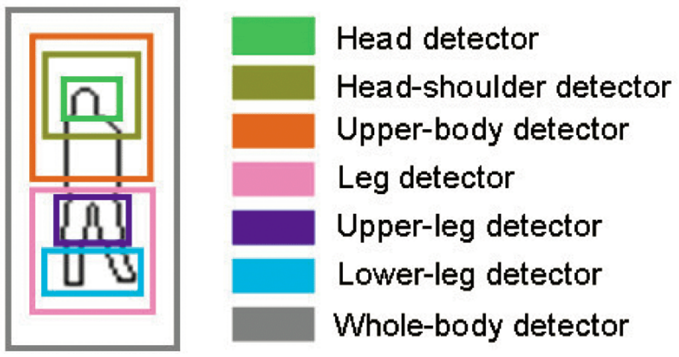

Figure 3.4: Multi Feature multi detector model

The results of this model are shown in the figure 3.3.

3.2 Crowd counting

Crowd counting is a difficult task and the accuracy of the proposed models depends on the scene. In case of large density of crowd, it is extremely hard to track the crowd for counting. Most recent paper [15] suggests that crowd counting can be done in 2 ways. The first is by using detection based models, where the crowd is being tracked with the help of body parts or the shape and the count is produced from the tracking model. The second is Regression based models, where the model predicts the number of the crowd with out tracking them. This model is based on developing a density map and estimating the count from the produced density map. A part from these 2 methods there is another method based on CNN, which also produced promising results. But, in the CNN models, there are cases where the system predicted various different objects as human heads causing a huge difference in the count. [15] deals with the combination of density based model and the CNN in order to fix the on going issue with the CNN models.

[16] is another interesting paper which can be grouped into the detection based models. In this paper, the writer creates a part-template tree with the human postures in different angles, poses and shapes. A hierarchical part-template matching algorithm is used to estimate human shapes and poses by matching local image. Multiple detectors are used to detect the multiple body shapes as shown in the figure 3.4. The segmentation is done based on these multiple detectors. Background subtraction is used to evaluate the model and produced promising results. [17] can be grouped under the regression based model of counting crowds. This paper suggests the motion segmentation of the crowd clustered based on the direction. The count on the each direction is estimated using the Gaussian process. This model is explained in the figure 3.5

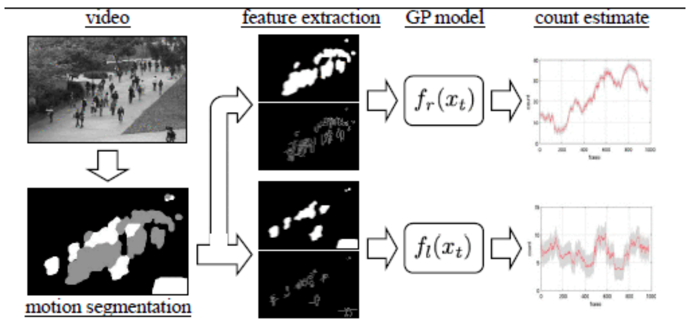

Figure 3.5: Regression Based crowd counting

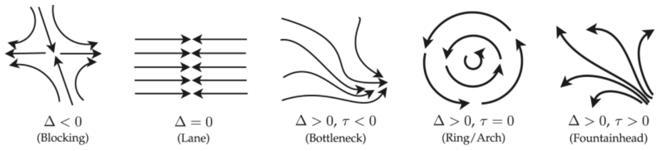

Figure 3.6: Types of motion based on the product and sum of eiganvalues

3.3 Motion detection and classification

In order to understand the behaviour of the crowd, it is important to detect the motion and also classify them. For example to check if the vehicles are moving in the right path, people following the suggested path, person walking in restricted area etc. The crowd motion can be categorised into different groups based on the scene. If the scene is to identify the anomaly in the crowd motion for example in the traffic, the motion can be grouped into Lanes, Arcs, blocks. In case of entering or exiting enclosed buildings the motion can be Bottlenecks or Fountainheads. In case of violent protest, it can be categorised into converging or diverging. In all the cases, it is important to study the type of the motion in order to tackle the situation.

[9] states that crowd motion can be categorised into 5 types: Lanes, Arcs, Bottlenecks, Fountainheads and Blocks. This paper suggests a model which can categorise the videos with out a training model. It used Particle advection to pick the interesting points in the frame and track these points as a spacio-temporal data. The motion is further produced in the equation using Taylor theorem. The stability of the motion is detected by calculating Jacobian Matrix. The eigenvalues of this matrix is used to classify the motion as shown in the figure 3.6.

Identifying the type of the crowd is key to understand the type of the action, scene of the incident and actions of the people. This paper [8] categorise to crowd into 3 types based on a model called BMO model (Behaviour, Mood and Organisation model). This model is a rule based model which helps

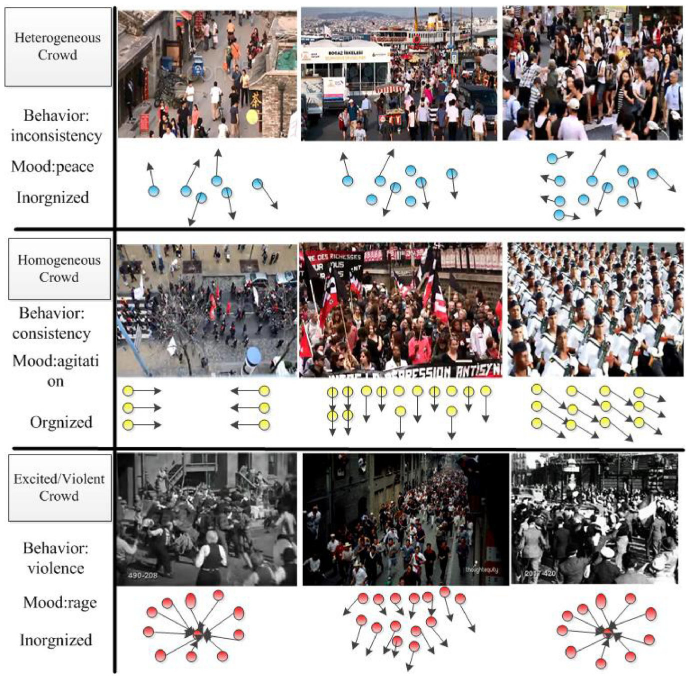

Figure 3.7: Crowd categorisation using 2 deep networks

to detect the crowd motion to be grouped into Heterogenous, Homogenous and Violent crowds. To implement this model, they trained 2 very deep VGG networks and combined the FCN layers of both the networks to predict the type of the crowd. The inputs to these networks are the motion map and key frame. The results and the implementation of this model are shown in the figure 3.7

3.4 Anomaly detection in the crowd motion

Another very interesting and key papers published are focused on the Anomaly detection. This type of papers focus on identifying unexpected behaviour in the frame. For example the paper [18] focuses on 3 types of anomalies, namely: Point Anomaly which points a single object in the frame. This can be an unexpected motion or sudden change in the magnitude of single object. The second type id the collective anomaly, where most of the objects in the frame experience sudden drift in the direction and velocity. This type of anomaly is usually found in the riots, explosions etc. The third type of anomaly is contextual anomaly which can be unexpected shaped object in the frame etc. The figure 3.8 explains the different types of the anomalies clearly.

This paper implements the model with the footages from the stable surveillance cameras and techniques like background reduction. It also proposes a new way of gathering the features like direction, change in the points, dis-

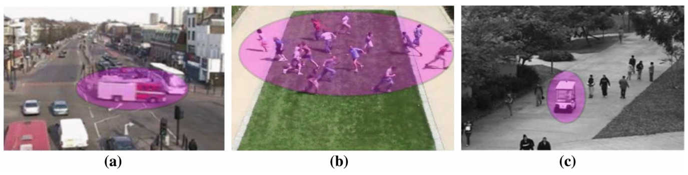

Figure 3.8: Types of anomalies

tance etc. These features are further classified using k-means clustering and distance calculation to predict the motion to be expected or unexpected.

4 Data Collection & Pre Processing

Video datasets can be grouped into 3 types: **Object centric**, **Location centric** and **Motion centric**. In the object centric video sets, the videos are the combination of multiple objects and the implementation is based on tracking a specific object, identifying multiple objects in the frame or auto captioning the objects in the frame. Most of this work is done by training multiple CNN and RNN architectures. Identification of a specific object requires segmentation architectures like DeepLab or FCNNet. The second group of videos are location centric where the number of videos in the dataset are less but the length of the videos are long enough to understand the location details and the movement in to the scene. Implementation on this type of videos can be understanding the scene by tracking the movements continuously. The third group of video sets are a combination of different types of motions by multiple objects. In this group the implementation is based on the motion analysis specially tracking the type of motion, anomaly detection and threat prediction.

4.1 Datasets

This paper focuses on 2 datasets: Virat Dataset, which is location centric to understand and track the movement in the scene which intern provides the motion concentrated areas in the scene and UCF Crowd Dataset which is motion centric to track multiple objects and classify different types of motion. Virat dataset is clubbed with 2 scenes which are split into 61 videos and UCF dataset contains 38 videos of different motions. In both the datasets it is considered the video is captured by the fixed lens cameras or

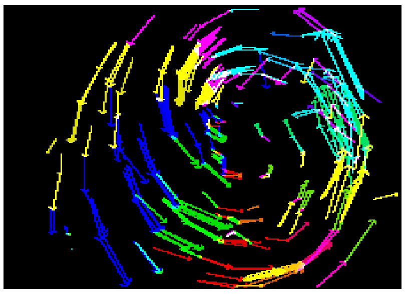

Figure 4.1: Motion Information Image

the drones capturing footages from a fixed position.

4.2 Data Preparation

This section gives a generic view on different types of datasets used to train the models for this project and how these data sets are obtained. The project works on 2 types of datasets, Motion Information Images (MIIs)and Block wise dominant motion information.

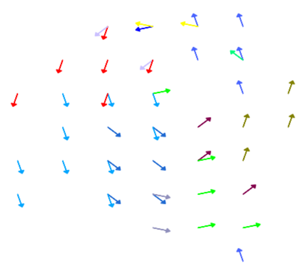
Figure 4.2: Block wise dominant motion Information

MIIs are the images with the motion information of the objects in every 5 frames. These images are created from the optical flow and manually assigned one label per frame. A threshold on the magnitude is created depending on the frame and this threshold is used to reduce the noise. These images are used to train the CNN model which classify the type of motion in the frame. Example of MII are shown in 4.1. Different colours in the image represents 12 directions the motion is pointing.

On the other hand for creating the datasets for Machine learning models, every frame is blocked into 8*8 blocks. A block wise dominant motion information is created after the implementation logic which provides the dominant motion in every block. This information is labelled manually on every frame and stored in a comma separated value file. The visual representation of this information is shown in 4.2. Different colours in the image represents 12 directions the motion is pointing.

4.3 Data Pre Processing

This project deals with 2 types of data, Images and processed Spacio Temporal data. Due to the unavailability of the annotated file on this context, the labels are manually created in both the cases.

5 Optical Flow Implementation

5.1 Feature Detection

Feature detection using open CV is the a very useful and important technique is most of the areas which deals with images and videos. Each image or the frame is the combination of pixels and each each pixel is the number that represents a colour. For a computer it is extremely difficult to understand the difference between these numbers and thus, feature detection is a complex yet interesting topic to understand. This paper tries to implements 3 different corner detection techniques 1) Harris Corner detection, 2) Shi Tomasi corner detection. 3) FAST algorithm for corner detection. The advantages and disadvantages of these techniques are discussed further.

5.1.1 Harris Corner Detection

This corner detection technique was first introduced by Chris Harris & Mike Stephens in their paper [19] in 1988. Idea behind this technique is to find the difference between the intensity for a displacement of (u, v) in all directions. The mathematical equation 5.1 for the same is given below.

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2 \tag{5.1}$$

Window function *w(x, y)* is either a rectangular window or gaussian window which gives weights to pixels underneath. The corners are detected by maximising the *E(u, v)* which means maximising the second term by using Tylor's theorem as shown in the equations 5.2 & 5.3

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \tag{5.2}$$

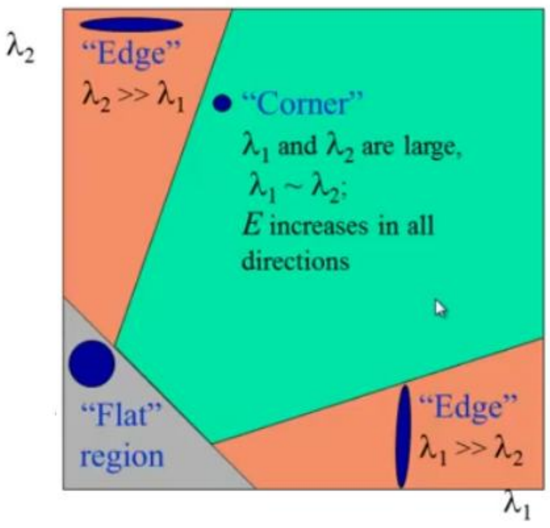

Figure 5.1: Harris corner detection using eigenvalues

where

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \tag{5.3}$$

Here, I_x and I_y are image derivatives in x and y directions respectively. *R* value is calculated from the eigenvalues of the matrix *M* from the equation 5.4

$$R = det(M) - K(trace(M))^2 \tag{5.4}$$

where

- det(M) = $\lambda_1\lambda_2$.
- trace(M) = $\lambda_1 + \lambda_2$.
- λ_1 and λ_2 are the eigenvalues of M.
- k - Harris detector free parameter in the equation.

edge, corner and flat region in the image are detected with the help of eigenvalues as shown in the figure 5.1

Listing 5.1: Python Code to compute Harris Corner detection

```
dst = cv2.cornerHarris(self.gray,2,3,0.04)
dst = cv2.dilate(dst, None)
tempframe2[dst > 0.01 * dst.max()] = [0, 0, 255]
print(points.shape,dst.shape)
cv2.imshow('Harris', tempframe2)
if cv2.waitKey(1) & 0xff == 27:
    cv2.destroyAllWindows()
```

5.1.2 Shi Tomasi Corner Detection

Shi Tomasi Corner detection was first proposed by J. Shi and C. Tomasi in the paper [20] in 1994. This approach is a small modification to the Harris Corner detection in calculating the *R* value. As mentioned before the *R* value is calculated by 5.4 But, as per Shi Tomasi Corner detection, the *R* value is calculated by minimising the product of eigenvalues as shown in 5.5.

$$R = min(\lambda_1, \lambda_2) \tag{5.5}$$

If this value is greater than the threshold, then it is considered as the corner.

Listing 5.2: Python Code to compute Shi Tomasi Corner detection

```
        self.gray = cv2.cvtColor(self.frame, \
                        cv2.COLOR_BGR2GRAY)
        points = cv2.goodFeaturesToTrack(self.gray, \
                        mask=None,\
                        **self.good_feature_params)
        corners = np.int0(points)
        for i in corners:
            x, y = i.ravel()
            cv2.circle(tempframe1, (x, y), 3, 255, -1)
        cv2.imshow('Shi tomasi', tempframe1)
        if cv2.waitKey(1) & 0xff == 27:
            cv2.destroyAllWindows()
```

5.1.3 FAST Algorithm for Corner Detection

FAST (Features from Accelerated Segment Test) algorithm was proposed by Edward Rosten and Tom Drummond in their paper [21] in 2006. Unlike Harris and Shi Tomasi corner detection techniques, FAST corner detection is considered as the simple and fast detection technique. The algorithm of this technique is as follows.

1. Select a pixel p to check for the corner detection and calculate the intensity I_p.
2. Create a threshold value to detect the corner.
3. Select 16 surrounding pixels and calculate the intensity of all those pixels.
4. If the intensity I_p is greater than or less than threshold value of the 16 intensities then It is considered as corner.
5. Later this algorithm is modified to decrease the computation time by comparing the intensity with only 4 key corners.

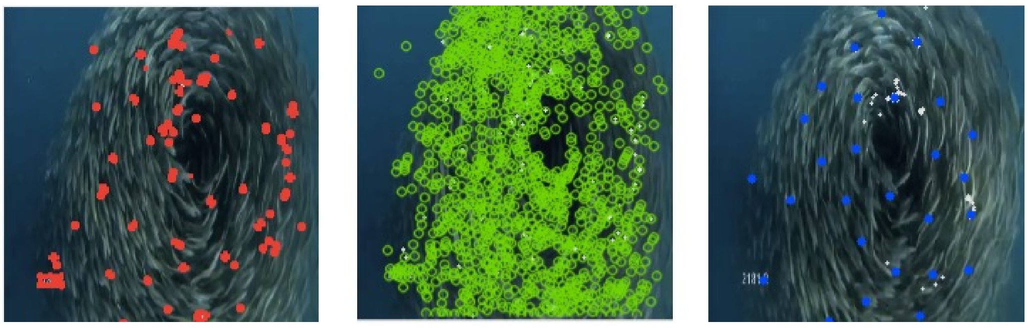
Figure 5.2: Left : Harris Corner Detection; Centre : FAST Algorithm; Right : Shi Tomasi Corner Detection

As the name suggest FAST algorithm is considerable fast in computing the corners.

Listing 5.3: Python Code to compute FAST Corner detection

```
fast = cv2.FastFeatureDetector_create()
fastpoints = fast.detect(self.gray, None)
fastarr = np.array([[[fpoint.pt[0], fpoint.pt[1]]]
                    for fpoint in fastpoints]
                   , dtype=np.float32)
points = np.concatenate((points, fastarr), 0)
img2 = cv2.drawKeypoints(tempframe3, fastpoints
                         ,np.array([]),
                         color=(0, 225, 0))
cv2.imshow('Fast', img2)
if cv2.waitKey(1) & 0xff == 27:
    cv2.destroyAllWindows()
```

5.1.4 Analysis on Feature Detection

All the 3 corner detection techniques implemented sequentially and verified on the basis of time consumption, Quantity and over all performance. The table 5.1provides the insights of the experiment and the detected corners are shown in the figure 5.2. Though FAST algorithm is faster in detecting the corners, it detects considerably more corners than the other 2 algorithms. Due to this high count, the performance of the model has decreased. The same applies to Harris corner detection as there are multiple corners detecting the same object. Due to this redundancy, the performance of the model degraded. Thus, Shi Tomasi Corner detection is used as the feature detection technique for this project.

Corner Detection	Harris	FAST	Shi Tomasi
Time	0.083s	0.01s	0.04s
Quantity	130	986	23
Performance	Medium	Low	HIgh

Table 5.1: Table to compare different corner detection techniques

5.2 Density Based Clustering

The main focus of the project is to track the motion of the objects. Two main issues to select the key features are noise reduction and redundancy detection. In order to achieve this target, it is necessary to remove those features without motion and reduce the dimensionality by eliminating the redundantly tracked features. Density based clustering is used to filter out the features without motion and reduce the noise in the frame.

5.2.1 DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) clustering method is widely used clustering algorithm. There are 3 components worth knowing to understand the DBSCAN algorithm.

- **Core Point :** The key point which is considered to be the centre of the circle of radius eps ϵ and minimum number of border points (*minPts*).
- **Border Point :** All the points in the circle of centre Core point and radius ϵ
- **Noise :** The points out side the circle and all clusters.

Algorithm:

1. Choose a point to be a core point and find all the border points.
2. If the count of border point greater than the minimum required points (*minPts*), add it to the cluster.
3. Continue the same on all the points, accumulating the new points to the cluster.
4. The points not in the circle and in the cluster are considered to be noise.

Listing 5.4: Python code to compute DBSCAN algorithm

```python
now = time.time()
dbscanclustering = DBSCAN(eps=30, min_samples=3).fit(tp)
finalpoints = np.expand_dims(
        dbscanclustering.components_, axis=1)
later = time.time()
dbscandiff = later - now
print(f"Computation Time of DBSCAN : {dbscandiff}")
```

5.2.2 OPTICS

Ordering Points To Identify Cluster Structure (OPTICS) algorithm as the name suggests, is to identity the cluster structure. This algorithm is inspired from the DBSCAN algorithm and requires less number of parameters to compute the similar results as DBSCAN algorithm. In addition to the 3 core components of DBSCAN, OPTICS add 2 more components to its list.

- **Core Distance :** The minimum value of radius required to classify a given point as a core point. If the given point is not a Core point, then it's Core Distance is undefined.
- **Border Point :** The Reachability distance between a point p and q is the maximum of the Core Distance of p and the Euclidean Distance between p and q.

OPTICS algorithm itself doesn't cluster the points instead calculates the reachability distances for all the points. These data is used to develop a visual representation of the clusters and eliminate the points which are not considered to be important.

Listing 5.5: Python code to compute OPTICS algorithm

```
        now = time.time()
        opticsclustering = OPTICS().fit(tp)
        opticspoints = tp[opticsclustering.labels_ > -1]
        finalpoints = np.expand_dims(opticspoints, axis=1)
        later = time.time()
        opticsdiff = later - now
        print(f"Computation Time of OPTICS : {opticsdiff}")
```

5.2.3 Analysis on Density based clustering

In order to clean the features both DBSCAN and OPTICS algorithms are tested sequentially and had shown promising results. The evaluation is done on the basis of time consumptions, complexity, quantity and project requirements. The results and consideration are mentioned in the table 5.2

From this table, it is understood that though DBSCAN algorithm is very fast in computing and also produce less number of points for evaluation. But requires different parameter inputs based on the scene. This algorithm can not adopt the requirement depending on the input points. On the other hand the computational complexity of OPTICS algorithm is low due to less number of parameters and its capability to understand and adopt the scene. After analysing the results from the evaluation, this project uses OPTICS for the density based clustering.

Clustering	DBSCAN	OPTICS
Time	0.002s	0.033s
Complexity	High due to input parameters	Low due to no parameters
Quantity	10	24
Requirement	Low due to multiple scenes	High due to adaptation of scene

Table 5.2: Table to compare different density based clustering algorithms

5.3 Lucas-Kanade Algorithm

Lucas-Kanade Algorithm works on the assumption that the pixel intensities of the object do not change in the consecutive frames and neighbouring pixels have same motion. Considering these assumptions, it is evident that the pixels can be tracked in the consecutive frames and keeping track of Spacio Temporal data of these pixels, the motion of each pixel can be tracked. This project uses Lucas-Kanade algorithm to track the pixels in the video sequences and track the motion of the pixels.

Algorithm:

Consider the pixels $I(x, y, t)$ in the first frame. It moves a distance (dx, dt) in the time dt. As per assumptions the intensities are same and can be written as 5.6

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{5.6}$$

Then take taylor series approximation of right-hand side, remove common terms and divide by dt to get the following equation:

$$f_x u + f_y v + f_t = 0 \tag{5.7}$$

where:

$$f_x = \frac{\partial f}{\partial x}; f_y = \frac{\partial f}{\partial y}; u = \frac{dx}{dt}; v = \frac{dy}{dt} \tag{5.8}$$

We have seen an assumption before, that all the neighbouring pixels will have similar motion. Lucas-Kanade method takes a 3x3 patch around the point. So all the 9 points have the same motion. We can find (f_x, f_y, f_t) for these 9 points. So now our problem becomes solving 9 equations with two unknown variables which is over-determined. A better solution is obtained with least square fit method. Below is the final solution which is two equation-

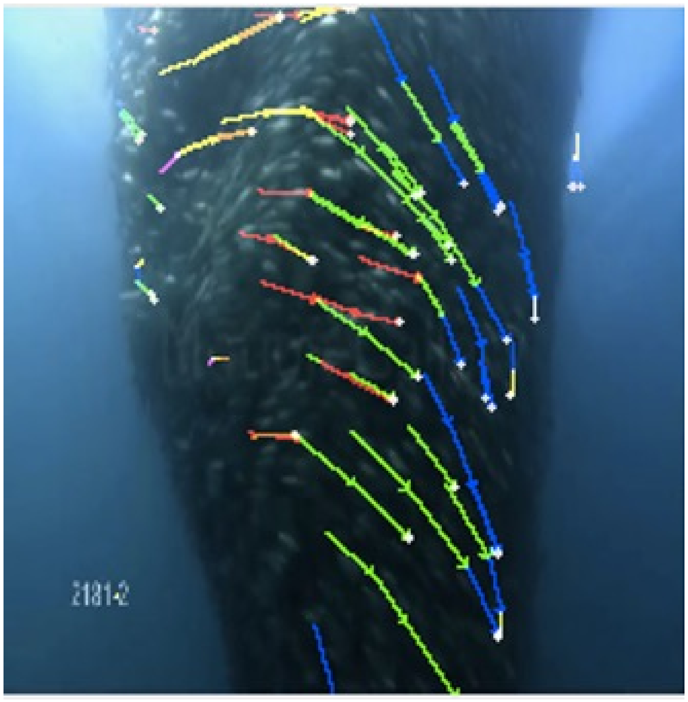

Figure 5.3: Lucas-Kanade Algorithm Implementation

two unknown problem and solve to get the solution.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i f_{x_i}^2 & \sum_i f_{x_i} f_{y_i} \\ \sum_i f_{x_i} f_{y_i} & \sum_i f_{y_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i f_{x_i} f_{t_i} \\ \sum_i f_{y_i} f_{t_i} \end{bmatrix} \tag{5.9}$$

From the equation 5.9, values of u and v are calculated. Using these equations the pixels are tracked from one frame to another. The implementation of the Lucas Kanade algorithm is altered as per the proposed model. The implementation of this algorithm is shown in the figure 5.3

Listing 5.6: Python code to compute Lucas-Kanade algorithm

```
frame = cv2.resize(frame, (224, 224))
frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
# calculate optical flow
p1, st, err = cv2.calcOpticalFlowPyrLK(old_gray,
        frame_gray, p0, None, **lk_params)
good_new = p1[st == 1]
```

6 Implementation

6.1 Model Architecture Diagram

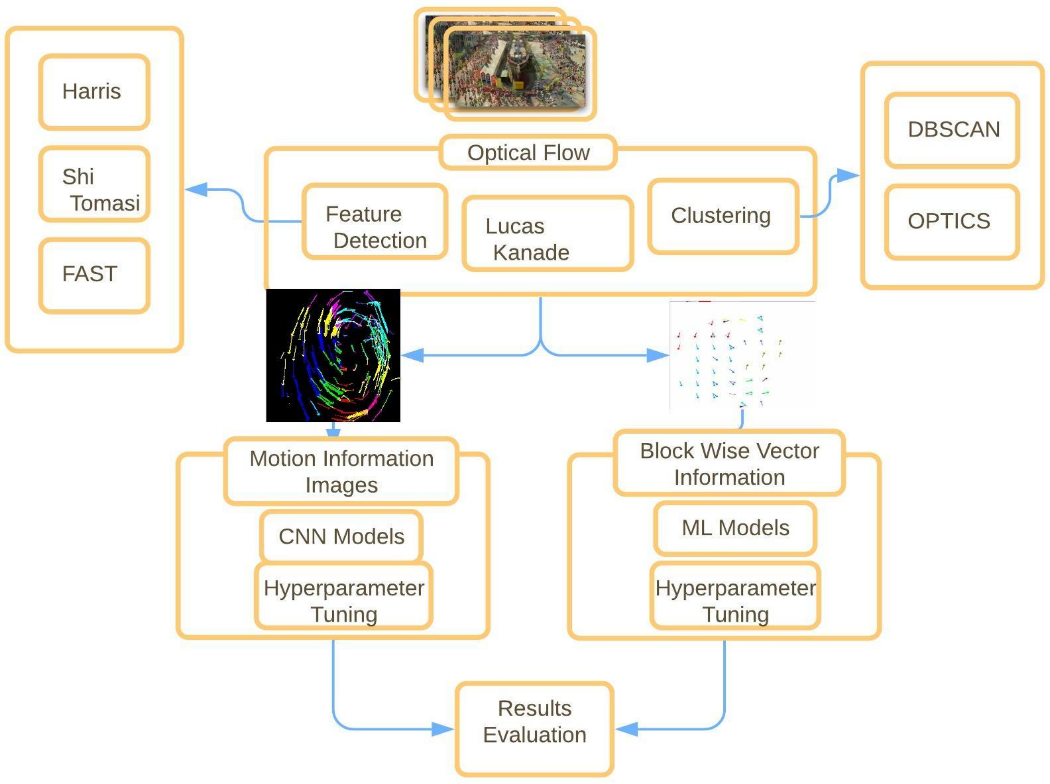

Figure 6.1: Model Architecture Diagram

6.2 Implementation Logic

Implementation logic of the model architecture 6.1 can be divided into 3 parts: Data Flow, CNN and Machine Learning models. Data Flow deals with the frame size, Key frames, Block size, Different optical flow techniques, noise reduction, MIIs generation, Block wise dominant motion detection and generation of input files to ML models. CNNs deal with the different types of CNN networks and hyper parameter tuning. Lastly, Machine learning models deal with the logic to read the block wise dominant motion data, training and testing different classification models.

6.2.1 Data Flow

Optical flow is considered to be one of the widely used technique to track the pixels in video footages. Besides the advantages of using optical flow, it is important to understand the computation process. Ideally, every pixel in the frame is tracked in the consecutive frames and thus the computation time strictly depends on the size of the frame. This can be tricky to compute the performance of the model as the computation depends on the input. In order to tackle this problem, this model propose to reduce the size of the frame to 224*224 pixels before processing.

Length of the video is another considerable factor in optimising the performance of the model. Also the main objective of the model is to track the motion of the objects and classify it. In order to achieve this objective, Spacio Temoral data needs to be tracked and this data consumes large amounts of memory and computation time. In order to reduce the memory consumption and computation time, this model tracks the information of the object in every 5 frames and stores the required information. The code block that implements this logic is shown below.

Listing 6.1: Logic to store data in every 5 frames

```
p1, st, err = cv2.calcOpticalFlowPyrLK(old_gray,
        frame_gray, p0, None, **lk_params)
good_new = p1[st == 1]
tmp_points = tmp_points[st == 1]
if counter == 5:
        counter = 0
        for i, (new, old) in enumerate(zip(good_new,
                        tmp_points)):
        a, b = new.ravel()
         c, d = old.ravel()
```

In order to create the block wise dominant motion data, direction and the magnitude of the object motion need to be calculated. This information is calculated from the old and new points computed from Lucas Kanade algorithm. The directions in frame are in the inverse direction and thus the angles are manually computed on the inverse direction plot. All the directions are grouped into 12 classes where all the vectors pointing in the angle between 0^o to 30^o are clubbed as 15^o. All the direction vectors are clubbed in other directions similarly. Apart from the directional data it is also important to understand the magnitude of the vectors and thus magnitude is also tracked on every vector. The magnitude and directions are computed with the following code.

Listing 6.2: Logic to calculate Direction and magnitude

```python
    def calculateDistance(self, x1, y1, x2, y2):
        dist = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
        return dist

    def calculateAngle(self, a, c, b, d):
        angle = np.degrees(math.atan2(b - d, a - c))
        return angle
```

Every feature to be tracked has the magnitude, direction and the start and end points of the line. This information is segregated in blocks. Blocks are the user inputs which helps the model to divide every frame into multiple blocks in X and Y axis. The implementation logic is capable of changing the number of blocks depending on the user Inputs. A Numpy array is created with the help of user inputs and the data is arranged into multiple blocks. Average of the magnitude and count per direction is calculated in all the blocks. This information is stored in .csv files using pandas data frame.

Noise reduction is done on the data at 2 instances: While performing the optical flow and while storing the block wise dominant motion data. In the first scenario, a threshold magnitude is defined and is used to filter the features. In the second scenario, the maximum value of the product of average magnitude and count of directions is calculated in all the blocks. This value is scaled according to the requirement to filter out less significant data.

The most critical task in the process is labelling the data. due to the unavailability of annotated file on the video datasets, Labelling is done manually on every frame. The Motion information images generated from the optical flow and the block wise dominant motion data calculated from the directions and magnitude was labelled dynamically by taking inputs. Each frame has been labelled with one label and on the whole every frame is classified into 4 classes, Arcs, Lanes, Converging/Diverging and Random/Blocks.

6.2.2 Convolutional Neural Networks

Motion information images are used to train the CNN models to classify the motion. Training the models and testing is done with the help of PyTorch pre-defined modules. The output classes are encoded using One hot encoding in the datasets and number of classified output classes from the model are modified in the PyTorch module as mentioned below.

Listing 6.3: Using PyTorch pre-defined modules

```
model = torch.hub.load('pytorch/vision:v0.6.0',
    'vgg11', pretrained = False)
print(model.classifier[6])
model.classifier[6] = torch.nn.Linear(in_features=4096,
    out_features=4, bias=True)
print(model.classifier[6])
```

The implementation of the CNN in this project is divided into 4 sections: DataSets, Data Loader, Training and Testing. Datasets for the input of the CNN are created with the help of Datasets which is imported fromPyTorch library. Here the data is split into 70-30 ratio for training and testing. The implementation is done on Cuda parallel processing. So, the data is converted into tensors and transformations of the data like size correction and converting the data into normalised form is handled at this point.

In the data loader part, the data is converted into batches for the input to the model. Also, segregated and shuffled training and test data are loaded into training set and test sets. In order to maintain the quality of the model, splitting is done before the training. In the training part, the training data set of 168 images are passed through the model and the loss is calculated with different criterion. The loss is used to train the weights with the help of different optimisers.

As this is a classification problem, the testing part deal with finding the accuracy of the model. Here the testing data set is used to test the model accuracy. The accuracy is calculated by counting the total images and correctly classified images from the test dataset. This process is repeated on different CNN models with different hyper parameters by changing the parameters like learning rate, Criterion and Optimiser.

The images are trained on 3 types of CNN architectures, AlexNet, VGG11 and ResNet101. All these architectures have different variations in the convolution layers. It is considered helpful to check the dependency on the depth of the CNN architecture. AlexNet have 5 convolution layers with a fully connected Layer, making it the smallest architecture compared to VGG11 and ResNet101. VGG11 architecture have 8 convolution layers and is considered to be a deep network. ResNet101 is the very deep network with 3 layers of multiple convolution layers.

6.2.3 Machine Learning Models

7 Results and Discussions

7.1 Dataset

Description of the dataset(s)

7.2 Experimental setup

Say what is the experimental set up, parameters that were used.

7.3 Results

Stand back and evaluate what you have achieved and how well you have met the objectives. Evaluate your achievements against your objectives in Section 1.2. Demonstrate that you have tackled the project in a professional manner.

The previous paragraph demonstrates the use of automatic cross-references: The "1.2" is a *cross-reference* to the text in a numbered item of the document; you do not type it as `1.2` but by using the `\Sec` command. The number that appears here will change automatically if the number on the referred-to section is altered, for example, if a chapter or section is added or deleted before it. Cross-references to section are entered with the `\ref` command just like for figures. The TEX code above reads

```
Evaluate your achievements against your objectives
in section \ref{objectives sec}.
```

For this to work, the code for the text on page **??** must read

```
    \section{Scope and Objectives} \label{objectives sec}
```

As with figure labels, the text inside of `\label` and `\Fig` never appears in the final pdf; you can make it whatever you want as long as you use the same text in each to complete the reference.

7.4 Discussions

Analyse your results and discuss it by including your insight. For example why the results are behaving like this, why there is an outlier etc.

8 Conclusions & Future Work

8.1 Conclusions

Summarise what you have achieved. Again do not say I achieved this. Say what the project has achieved.

8.2 Future Work

Explain any limitations in your results and how things might be improved. Discuss how your work might be developed further. Reflect on your results in isolation and in relation to what others have achieved in the same field. This self-analysis is particularly important. You should give a critical evaluation of what went well, and what might be improved.

Bibliography

[1] G. Sreenu and M. S. Durai, "Intelligent video surveillance: a review through deep learning techniques for crowd analysis," *Journal of Big Data*, vol. 6, no. 1, p. 48, 2019.

[2] G. Tripathi, K. Singh, and D. K. Vishwakarma, "Convolutional neural networks for crowd behaviour analysis: a survey," *The Visual Computer*, vol. 35, no. 5, pp. 753–776, 2019.

[3] D. Bhowmik and A. Wallace, "Statistical t+2d subband modelling for crowd counting," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 1533–1537.

[4] H. Idrees, I. Saleemi, C. Seibert, and M. Shah, "Multi-source multi-scale counting in extremely dense crowd images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2547–2554.

[5] B. E. Moore, S. Ali, R. Mehran, and M. Shah, "Visual crowd surveillance through a hydrodynamics lens," *Communications of the ACM*, vol. 54, no. 12, pp. 64–73, 2011.

[6] F. Santoro, S. Pedro, Z.-H. Tan, and T. B. Moeslund, "Crowd analysis by using optical flow and density based clustering," in *2010 18th European Signal Processing Conference*. IEEE, 2010, pp. 269–273.

[7] C. Direkoglu, "Abnormal crowd behavior detection using motion information images and convolutional neural networks," *IEEE Access*, vol. 8, pp. 80 408–80 416, 2020.

[8] X. Wei, J. Du, Z. Xue, M. Liang, Y. Geng, X. Xu, and J. Lee, "A very deep two-stream network for crowd type recognition," *Neurocomputing*, vol. 396, pp. 522–533, 2020.

[9] B. Solmaz, B. E. Moore, and M. Shah, "Identifying behaviors in crowd scenes using stability analysis for dynamical systems," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 10, pp. 2064–2070, 2012.

[10] B. Galvin, B. McCane, K. Novins, D. Mason, S. Mills *et al.*, "Recovering motion fields: An evaluation of eight optical flow algorithms." in *BMVC*, vol. 98, 1998, pp. 195–204.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[12] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[13] N. Rosandić, T. Petrović, R. Tušek, Z. Kučiš, A. Štruklec, M. Liashuha, and A. Alatas, “Crowd analysis by using optical flow and density based clustering,” 2019.

[14] A. M. Cheriyadat and R. J. Radke, “Detecting dominant motions in dense crowds,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 4, pp. 568–581, 2008.

[15] J. Chen, W. Su, and Z. Wang, “Crowd counting with crowd attention convolutional neural network,” *Neurocomputing*, vol. 382, pp. 210–220, 2020.

[16] Z. Lin and L. S. Davis, “Shape-based human detection and segmentation via hierarchical part-template matching,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 604–618, 2010.

[17] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos, “Privacy preserving crowd monitoring: Counting people without people models or tracking,” in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–7.

[18] S. D. Bansod and A. V. Nandedkar, “Crowd anomaly detection and localization using histogram of magnitude and momentum,” *The Visual Computer*, vol. 36, no. 3, pp. 609–620, 2020.

[19] C. G. Harris, M. Stephens *et al.*, “A combined corner and edge detector.” in *Alvey vision conference*, vol. 15, no. 50. Citeseer, 1988, pp. 10–5244.

[20] J. Shi *et al.*, “Good features to track,” in *1994 Proceedings of IEEE conference on computer vision and pattern recognition*. IEEE, 1994, pp. 593–600.

[21] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *European conference on computer vision*. Springer, 2006, pp. 430–443.

Appendix 1

You may have one or more appendices containing detail, bulky or reference material that is relevant though supplementary to the main text: perhaps additional specifications, tables or diagrams that would distract the reader if placed in the main part of the dissertation. Make sure that you place appropriate cross-references in the main text to direct the reader to the relevant appendices.

Note that you should ***not*** *include your program listings as an appendix or appendices.* You should submit one copy of such bulky text as a separate item, perhaps on a disk.

Appendix 2 – User guide

If you produced software that is intended for others to use, or that others may wish to extend/improve, then a user guide and an installation guide appendices are ***essential***.

Appendix 3 – Installation guide

If you produced software that is intended for others to use, or that others may wish to extend/improve, then a user guide and an installation guide appendices are ***essential***.