Detection of pedestrians and analysis of movement in PETS dataset

Bharath Nagachandra Surampudi
School of Computer Science and Engineering
University of New South Wales
Sydney, Australia
z5251342@student.unsw.edu.au

Chirag Panikkasseril Unni
School of Computer Science and Engineering
University of New South Wales
Sydney, Australia
z5241855@student.unsw.edu.au

Philip Thomas Nedumpuran
School of Computer Science and Engineering
University of New South Wales
Sydney, Australia
z5188873@student.unsw.edu.au

Suchithra
School of Computer Science and Engineering
University of New South Wales
Sydney, Australia
z5230257@student.unsw.edu.au

Vandhana Visakamurthy
School of Computer Science and Engineering
University of New South Wales
Sydney, Australia
z5222191@student.unsw.edu.au

***Abstract*— Real-time object detection and tracking is an immensely vast and important field of Computer Vision. It is an inconclusive and complex application because of its wide usage in reconnaissance and tracking frameworks. It is used in security, automotive systems and numerous other applications. It has pushed researchers and industry experts to constantly devise improved and efficient algorithms. However, issues may occur in implementing object detection and tracking, where the environment and the domain under consideration changes with each situation, which makes the problem not only difficult but also costly for computation. Nevertheless, numerous methods and systems have been developed over the years to tackle various real-world challenges. In this project we will be examining and implementing few of the hand-crafted concepts and also state of the art systems for object detection and tracking. We aim to provide the outcomes of these techniques and compare them for their effective uses against the prescribed tasks with the available evaluations and results.**

I. Introduction

Computer Vision, in the past decade, is one of the most important fields of study. With the advancements in Artificial Intelligence and Pattern Recognition, the demand for Computer Vision and its applications has seen immense increase. Computer Vision is the field of study that deals with the ability of a system to perceive its environment and interpret the visual details through feature extraction. One of the major applications in this field is pedestrian detection. Pedestrian Detection is widely used in the automobile industry and for surveillance. In fact, it has helped high-end automobile manufacturers come up with not just self-driving cars but develop and deploy advanced braking systems that do not need human interference to reduce impact during a collision and prevent it altogether. Pedestrian Detection has also improved surveillance and monitoring to improve safety. This is achieved through object detection and tracking. Apart from detecting and tracking pedestrians, object detection and tracking methods help monitor every object that is in some form of motion. There are a variety of object tracking and detection algorithms and the objective of this project is to analyse the various methods and implement an advanced system that can detect and track pedestrians in the given PETS dataset with high precision and visual accuracy.

Object detection is the study of identifying features of an image to predict the class and type of an object while object tracking refers to monitoring the trajectory or path of motion of an object. A combination of object detection and tracking enables systems to perceive, analyse and monitor various instances of an image. The various methods used to detect objects in this project are HOG+SVM (Histogram Oriented Gradients + Support Vector Machines), Haar Cascades, Contour Detection, Image Subtraction algorithm, YOLO (You Only Look Once). These methods are analysed, and the best detection method is incorporated with tracking algorithms like SORT (Simple Online Real Time Tracking) and Kalman filters to build a wholesome pedestrian tracking algorithm. A centroid based tracker is used instead of the above mentioned methods. The novel features of this system include its ability to count the number of pedestrians in every frame in the given sequence of frames and to track individual trajectories. The system is also trained to be occlusion resistant and perform fairly in detecting the number of pedestrians moving in groups and individually.

II. Ground Truth and Datasets

A. Ground Truth

The ground truth defines detecting pedestrians by highlighting the presence of pedestrians using rectangular bounding boxes. [2] Tracking of detected pedestrians is being involves being able to predict the trajectory of movement. The ground truth for detecting and tracking is established using an object detection method that can track all pedestrians in the foreground and generate the count of the number of pedestrians in every frame [2]. When moving in groups the system should be able to differentiate pedestrians individually and perform efficiently when occlusion occurs. The detection method that's able to validate the ground truth is deemed the most efficient solution for the given problem statement. Errors during occlusion reduce the accuracy of detection and tracking.

B. Dataset

The dataset for the given project was provided by the eleventh IEEE international workshop on Performance Evaluation of Tracking and Surveillance (PETS 2009). It consists of a sequence of images that were extracted from a video. There are a total of seven hundred and ninety five frames that consist of multiple pedestrians walking individually and in groups. The dataset consists of frames that can easily detect pedestrians and frames where there is high occurrence of occlusion. This poses a challenge in building a system that can efficiently handle the rate of occlusion.

III. Literature Survey

A. Object Detection

Object detection is the study of identifying features of an image to predict the class and type of an object while object tracking refers to monitoring the trajectory or path of motion of an object. A combination of object detection and tracking enables systems to perceive, analyse and monitor various instances of an image. The various methods used to detect objects in this project are HOG and SVM (Histogram Oriented Gradients - Support Vector Machines), Haar Cascades, Contour Detection, Image Subtraction algorithm, YOLO (You Only Look Once). These methods are analysed, and the best detection method is incorporated with tracking algorithms like SORT (Simple Online Realtime Tracking) and Kalman filters to build a wholesome pedestrian tracking algorithm. The novel features of this system include its ability to count the number of pedestrians in every frame in the given sequence of frames and to track individual trajectories. The system is also trained to be occlusion resistant and perform fairly in detecting the number of pedestrians moving in groups and individually.

B. Multiple object tracking

Multiple object tracking is an extensively researched field in computer vision. The academic and practical applications of the field are numerous, and it is widely implemented in technologies which are of commercial value. The topic plays a critical role in systems involved in applications such as traffic monitoring, defence, surveillance technologies, automated driving systems, robotics, etc.

The tracking usually involves locating the object, identifying individual objects, tracking the motion and trajectory, classifying the object and its actions. Some applications involve fast tracking especially in capturing fast moving objects in fields of science, defence or sports. Some trackers focus on accuracy especially in surveillance, reconnaissance. These objectives differ with different domains and its applications. Over the years a wide range of research has gone into developing pedestrian tracking models based on state-of-the-art MOT models along with Neural network-based Object detection systems.

Key criteria the single object tracking SOT or MOT take into consideration when designing the solution for each application are the way it handles frequent occlusions, initialization and termination of tracks, interaction among multiple objects and similar appearance of objects. Various solutions such as Appearance model, Motion model, Interaction model, Exclusion model has been developed over the years in the MOT field.

The inferences of the model are derived by using approaches such as probabilistic methods which involve filters like Kalman filter, particle filter or Deterministic Optimization approach which uses algorithms like dynamic programming, bipartite graph matching or conditional random fields.

C. Histogram of Oriented Gradients (HOG)

A wide range of research [1],[2] has gone into HOG, or Histogram of Oriented Gradients, in the computer vision field especially in the pedestrian detection application. HOG feature descriptor counts the occurrences of gradient orientation in localized portions of an image focusing on the structure or the shape of an objects in the image. HOG provides edge direction by extracting the gradient and orientation of the edges where orientations are calculated in 'localized' portions. The complete image is broken down into smaller regions and for each region, the gradients and orientation are calculated, and a histogram is generated for each of these regions respectively. The histograms are created using the gradients and orientations of the pixel values in the image

D. Linear Support Vector Machine Classifier

Support Vector Machine is a supervised machine learning algorithm where labelled training data, the algorithm outputs an optimal hyperplane which classifies the testing image examples. SVM has different kernel versions like linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid. Linear kernel maximizes the margins from nearest training point. To get the best performing kernel we can use grid optimization technique. Once training the model using the training data, we can test the model for testing data and get evaluation metrics to quantitate the performance of the model.

E. You only Look Once(YOLO)

You only Look Once or YOLO is a convoluted neural network-based object detector and classifier. The idea was to produce a model which does detection classification and locate the object in the referenced frame by adding a bounding box around it. This is done at one go for the YOLO system and that makes it fast and efficient.

YOLO v1 was introduced in 2016 May which introduced the core of the algorithms and further improvements were made in later versions which took care of the drawbacks which had. It was inspired by the GoogleNet architecture of implementing convoluted layers of neural networks. The initial version had 24 convolutional layers which functioned as feature extractors and 2 dense layer which was employed to do classification of the

features. The architecture used was darknet developed by Joseph RedMon. YOLO v2 has some improved functions which can do feature extraction over fine grained images, smaller objects and detect object which are nearby. YOLO v3 the current model which uses the improved darknet architecture now holds 53 convolutional layers

Three main features of YOLO is that it is extremely fast. It can very well process at 45fps rate and faster version uses over 15 fps rate. Since YOLO is exceptionally generalizable it is less inclined to dysfunction when applied to new areas or unexpected data sources. This makes YOLO outperform some of the other widely used systems like R-CNN or DPM especially when it comes to detection of natural images or artworks. Although it lacks accuracy in comparison with other popular neural network systems it is ideal for applications that rely on fast, robust object detection.

F. Haar cascades

Haar cascades are feature extraction frameworks that are used to detect the presence of an object in a frame. Image intensities and their histograms are traditionally used to extract features of an image. Methods like HOG and contour detection used pixel densities and are proven successful. However, these methods are computationally expensive as they repeatedly scan the pixels in a region-to-region manner. This was a drawback that was repeatedly brought up by the scientific community. To reduce the computational Complexity, Viola and Jones proposed to use Haar-like features (later known as Haar features) to identify regions of varying pixel intensities. This method was developed by calculating the sum of the pixel intensities of a region and finding the differences between the sum until the entire image is covered. The difference of the sum of pixel intensities helps categorize the various sections of the image. Regions of greater pixel densities will stand out and can be recognized by the cascade classifier in OpenCV. The pedestrian detection cascade is used in this case and the hyperparameters like scaleFactor and minNeighbors are adjusted to ensure all relevant objects in the frame are detected.

G. Contours

Contours are boundaries or the path of points that define the outline of an object. Contours of objects in images are detected using the inbuilt OpenCV method. The boundary points of every object are detected and stored in a hierarchical manner. These boundary points are calculated using chain approximation rule by splitting the image into the foreground and background and isolating objects from their backgrounds. These contours can be visualized using draw contour function in OpenCV.

H. Image subtraction

The image subtraction algorithm is used to highlight and obtain the features of moving objects by subtracting the background of two subsequent frames to produce the image difference. The difference between these two frames is obtained using the absolute difference to highlight only objects in motion. The output can be further highlighted using thresholding and noise reduction by using a low pass filter like Gaussian Blur.

IV. Method

A. Object Detection using You Only Look Once

The YOLO approach uses the CNN neural network architecture based on YOLOv3 pre trained over Darknet architecture. The model can predict around 80 classes as per the latest update and for our scope we will be using the pedestrian or person class for implementation of various tasks. In the experiment we use pretrained weights and configurations for the model. Here we will be discussing the implementation of the framework and the corresponding output vector obtained.

1) The Prediction Vector

The first step involved in the object detection is dividing the object into S x S grid of cells. For an object present on the given image a cell is going to be the "responsible cell which gives the centroid of the object which describes its location. For every grid cell a prediction of *B* bounding boxes and *C* class probabilities are computed. The main components involved in bounding box prediction are components: *(x, y, w, h, confidence)*. The center of the box identified is given by the *(x, y)* coordinates, relative to the grid cell location (remember that, in case if the centroid thus calculated doesn't fall into the grid then it is not identified as the "responsible" grid). The location *(x, y)* coordinates and the *(w, h)* coordinates is then normalized in the range of [0,1] with respect to the image scale. The parameter confidence is defined as

$$\Pr(\text{Object}) * \text{IOU}(\text{pred}, \text{truth}) \qquad \text{eq [1]}$$

The confidence score should be zero when there is no object in the corresponding grid. Otherwise the confidence score is calculated by computing the intersection over union (IOU) between the predicted box and the ground truth. It uses logistic regression method for computing the bounding box prediction

The conditional probabilities *Pr(Class(i) | Object)* is then computed. The loss function used in the neural network will not penalize the classification on wrong class probabilities as it only predicts one set of class probabilities per cell regardless of the bounding boxes B. Thus, the computation provides a total of *S x S x C* class probabilities. Therefore, the output vector will give you *S x S x (B * 5 +C)* tensor as output.

The prediction is also done over three different scales and then over multiple labels based on the confidence matrix arrived in the computation of bounding box classification.

2) The Network

The network structure involves a CNN for prediction that comprises of convolutional and max pooling layers, followed by 2 fully connected layers in the end. YOLO v3 uses Darknet variant which originally has a network of 53 layers. 53 more layers are stacked onto it for the detection task, giving us a 106-

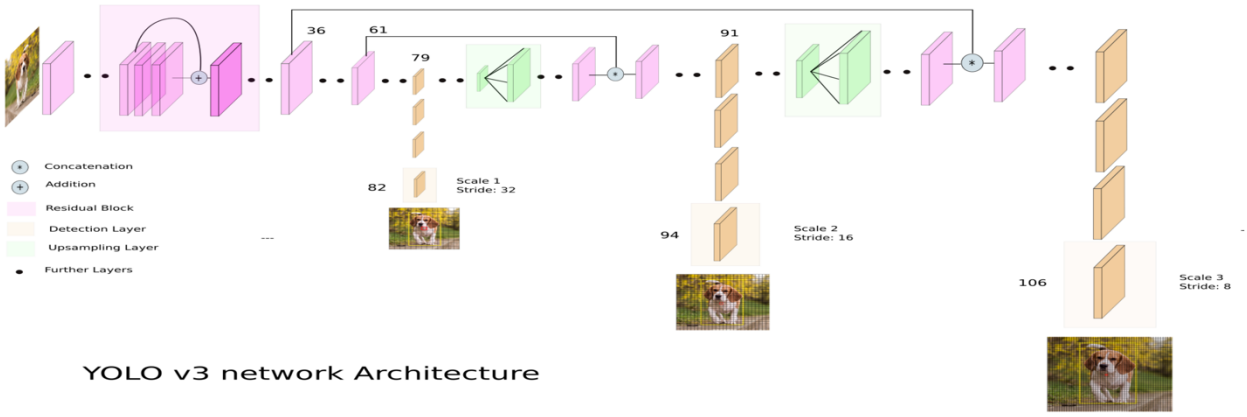


Figure 1. Architecture of YOLO v3.

layer fully convolutional architecture for YOLO v3. The network uses successive 3 × 3 and 1 × 1 convolutional layer.

Applying a 1 x 1 kernel on a function map will generate the eventual output. In YOLO v3, the detection is achieved by applying 1 x 1 detection kernels on three-size feature maps at three different locations in the network. The newer architecture features residual skip and up sampling connections. The kernel form for detection is 1 x 1 x (B x (5 + C)). In case of YOLO v3 on COCO dataset B is equal to 3 and C is 80 which forms a kernel size of 1 x 1 x 255

B. *Object Detection using HOG and SVM*

General approach for an object detection model usually employs a two-step process which involves feature extraction method and classification of the detected objects. Feature detection is the process of extracting characteristics of an image like edges, lines, and contours to define or make sense of the image. It serves as the foundation of object detection. The feature extraction can be done by classic computer vision techniques based on gradient, shape, texture, motion or part-based feature extraction and also by using advanced neural net-based Machine learning algorithms. The extracted features from the image or video is identified by the approach. The observed data is then pre-processed to remove other noises and blur effects and later used on a classification model to train the model on various aspects. The trained model is then applied to identify or classify the objects in the region of interest. A bounding box is then generated around the identified pedestrian features in the given image or the video frame.

In this method we approach the task as a pipeline of objectives. First part of the task is to do feature extraction where we will be using Histogram oriented Gradient method for the extraction of features and then do classification of the objects in the image. For implementing classification, we will be applying Linear support vector machine classifiers. The trained model will then be used to detect objects in the image, and we will be using a threshold to obtain pedestrians from the image. The observation may produce multiple bounding boxes over the same pedestrian therefore to overcome the problem we apply non-maximum suppression to get the final bounding boxes over detected pedestrians in the referenced images.

1) *Histogram Oriented Gradient*

HOG is a feature descriptor that is used to extract features from image datasets based on the gradients formed. The feature extraction technique detects the edges by extracting gradient and the orientation. The technique also focuses on the shape and the structure of the object in the image. It uses a localized portion approach on calculation of the features. That is, it takes a full image and breaks it down to smaller image regions and for each region, calculation of gradient and orientation is computed. Separate Histograms will be computed for each of these localized regions in the image i.e., for each such regions a 1-D histogram of edge orientations or gradient directions will be computed over the pixels of the region using orientations and gradients technique and hence the name or Histogram of Oriented Gradients

2) *Linear Support Vector machine*

Linear Support Vector machine (SVM) algorithm creates a hyperplane or a line which separates the data into appropriate classes, in this case detecting the presence of pedestrians in the region of interest. Linear SVM finds the line separating the data into pedestrian or no pedestrian from the data obtained from feature extraction. The hyperparameters which are optimized by tuning are C value which controls the trade-off between classifying training points correctly with high C and smooth decision boundary with low C and the Gamma values which defines how far the influence of a single training reaches as in low value means far reach and high means close reach. SVM helps in identifying and classifies the data points based on the features. This is done by finding points closest to the classes. These points are support vectors and the distance between them is called margin. The Hyperplane obtained by getting maximum margin upon computation is the optimal hyperplane. The optimal hyperplane provides the model the

decision boundary between the classes with maximum separation. In this way the model detects the presence of pedestrians.

3) Non-maximum suppression

The object detection performed by HOG and SVM model may produce multiple bounding boxes over the same object. Non-max suppression is a way to ensure each object is only detected once by the algorithm. For the algorithm we take a threshold and compare it with the localized predicted box probabilities and discard some of the boxes. Later for every bounding box for each classification the algorithm search for boxes with max probabilities and discard the non-maximal probabilities. Thus, it provides a single bounding box and hence the name non-maximum suppression.

C. Contour Based detection

The images are first pre-processed to detect the presence of contours. Each frame is initially processed using a low pass filter. In this case, Gaussian Blur is used with a 5x5 kernel. Then the image is processed using binary threshold with minimum pixel intensity as 20 and maximum pixel intensity as 255. This is followed by dilation which is set for three iterations. The frame is now ready for detection. Contours are calculated using the findContours function available in OpenCV toolkit. The function is then called to detect the available contours in the frame. The contours are calculated by dividing every frame into multiple layers and hierarchically searching for pixel regions that make sense. This is achieved by parameters like cv2.RETR_TREE and cv2.CHAIN_APPROX_SIMPLE. cv2.RETR_TREE stores features in a tree like manner, to keep track of reference points that exist in previous, next and current. cv2.CHAIN_APPROX_SIMPLE stores only boundary points of the contour instead of every point in the contour. These boundary points are accessed as a tuple and used to draw the rectangular bounding box for detection.

D. Image subtraction based detection

The image subtraction algorithm is used to highlight and obtain the features of moving objects by subtracting the background of two subsequent frames to produce the image difference. The difference between these two frames is obtained using the absolute difference to highlight only objects in motion. The image is converted to grayscale and then split into background and foreground. This is done by setting threshold values. Once the foreground is obtained, all the objects in the foreground are obtained by searching for connectivity. This is obtained by looking for centroids using cv2.connectedComponentwithStats function. Upon obtaining the centroids, the bounding box can be drawn. This helps detect pedestrians efficiently. Figure 2 visually explains how the foreground and background objects are separated and detected.

E. Detection using Haar Cascades

Haar Cascades are open source xml files that can be used to detect people, faces, number plates, and much more. The

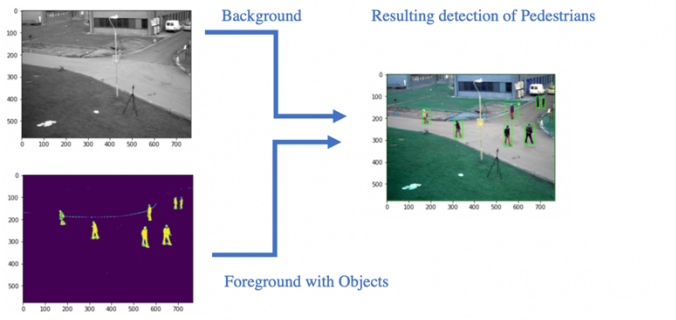


Figure 2. Image Detection using Image Subtraction Algorithm

pedestrian.xml file was obtained from the opensource library and processed using the cascade classifier function available in OpenCV. The frames are then sent sequentially to detect pedestrians using the DetectMultiscale function. This function retrieves all objects that have connected components or contours. These values are accessed as a tuple (x,y,h,w) and used to draw the rectangular bounding boxes for detection.

F. Tracking using Centroids of an Object

The centroid of an object can be obtained using YOLO and these centroids are used as reference points along with the boundary points of objects to draw the rectangular bounding box. The centroids are marked with a unique ID for every object and tracked when the object is in motion. Each centroid is also assigned a unique color that enables viewers to easily differentiate between two pedestrians. This method seemed more relevant to use with YOLO than using SORT or Kalman Filters. This proved efficient and accurate by validating the established ground truth and hence seemed unnecessary to explore any other option.

G. Counting and analysing contruction and destruction of groups

The centroids obtained during detection are used to calculate the distance. The difference of centroids provides a good distance estimate. The distance calculated in this case is the Euclidean distance. A threshold distance of 55 is used to judge if the pedestrians are walking in a group or not. The count obtained is projected in the frame. The logic works efficiently for the given dataset. It detects group formation and destruction and the count of number of pedestrians and the total helps obtain the number of pedestrians that are not in a group. The same was implemented using clustering but failed to provide good results.

V. Experimental Setup

A. Environment and Tools:

The detection component of the assignment was developed on the Anaconda platform using Python 3.7 and OpenCV version 4.2.0 for object detection techniques like HOG+SVM, Image Subtraction, Contour Detection, Haar Cascades, and YOLO. The various libraries used for these methods include

numpy, matplotlib, and cascades like pedestrian.xml from the Haar cascade library(open source). Object detection using HOG and SVM used tools in sklearn and OpenCV functions. The functions with optimum hyperparameters served as an efficient detector(include hyperparameter values).The cascade classifier function is used to process open source cascades like Haar cascades and Viola Jones cascades. Image Subtraction algorithm and Contour Detection made use of tools like thresholding and low pass filters in OpenCV. YOLO is an open source model that enables people to detect objects and label them. The model is built on TensorFlow and uses coco.txt to label all objects it detects.

B. Task1

The experiment uses a python-based solution to detect and classify pedestrian in the video frame using above methods. The task also obtains the trajectory of identified pedestrians and the total count of moving pedestrian in all the frames

1) Detection of pedestrians on the image

a) HOG based detection

For this method we mainly use `skimage.feature` library's `hog` function for feature extraction with the following parameters

```
hog(portion, orientations=9, pixels_per_cell=(8, 8), cells_per_block=(2, 2), block_norm="L2")
```

where portion is the split portions of the video frame individually supplied to the function for feature extraction. The obtained data is then stored in features and then supplied to a pretrained model which uses LinearSVC classifier in `svm` function from `sklearn` library. The parameters used in the pretrained model are as below

```
svm.LinearSVC(C=0.01,max_iter=1000,
class_weight = 'balanced', verbose = 1)
```

The trained model uses C as 0.01 gamma is default (auto), class weight is balanced and max iteration used is 1000.

The result obtained by fitting the model is used to find the confidence or the probability of obtaining the object in the corresponding portion in the supplied frame. The is then parsed to `appendRects(i, j, confidence, count, rects)` to obtain bounding boxes.We use nms function to avoid the duplicate bounding boxes formed in the above step. Nms function uses the non-maximum suppression logic to produce the final detected bounding boxes and the bounding boxes thus obtained is stored in `nms_rects` variable. The function has two separate sub functions one handles the overlap area and the other check for the confidence in these areas. The threshold range between [0,1] and for our experiment we set the threshold value of probability at 0.4 to obtain optimal results. And the results will give the bounding box around the detected pedestrians.

b) YOLO based detection

The YOLO uses pretrained weights 'yolov3.weights' and configuration 'yolov3.cfg' for the detection and the classes in the coco.names file for labelling. For our detection we set the nms threshold as nmsThreshold = 0.40 same as before. The image portion size is set as inpWidth = 416 and inpHeight = 416. Although YOLO can classify 80 objects, we only consider the pedestrians in image and consider label name 'person'. We set the ignore_thresh as .7 which only considers detected objects with confidence value greater than the set threshold. The configuration for the neural net also set parameters decay=0.0005, saturation = 1.5, exposure = 1.5, hue=.1, learning_rate=0.001, for the training purpose. Further parameters such as batch_normalize, filters, size, stride, pad, activation are also set for the individual convoluted layers, shortcut layers and the upsamble or downsamble layers in the network architecture separately.

We feed the frames of images to the yolo net which is initialized in the program. The iterative training and classification inside the network produce the corresponding output vectors for each frame. The output of the yolo contains the confidence value, scores, class id, label, and the detection can be used to get the bounding box coordinates. The boxes variable obtained from these coordinates can be used to draw the bounding box around detected pedestrians.

c) Contour based detection

We use openCV functions to get the detected objects. The images are loaded and consecutive frames are used to get the background using the absdiff function. The diff is applied gaussian blur. The blur is applied in threshold function in OpenCV with range parameters as

```
cv2.threshold(blur,20,255,cv2.THRESH_BINARY)
```

The threshold thus obtained is applied over dilation function and later used in contour function to obtain the detections. The parameters used here are based on the chain approximation method built in inside OpenCV module.

```
cv2.findContours(dilated,cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
```

The contours can be used to get the bounding box coordinates

d) Image subtraction based detection

The image subtraction algorithm is used to highlight and obtain the features of moving objects by subtracting the background of two subsequent frames to produce the image difference. The difference between these two frames

is obtained using the absolute difference to highlight only objects in motion. The image is converted to grayscale and then split into background and foreground. This is done by setting threshold values. Once the foreground is obtained, all the objects in the foreground are obtained by searching for connectivity. This is obtained by looking for centroids using cv2.connectedComponentwithStats function.

e) Haar cascades based detection

The haar cascade utilize CascadeClassifier function from openCV and the configuration for the cascade for the experiment is passed as pedestrian.xml. The function is initialized as cv2.CascadeClassifier('pedestrian.xml') we obtain the detected pedestrian from the function cascade.detectMultiScale(gray,1.3,2). The bounding box coordinates can be obtained from the result of the above function

2) Finding Trajectory of the detected pedestrians

For this we use the bounding box locations from the detection algorithm and obtain the centroid of the detected object. For each frame we trace the centroids of all the objects from initial to end frames. This will form the trajectory of the pedestrian obtained.

Using the centroids of the bounding boxes as the reference point, the pedestrians are marked with an unique object ID. Each centroid is then stored and retrieved for previous frame and current frame. The centroids are mapped from previous frame to current frame using a function called cDist() (Available in sklearn). If a person P1 has centroid c1 for previous, then the centroid c2 for P1 in current, is calculated to be within a distance of 150 which is set as the threshold. The unique ID for a pedestrian lasts for 75 frames by default to ensure that the centroid is not lost during tracking when the pedestrian remains stationary for a long time. The number of persons or centroids in every frame contributes to the count of pedestrians for every frame.

3) No of persons in the frame

The total number of pedestrians in the frame are obtained using YOLO by calculating the number of people class objects in the frame. Counting the number of objects with the label is one way of counting used. The other is counting the number of centroids available from step 2 of finding trajectory.

C. Task 2

The centroids of the pedestrians are obtained from the bounding boxes from Task 1 - YOLO. Each centroid is marked by an unique ID and color coded for better visualisation. The bounding box is drawn using user defined coordinates to create ROI. The number of pedestrians in the ROI is obtained by repeating Task 1 count function. The no. of pedestrians who enter and exit the bounding box is supplied in the bottom left corner using centroid tracking. The no. of centroids in the ROI provide the count.

D. Task 3

1. The centroids obtained during detection are used to calculate the distance. The difference of centroids provides a good distance estimate. The distance calculated in this case is the Euclidean distance.
2. A threshold distance of 55 is used to judge if the pedestrians are walking in a group or not. The count obtained is projected in the frame.
3. The logic works efficiently for the given dataset. It detects group formation and destruction and the count of number of pedestrians and the total helps obtain the number of pedestrians that are not in a group
4. The same was implemented using clustering but failed to provide good results.

VI. Results and Analysis

The team went on to implement five different detection methods and found YOLO to be the most efficient. This was established by visually inspecting the results of detection and using a set of evaluation factors to understand which method aligned best with the ground truth. The evaluation factors included detecting pedestrians individually and in a group, detecting all the pedestrians in the frame, and detecting pedestrians efficiently during occlusion. The results of the various detection methods are tabulated below.

Table 1. Comparison of various detection methods

EVALUATION FACTORS	HOG + SVM	CONTOUR DETECTION	IMAGE SUBTRACTION ALGORITHM	HAAR CASCADES	YOLO
DETECTING INDIVIDUAL PEDESTRIANS	YES	YES	YES	YES	YES
DETECTING PEDESTRIANS IN A GROUP TOGETHER	YES	YES	YES	NO	YES
DETECTING ALL THE PEDESTRIANS IN A FRAME	YES	NO	YES	NO	YES
DETECTION DURING OCCLUSION	NO	NO	NO	NO	YES

From the results, YOLO has proven to satisfy all the evaluation factors. The team decided to proceed with YOLO. Next the basic detection system was used to obtain centroids to track the trajectory of the pedestrians. The number of centroids or the number of objects with the class label person provided the real time count of the number of pedestrians.

The accuracy of counting pedestrians and establishing their centroids are tabulated as below.

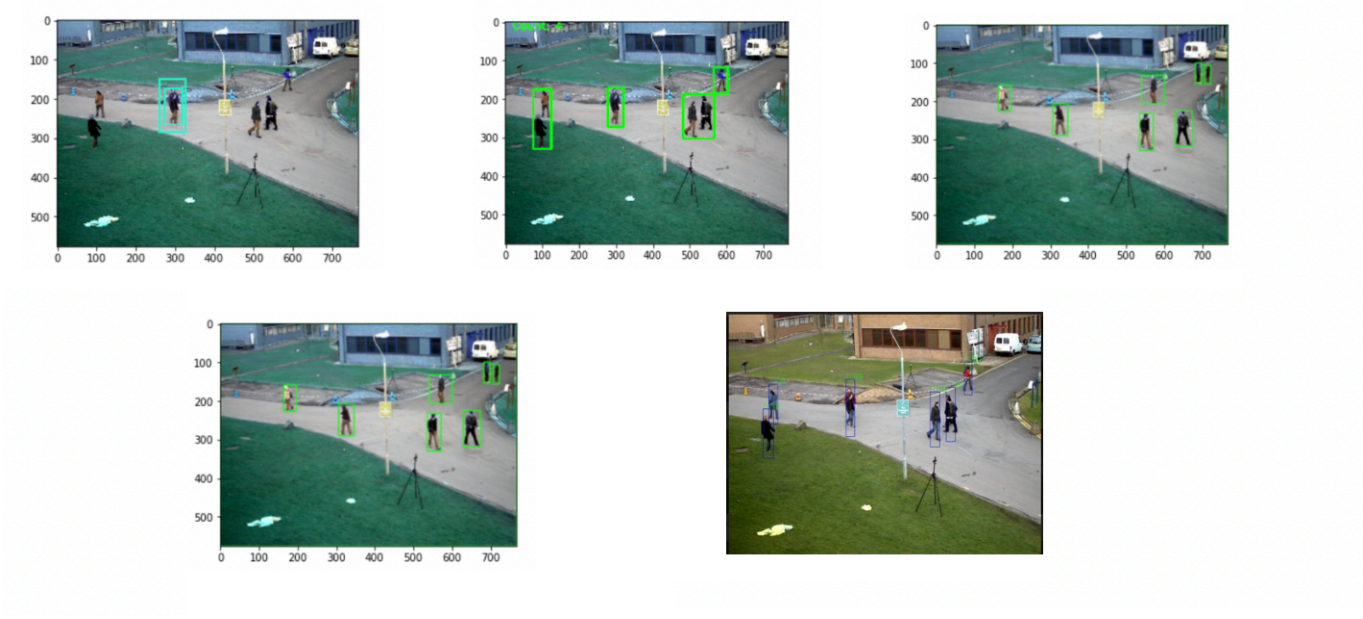


Figure 3. Accuracy of detection methods by visual evaluation using Haar cascades, Image Subtraction, Contour Detection, Hog+SVM, and YOLO. (From left to right clockwise)

Figure 4. Tracking and Counting pedestrians in the frame sequence using YOLO for detection. (Left- high occlusion and right – tracking multiple pedestrians.)

Figure 5. Counting pedestrians in the bounding box frame sequence using YOLO for detection. (Left – entering detection and right – count after multiple enters and exits.

Figure 6. Counting pedestrians in the frame sequence using YOLO for detection. (Left- clustering and right – centroid distance mapping).

Table 2. Evaluation of Accuracy of YOLO

Total number of pedestrians	4744
Total number of pedestrians detected	4895
Accuracy	0.9691

Pedestrian tracking and counting is evaluated by visual inspection of frames. The pedestrians that are detected will be assigned a centroid. Each centroid has been color coded. The path or trajectory follows the centroid until the pedestrian leaves the frame by unique identification and centroid distance mapping. The centroids are accurately mapped from one frame to another. The trajectory drawn for each centroid follows the respective pedestrians accurately. The results are provided in Figure 5. On rare occasions, the centroids might swap during occlusion. But this occurs only rarely in entire sequence of 795 frames. Thus, this method validates the ground truth and works efficiently.

For Task 3, clustering and centroid based grouping were implemented. The results are provided in Figure 6. As shown, clustering doesn't group pedestrians walking together accurately. However, centroid based grouping was able to group centroids whose Euclidean distance was within the defined threshold. The error rate for centroid based grouping was lower using visual inspection.

The results for Task 3 can be further improved by using unsupervised clustering. Task 2 can be improved by using Kalman filter and DEEPSORT techniques with the right hyper parameter settings. These areas were explored but not implemented as the above-mentioned methods work efficiently. Also, other quantitative approaches could have been used to effectively validate the ground truth like Intersection of Union.

VII. Conclusion

Thus, the given PETS dataset is used to detect, track and count the number of pedestrians accurately using YOLO, centroids based tracking, and centroid based grouping. Task 2 was implemented using the functions in Task 1 and by defining a region of interest. Task 3 was implemented using the centroid grouping method. The results also validate the defined ground truth.

VIII. Contribution of Individual Team Members

Each member of the team went on to implement a method of detection mentioned in the report. Then the team went on to use YOLO and build the rest. The work was shared equally and everyone contributed as they were required to do so.

(A) Bharath Surampudi
Bharath implemented the YOLO framework for basic detection. He went on to explore hyper-parameter tuning and redefining threshold values for definition. He wrote the introduction and abstract of the report and contributed to material for YOLO. He also took the responsibility of preparing the presentation slides.

(B) Chirag Unni
Chirag implemented HOG+SVM for pedestrian tracking and assisted Philip with Task 3 using centroid based grouping. Chirag helped develop the bounding box counter as well. He also wrote the literature survey and the respective portions of method and experimental setup which he worked on in the report.

(C) Philip Thomas
Philip implemented HOG+SVM with NMS for pedestrian tracking and went on to develop a pedestrian counter for group formation and destruction using centroid based method in Task 3. Philip wrote the respective sections of his work in the report and assisted others as and when required.

(D) Suchithra
Suchithra implemented Image Subtraction for pedestrian detection and went on to use YOLO to develop the centroid based tracking model. She also mapped the trajectory of the pedestrians. Suchithra also developed the bounding box counter for Task 2 with the assistance of Chirag and Bharath. She also

wrote the respective sections of the report that she helped develop.

(E) Vandhana Visakamurthy

Vandhana implemented Haar Cascade based detection and Contour Detection methods in Task 1. She also worked on clustering of centroids in Task 3 with Philip and assisted others when required. She took the responsibility of writing portions of the literature survey, method, experimental setup for the report along with result analysis and obtaining frames for the visual analysis of each method.

REFERENCES

[1] G. Overett, L. Petersson, N. Brewer, L. Andersson and N. Pettersson, "A new pedestrian dataset for supervised learning," 2008 IEEE Intelligent Vehicles Symposium, Eindhoven, 2008, pp. 373-378.

[2] Ferryman, J. & Shahrokni, A. PETS2009: Dataset and challenge. In 11th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS), 2009.

[3] Luo, Wenhan & Xing, Junliang & Milan, Anton & Zhang, Xiaoqing & Liu, Wei & Zhao, Xiaowei & Kim, Tae-Kyun. (2017). Multiple Object Tracking: A Literature Review.

[4] X. Yuan, L. Cai-nian, X. Xiao-liang, J. Mei and Z. Jian-guo, "A two-stage hog feature extraction processor embedded with SVM for pedestrian detection," 2015 IEEE International Conference on Image Processing (ICIP), Quebec City, QC, 2015

[5] Dalal N., Triggs B. Histograms of Oriented Gradients for Human Detection; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; San Diego, CA, USA. 20–25 June 2005

[6] Witten, I. H., Frank, E., Hall, M. A., Pal, C. J.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann (2016).

[7] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016.

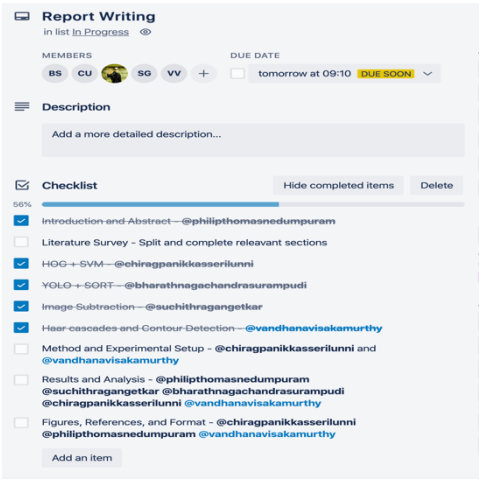
[8] A. Bewley, Z. Ge, L. Ott, F. Ramos and B. Upcroft, "Simple online and realtime tracking," 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, 2016, pp. 3464-3468.

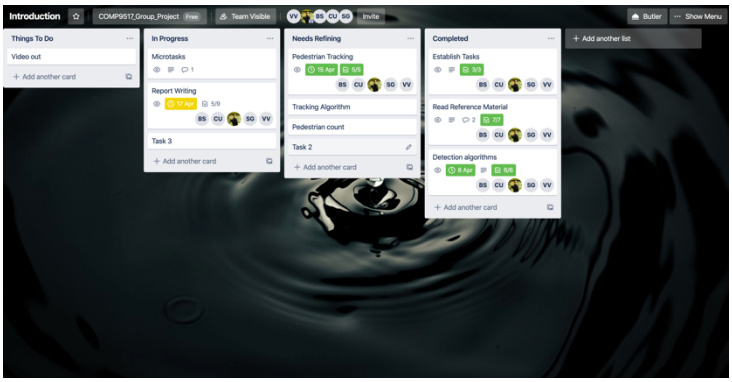
[9] Kothiya, Shraddha V., and Kinjal B. Mistree. "A review on real time object tracking in video sequences." In Electrical, Electronics, Signals, Communication and Optimization (EESCO), 2015 International Conference on, pp. 1-4. 2015.

[10] R. G. J. Wijnhoven and P. H. N. de With, "Fast Training of Object Detection Using Stochastic Gradient Descent," 2010 20th International Conference on Pattern Recognition, Istanbul, 2010, pp. 424-427

Appendix

Trello Boards and task distribution among team members were achieved by splitting, assigning tasks and marking it when done.





Github link - https://github.com/philiptn001/COMP9517_Group_Project