ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

Object Tracking using End-to-End Detection and Deep Association Metric

Authors

Arowa Yasmeen - 160041073

Fariha Ishrat Rahman - 160041069

Inara Zahin Hassan - 160041055

Supervisor

Md. Kamrul Hasan

Affiliation

Co-Supervisor

Hasan Mahmud

Affiliation

A thesis submitted in partial fulfilment of the requirements

for the degree of B. Sc. Engineering in Computer Science and Engineering

Academic Year: 2019-2020

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT)

A Subsidiary Organ of the Organization of Islamic Cooperation (OIC)

Dhaka, Bangladesh

March 13, 2021

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by Arowa Yasmeen, Fariha Ishrat Rahman and Inara Zahin Hassan under the supervision of Md. Kamrul Hasan, Professor of the Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Dhaka, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

Arowa Yasmeen
Student ID - 160041073

Fariha Ishrat Rahman
Student ID - 160041069

Inara Zahin Hassan
Student ID - 160041055

Object Tracking using End-to-End Detection and Deep Association Metric

Approved By:

Supervisor:

Md. Kamrul Hasan 24/04/21

Md. Kamrul Hasan, PhD

Professor

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT), OIC

Acknowledgement

We would like to express our grateful appreciation for **Professor Md. Kamrul Hasan**, Department of Computer Science & Engineering, IUT for being our adviser and mentor. His motivation, suggestions and insights for this research have been invaluable. Without his support and proper guidance this research would never have been possible. His valuable opinion, time and input provided throughout the thesis work, from first phase of thesis topics introduction, subject selection, proposing algorithm, modification till the project implementation and finalization which helped us to do our thesis work in proper way. We are really grateful to him.

We are also grateful to **Hasan Mahmud**, Assistant Professor, Department of Computer Science & Engineering, IUT for his valuable inspection and suggestions on our proposal of Object Tracking using End-to-End Detection and Deep Association Metric.

Object Tracking using End-to-End Detection and Deep Association Metric

Abstract

Object Tracking has multiple major applications such as video surveillance for security, traffic control, contact tracing, human computer interaction, gesture recognition, augmented reality, video editing, robotics etc. Often, to perform real-time tracking, video surveillance applications forgo detection accuracy in favour of detection speed. This paper proposes a combination of object detection and object tracking algorithms that gives an improvement on both detection accuracy and speed compared to existing video surveillance solutions. It also includes a method to trace the movement of a target from video surveillance footage and visualise the target's path on a 2D map.

Keywords: *Object Detection, Object Tracking, End-to-End Detection, Deep Association Metric, Mapping movement*

Contents

1 Introduction

1.1 Overview

Object Tracking is often used as an umbrella term for both Object Detection and Object Tracking itself. We detect before we track. Using *Object detection* algorithms, the different instances of a particular class, e.g. people, cats, dogs, cars, building etc., can be detected. *Object tracking* hereafter, estimates where the detected object would be present in the next scene by using previous information collected from earlier frames.

A visual surveillance system extracts information from a large scale dataset. The large scale dataset is usually taken from multiple cameras spanning the area under surveillance. There are various technologies such as CCD cameras, night vision devices like goggles and thermal imaging cameras. In this paper, we use the COCO dataset which is the industry standard. COCO is a large-scale object detection, segmentation, and captioning dataset which has 1.5 million object instances, 80 object categories/classes and 250,000 people with keypoints.

Given a video surveillance footage of an area, the aim is to effectively track a specified target and to highlight the target's movement and create a path trace on a 2D map. To achieve this goal we need to build a surveillance system that offers 3 things - object detection, object tracking and mapping.

The system visualisation would look as follows:

(a) Video (b) Map

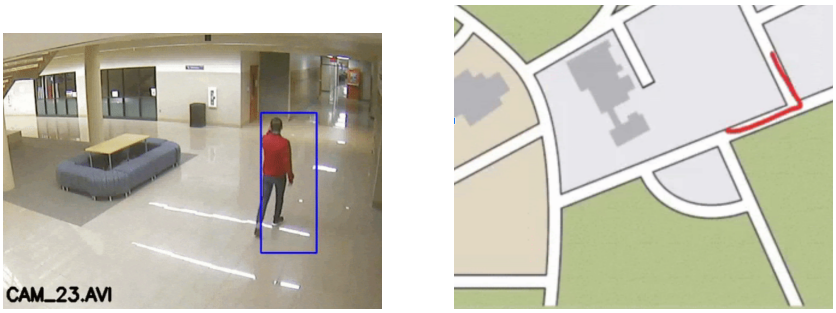
Figure 1: Position 1

(a) Video (b) Map

Figure 2: Position 2

(a) Video (b) Map

Figure 3: Position 3



(a) Video (b) Map

Figure 4: Position 4

1.2 Problem Statement

Using Object Detection for tracking is possible but it is extremely computationally expensive. This is because doing so means having to perform a detection per frame. Imagine having a 20 second 60fps (frame per second) video, if it takes roughly 1 second to perform a detection then it would take 20 minutes to compute a track for those 20s. Thus, it is quite obvious this is not a feasible choice for real-time tracking. Object tracking, however, is not class-dependent. In fact, it can track any object that has been marked via the detection algorithm. This is done by only looking for the detected object's distinctive features in the frames that come after. In existing video surveillance systems, the speed-accuracy trade-off usually leans towards speed to reduce the computational load and also because speed is an important metric to consider when performing tracking in real-time. Therefore, to develop an efficient object tracking system, there needs to be a combination of detection and tracking algorithms that will optimise the trade-off between speed and accuracy and also include a comprehensive way to visualise the movement of a target.

1.3 Motivation and scope of research

The current standard for object tracking systems prioritizes speed over accuracy. The systems allow some mis-identifications to maximise the overall inference speed. While speed is certainly important in real-time applications, this trade-off also

leads to serious consequences when applied to real-life scenarios. There have been multiple cases of wrongful arrests because of surveillance systems incorrectly identifying individuals. This problem sets the premise for our motivation behind our research. This paper discusses how to build an object tracking system that prioritizes accuracy while also displaying reasonable speed. The scope of this research not only limits itself to building a robust surveillance systems, but may also be applied to other avenues such as contact tracing.

1.4 Research Challenges

The typical challenges of background subtraction in the context of video surveillance have been listed below.

- *Illumination Challenges*: The models need to be robust and adapt to gradual changes in the environment. Abrupt changes in lighting conditions can occur both in outdoor and indoor environments. In outdoor cases such a scenario occurs when there is a fast transition from the sky being cloudy to it becoming completely clear. The position of the sun and the way sunlight, in general, illuminates the object of interest compared to the appearance of the background is what gives rise to this challenge. In an indoor setting, a similar situation is caused by the light source of the room e.g. turning it on/off.

- *Dynamic Background*: Handling background dynamics is extremely difficult due to its irregularity. Object tracking usually uses velocity to predict where the object is going to be next, so in the case where the background has significant movement it becomes difficult to deduce which movement should be classified as background movement and as target movement.

- *Occlusion*: Occlusion (partial/full) effects to process of discerning which is the background and which is the object of interest. This is an unpredictable phenomenon because at any time the object can be obstructed from the view of the camera - a person can pass by, the object itself might move out of the

camera frame etc.

- *Clutter*: Having clutter in the background makes segmenting out the moving objects in the foreground an even more difficult task.

- *Camouflage*: Tracking considers the distinguishable features of the object to predict its next position. However, if the object itself looks very similar to the background then it becomes even more difficult to not only make a prediction but also to correctly detect the object in the first place. It is important for surveillance systems to be especially robust in such a case so that criminals cannot take advantage of such a flaw.

- *Presence of Shadows*: The steps which come after background subtraction, such as shadow separation and classification, are often adversely affected due to the shadows of foreground objects overlapping with each other.

- *Video Noise*: In real life scenarios, and especially in the case of surveillance, it is common to find the superimposition of noise and the main video signal. The phenomena might occur due to a number of reasons such as sensor noise or compression artifacts etc. This causes video quality degradation and adds to another challenge during background subtraction.

1.5 Thesis Outline

In Chapter 1, a brief outline of the topic in hand was discussed. Chapter 2 deals with a more in-depth study of the research area and is primarily focused on two parts - Object Detection Algorithms and Object Tracking Algorithms. It also includes a study on the state of the art algorithms used in video surveillance and a detailed study on the algorithms used in the proposed solution. Chapter 3 delves into the overall system of the proposed solution and provides a more detailed outlook. In Chapter 4, the current progress stage of the proposal is discussed. Chapter 5 wraps up the proposal and includes all the references used.

2 Background study

2.1 Object Detection

Object detection is the task of identifying and locating objects of a certain class in digital images. On a high-level overview, all object detection algorithms can be classified into two types - One-stage methods and Two-stage methods.

One-Stage Methods: *(prioritises speed over accuracy)*

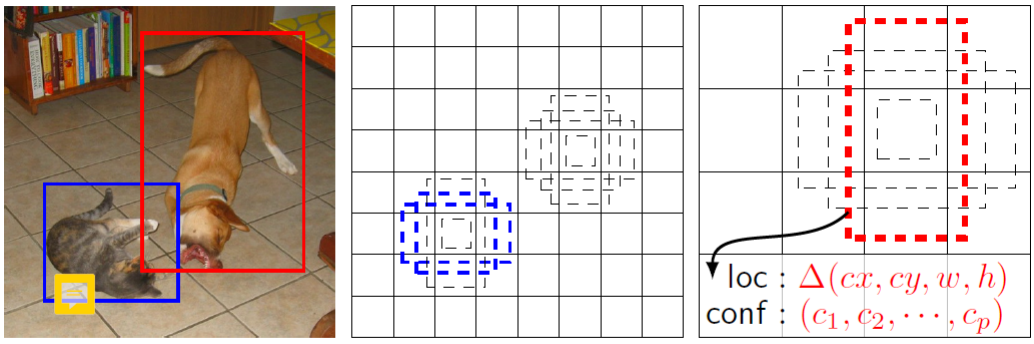
- These methods use a single deep neural network for detection.
- Ex: SSD, YOLO and RetinaNet.

Two-stage Methods: *(prioritises accuracy over speed)*

- In the first stage, ROIs (Regions of Interest) are proposed
- In the second stage, the previously proposed ROIs are processed by a classifier to look for targets.
- Ex: Mask R-CNN, Faster R-CNN and Cascade R-CNN.

2.1.1 Single Shot MultiBox Detector (SSD)

SSD[4], proposes a set of default boxes over different aspect ratios and scales for every feature map location. These default boxes are then searched for the presence of any target objects and assigned scores during prediction time. SSD, then adjusts these default boxes to accommodate the object shape. Multiple features maps of various resolutions are combined by the network to deal with objects of various sizes.



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Figure 5: SSD Framework

The SSD framework follows the steps below:

- During training, SSD needs to be provided with the input images annotated with the ground truth boxes.
- As noticeable in (b) and (c), the default boxes are of different size and aspect ratios and are evaluated using convolutions in several feature maps of different scale (can be 8x8 or 4x4 etc).
- The shape offsets and scores for all target objects are predicted for each default box as ((c_1, c_2,...,c_p)).
- During training, default boxes are matched with the ground truth boxes and if it's a match then they are classified as positives and the remaining are negative.
- The loss function in this model is a weighted sum between localization loss (e.g. Smooth L1) and confidence loss (e.g. Softmax).

The pros of SSD:

1. Simple.
2. Easy to train.
3. Straightforward to integrate into systems.
4. Provides a unified framework for both training and testing.
5. Gives a better accuracy for input images which are of smaller size

The cons of SSD:

1. Faster R-CNN would be a better choice than SSD in the case of objects of a small-scale.
2. Accuracy and the number of default boundary boxes proposed are directly proportional however this greatly reduces speed due to increased computations.
3. SSD has lower localization error compared with R-CNN but more classification error when target objects are of similar classes.

2.1.2 You Only Look Once (YOLO) [5]

The YOLO framework follows the steps below:

1. Bounding Box Prediction
2. Class Prediction
3. Predictions Across Scales
4. Feature Extractor: Darknet-53

Bounding Box Prediction

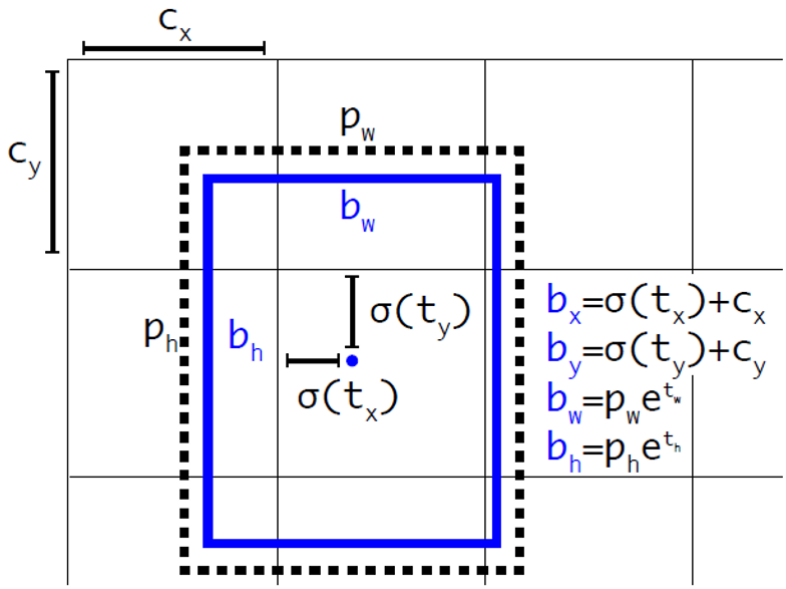


Figure 6: YOLO

- The principle is exactly same as YOLOv2.
- Predictions of t_x, t_y, t_w, t_h are made.
- The loss function used during training is sum of squared error loss.
- Objectness score is a logistic regression prediction. 1 if the intersection between the bounding box prior and ground truth object is far greater than any other bounding box prior. Only one bounding box prior is assigned for each ground truth object.

Class Prediction

- Softmax is not the choice for classification.
- Independent logistic classifiers are used and binary cross-entropy loss is the loss function that is used. This is mainly to account for cases of overlapping labels for multi-label classffication. Example: YOLOv3 moved to a more complex domain such as Open Images Dataset.

Predictions Across Scales

- 3 different scales are used to extract features
- Multiple convolutional layers are added to the base feature extractor Darknet-53 (discussed more elaborately in the next section).
- The last layer then predicts the bounding box, objectness and class predictions.

3 boxes at each scale are used [4] for running on the COCO dataset. As a result, the output tensor is N×N×[3×(4+1+80)], where the number correspond to 4 bounding box offsets, 1 objectness prediction, and 80 class predictions. The encoder-decoder architecture consists of a feature map extracted from 2 layers prior which is then upsampled by times 2. A feature map is also extracted from earlier in the network and combined with the upsampled features via concatenation. Thus, more meaningful semantic information is obtained from the upsampled features and more detailed information from the previous feature map. Additional convolutional layers then process this concatenated feature map, and eventually predict a similar tensor, but now it will be twice the size. A better bounding box prior is then computed using K-means clustering. Lastly, on COCO dataset, (10×13), (16×30), (33×23), (30×61), (62×45), (59×119), (116×90), (156×198), and (373×326) are used.

Feature Extractor: Darknet-53

- In YOLOv3, a much deeper network (compared to Darknet-19 used in Yolov2) Darknet-53 is used. It has 53 convolutional layers.
- Batch Normalization is used in both YOLOv2 and YOLOv3.
- The shortcut connections are shown in Figure 7.
- The above comparison is done on 1000-class ImageNet using a Single Crop 256x256 image. The testing was carried out using a Titan X GPU.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 7: DarkNet 53

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

1000-Class ImageNet Comparison (Bn Ops: Billions of Operations, BFLOP/s: Billion Floating Point Operation Per Second, FPS: Frame Per Second)

Figure 8

- Darknet-53 is 1.5 times faster than ResNet-101 and 2 times faster than ResNet-152.
- Performance-wise it is better than ResNet-101 and achieves a similar level performance compared to ResNet-152.

2.1.3 RetinaNet [10]

RetinaNet is a one stage method. It generates the bounding box and the class probabilities using one deep neural network. It uses the novel focal loss to back-propagate and learn. The focal loss gives higher weight to hard examples and a

smaller weight to easy examples. It uses a CNN architecture to extract important features from the input. The CNN backbone uses a combination of ResNet 50 and FPN. The input is then passed to a feed forward network that uses a softmax classifier for class prediction and regression to predict the bounding boxes.

There are four major components of a RetinaNet model architecture.

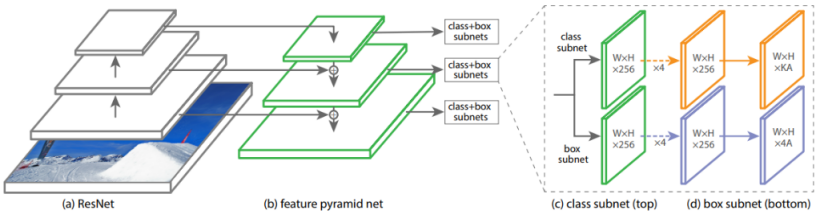


Figure 9: RetinaNet Architecture

1. *Bottom-up Pathway* - The CNN backbone is comprised of the ResNet architecture. It takes in the input and generates an activation map that extracts important features from the input. The CNN architecture is scale invariant.

2. *Top-down pathway and Lateral connections* - The top down layers and the bottom up layers are of variable sizes. Therefore it is required to merge the connection between them such as to ensure that they are of the same size. To do this, the top down approach upsamples the feature maps.

3. *Classification subnetwork* - The classification network is responsible for generating the class labels.It predicts the probability of an object being present at each spatial location for each anchor box and object class.

4. *Regression subnetwork* - The regression subnetwork is used to generate the bounding boxes. The network learns such as to increase the IOU overlap between the predicted bounding box and the ground truth.

One stage methods suffer from a class imbalance problem. To achieve this, the cross entropy loss is replaced with the novel focal loss. Easy examples are defined as predictions with high confidence i.e high probabilities whereas hard examples

are defined as predictios with low confidence i.e low probabilities. The focal loss emphasizes on these hard examples. So the model suffers from a higher loss when misidentifying hard examples and smaller loss when misidentifying the easier problems. This helps relieve the class imbalance problem and the model is not biased toward classes with greater number of examples.

2.1.4 Mask Recurrent Convolutional Neural Network (Mask R-CNN) [6]

Mask R-CNN is a two stage method. The first network is used to generate region proposals of where the objects may be located and the second network uses these region proposals from the first network to generate bounding boxes and class labels. The region proposal network of mask RCNN is the same as that of Faster RCNN. It differs in its second stage. in addition to generating the class and bounding box predictions, it also generates a binary mask for each region of interest. The CNN backbone is built on the ResNet architecture.The early layer of the network detects low-level features, and later layers detect higher-level features. Mask RCNN is a image segmentation algorithm that separates the foreground from the background. This image segmentation procedure leads to many inaccuracies. This is because it is dependent on the specific pixel values, and hence cannot accurately handle occlusions. To counter this problem, Mask RCNN uses the ROIAlign feature. It handles the occlusion problem by sampling the CNN feature map and then using bilinear interpolation to fill in the missing values. This is then passed onto a CNN architecture which takes the proposed regions and generates a binary mask for them.

2.1.5 Faster R-CNN + Feature Pyramid Network (FPN) [7]

Feature pyramids are a fundamental element in detection methods for detecting targets at various ranges. The pyramids comprise an integrated multi-level, pyramidal hierarchy of extensive convolutional layers.

In Feature Pyramid Network (FPN) a top-down architecture with lateral con-

nections is built for building high-level semantic feature maps at all scales. FPN showed noticeable development as a general feature extractor in numerous areas.

A deep ConvNet computes a feature hierarchy bu succeeding through many layers, and with sub sampling layers the feature hierarchy possesses an intrinsic multiscale, pyramidal configuration.

This feature hierarchy integrated within the network generates feature maps of varying spatial resolutions but adds considerable semantic gaps made by varying depths.

The aim is to easily leverage the pyramidal configuration of the feature hierarchy of a ConvNet while building a feature pyramid that holds powerful semantics at all levels.

2.1.6 Cascade R-CNN [8]

Cascade R-CNN is a multi-stage method of object detection. It has a series of detectors trained step by step with increasing Intersection over Union (IoU) thresholds, to be progressively more particular against close false positives. At each stage, it ensures that the output of a detector is a useful distribution for training the succeeding higher level detector. This method of training assures that all detectors have a positive set of cases of comparable size, decreasing the overfitting problem.

As the name suggests, Mask RCNN is a multi-stage continuation of the R-CNN, where detection steps are more profound into the cascade. The detection stages are sequentially more selective against close false positives.

The regression task is broken down into a sequence of shorter steps. In Cascade, R-CNN is a cascaded regression problem. It depends on a cascade of specialized regressors. Cascaded regression is a resampling procedure that changes the distribution of hypotheses to be processed by different stages. Because it is used at both training and inference, there is an insignificant difference between training and inference patterns. The multiple specialized regressors $\{f_T , f_{T-1}, \cdots , f_1\}$are optimized for the resampled distributions of the different stages.

The main areas of improvements of Cascade R-CNN are that it diminishes overfitting issues. Next, the detectors of the deeper stages are optimized for larger IoU thresholds.

2.1.7 Comparison of Detection Algorithms

The following table shows the comparison of different one stage and two stage methods. The MS COCO dataset was used to generate these results. We can quantitatively prove what we had mentioned before. Two stage methods perform significantly better than one stage methods when it comes to accuracy. However, due to their bulky architecture, two stage methods perform poorly when it comes to speed.

One-Stage Methods				Two-Stage Methods			
Algorithms	**AP**	**AP50**	**AP75**	**Algorithms**	**AP**	**AP50**	**AP75**
Single-Shot MultiBox Detector: SSD [4]	28.8	48.5	30.3	Mask R-CNN	39.8	62.3	43.4
YOLOv3 + DarkNet53 [5]	33.0	57.9	34.4	Faster R-CNN + FPN	36.2	59.1	39.0
RetinaNet	34.3	53.2	36.9	Cascade R-CNN	40.6	59.9	44.0

Figure 10: Accuracy Comparison of Detection Algorithms

One-Stage Methods		Two-Stage Methods	
Algorithms	**FPS**	**Algorithms**	**FPS**
YOLO	45	Faster R-CNN VGG-16	7
YOLOv2	91	Faster R-CNN ResNet	5

Figure 11: Speed Comparison of Detection Algorithms

2.1.8 End-to-End Object Detection with Transformers (DETR) [9]

The DETR architecture comprises of three basic blocks - a CNN backbone, a transformer encoder-decoder architecture and a feed forward network. It uses a bipartite matching loss function to back-propagate and update parameters. It

generates bounding box coordinates for the detected objects and also returns class probabilities.

CNN Backbone

The CNN backbone takes a RGB image as input, performs a series of convolution operations and generates a compact feature representation of the image i.e the width and height of the output is smaller than the input, however, the number of channels increase.

Transformer encoder

The specialty of the transformer architecture is its attention mechanism. It is able to refer or tend to previous inputs similar to RNN and LSTM. However, while the latter models have short reference windows, transformers have significantly longer windows. Also, to reduce computation speed, unlike RNN and LSTM which process input sequentially, transformers process inputs parallelly. They achieve this by adding positional encoding to the input. The encoder architecture consists of multi-headed attention layers, linear and normalisation layers. They generate attention weights which are then added to the input. This output is then passed onto the decoder architecture.

Transformer decoder

The decoder takes the output from the encoder module as well output embeddings known as objects queries as input. The objects queries are learned parameters. They are randomly initialised and are updated as the model learns. The decoder architecture is auto regressive and takes previous output as input. Similar to the encoder architecture, it consists of multi-headed attention layer, linear and normalisation layers. The outputs from the decoder architecture is then passed on to a feed forward network for prediction.

Feed forward network

The output from the decoder is passed to a feed forward network which generates two outputs- bounding box coordinates and class probabilities. The output is passed to a FFN with ReLU activation function and a final linear prediction layer. The object queries define the number of outputs. If N object queries are fed into the deocoder, it predicts N bounding boxes. The FFN generates the width and height of the box along with its normalised center coordinates. A softmax classifier is used to generate the class probabilities. In our model, we used the MS COCO dataset which consists of 80 classes, hence the softmax classifier predicts the probability for 80 different classes. If N objects are not present in the image, the other predictions are defined as null.

Loss Function

A bipartite matching function is used as the loss function. It creates a one to one association between the predicted outcome and ground truth such as to reduce the total cost.

$$\hat{\sigma} = \underset{\sigma \in \mathfrak{S}_N}{\arg\min} \sum_{i}^{N} \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}),$$

Figure 12: Bipartite Matching Loss

$L_{match}()$ is a pair-wise matching cost between ground truth y_i and a prediction with index σ (i). The optimal assignment described above is achieved through the Hungarian assignment algorithm. It efficiently performs a series of permutations to generate the best match with minimal cost.

$$\mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}}(i)) \right]$$

Figure 13: Hungarian Assignment

The bounding box loss i.e the box loss shown above considers the intersection over union of the predicted bounding box and the actual bounding box. It is defined as,

$$\lambda_{\text{iou}}\mathcal{L}_{\text{iou}}(b_i, \hat{b}_{\sigma(i)}) + \lambda_{\text{L1}}||b_i - \hat{b}_{\sigma(i)}||_1$$

Figure 14: Box Loss

2.1.9 Comparison with DETR

We compare DETR with the one-stage method YOLO and the two-stage method Faster RCNN. YOLO is the existing SOtA for object tracking eventhough its accuracy is lower than Faster RCNN. This is because Faster RCNN has a very small FPS which makes it unsuitable for real time tracking. Hence, DETR seems to optimize this tradeoff between speed and accuracy while being as accurate as Faster RCNN but consuming half the computational power.

Model	GFLOPS/FPS	AP	AP50	AP75
Faster RCNN	180/7	42.0	62.1	45.5
YOLO	63/91	33.0	57.9	34.4
DETR	86/28	42.0	63.1	45.9

Figure 15: Comparison with DETR

2.2 Object Tracking [1]

The aim of object tracking is to trace a motion of the objects in successive video frames. Traditional Object Tracking Algorithms include:

- Mean Shift Method
- Optical Flow Method
- Multiple Hypothesis Tracking

- Joint Probabilistic Data Association Filter

The State of the Art Tracking Algorithms are:

- Simple Online Real-Time Tracking (SORT)
- Simple Online Real-Time Tracking with Deep Association Metric (Deep SORT)

2.2.1 Mean Shift Method

The mean-shift algorithm is an efficient approach to tracking objects whose appearance is defined by histograms. Mean Shift Method is a tool for finding modes in a set of data samples, manifesting an underlying probability density function (PDF) in R^n.

Strengths:

- This tool is application independent
- Mean Shift Method is fit for data analysis
- Does not consider any former shape on information groups
- Has the ability to manage arbitrary feature spaces
- Just ONE parameter to select
- h (window size) has a physical meaning, unlike K-Means 21

Weaknesses:

- The size of the window, a.k.a bandwidth selection, is not insignificant The size of the window is unsuitable it can prompt methods to be merged, or produce further "shallow" methods. Use adaptive window size

2.2.2 Optical Flow Method

Optical flow is a method applied for object tracking which observes the movement of features due to the relative movement across frames between the clip and camera. For instance, if we are detecting the motion of a vehicle travelling left, we can determine the motion vectors in frame 2 relative to frame 1. Now if our the vehicle is travelling at a specific speed, we can use these motion vectors to track and even estimate the trajectory of the object in the following frame. A regularly applied optical flow tool is called Lucas Kanade.

2.2.3 Multiple Hypothesis Tracking

MHT produces a tree of possible track hypotheses concerning each object, thus producing a methodical solution to the information association query. We can consolidate MHT with various filter models, for example, the ones implemented by the IMM (interacting multiple model) methods. Weaknesses:

- No Occlusion Handling
- Uses up a lot of computational power for group tracking

2.2.4 Joint Probabilistic Data Association Filter

Human tracking in video is vital for areas like interactive multimedia, activity identification, and monitoring. Two of the principal difficulties in tracking are forming sufficient features for tracking and fixing information uncertainties to outline trajectories. Joint probabilistic data association filter is a silhouette based tracker with decreased complexity for determination of hurdles related to measurement-to-track.

JPDAF contributed in the following three areas. Firstly, the application of geometric limitations in decreasing unreliability in computations to any arbitrary degree of precision. Also, JPDAF is performed like table lookup allowing parallel tracking of a number of objects. And lastly, to track people a crowd, several feature clustering is employed to the silhouette region to distinguish uniform area for

tracking. Output from four video sequences that were used for testing proves real-time execution and great accuracy over mean shift method of tracking provided the identical primary human positions as input.

2.2.5 Deep SORT [2]

DeepSORT is an object tracking algorithm and the characteristic that makes it unique and superior to all tracking algorithms is that it tracks based on not only the distance travelled by the object, or the velocity of the object but also based on the appearance of the person. DeepSORT enables this feature to be added by calculating deep features for all bounding boxes and uses the correlation between deep features to factor into the tracking logic.

he first step of DeepSORT is the essential component called a Kalman Filter. The Kalman tracking scenario is set on eight dimensions *(u,v, a,h,u',v', a',h')*. *(u,v)* represents the center of the bounding box. a represents the aspect ratio and h is the height of the box. The additional variables namely *u',v',a',h'*, are the respective velocities of the variables.

The variables hold absolute position and velocity factors as we are considering the velocity of the model is linear. The Kalman filter aids in reducing noise and employs the previous state in outlining a suitable fit for bounding boxes.

For every detection, a "Track" is created. The track has all the required state knowledge. It also has a parameter to track and eliminate tracks that had their latest successful detection quite a long ago, as those objects are supposed to have left the scene already. To delete duplicate tracks, a threshold is set for the minimum number of detection for the first few frames.

After Kalman Filter is done tracking the current bounding boxes, the following task is to incorporate upcoming detections with the upcoming predictions. To solve the association of a track with incoming detection, we require two elements: a distance metric to quantify the association and an effective algorithm to associate the data.

The founders of Deep SORT chose the squared Mahalanobis distance which is an effective metric when working with distributions, to include the unreliability from the Kalman filter. Using Mahalanobis distance is more reliable than euclidean distance as we are adequately calculating the gap between 2 distributions of the Kalman filter. As the efficient algorithm, Deep SORT authors selected the standard Hungarian algorithm, as it is is a very powerful yet uncomplicated combinatorial optimization algorithm that deals with the assignment problem.

The appearance feature vector

So far, we had an object detector that provided us with the detections. Then the Kalman filter tracked detections and provided us with missing tracks. Lastly, the Hungarian algorithm does the association. Kalman Filter itself is quite powerful. However, it is incapable of tracking effectively in numerous real-world situations like occlusions, various perspectives, etc. To rectify this, the founders of Deep SORT included a new metric based on how the object looks. Hence, the designers first constructed a classifier over the dataset. Afterwards, the classifier was trained until it attained a fairly reliable accuracy. Lastly, the fn=inal classification layer was stripped. Considering a traditional classifier architecture, they were left with a dense layer that generated a single feature vector, expecting to be classified. That feature vector was the "appearance descriptor" of the subject being tracked.

Name	Patch Size/Stride	Output Size
Conv 1	$3 \times 3/1$	$32 \times 128 \times 64$
Conv 2	$3 \times 3/1$	$32 \times 128 \times 64$
Max Pool 3	$3 \times 3/2$	$32 \times 64 \times 32$
Residual 4	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 5	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 6	$3 \times 3/2$	$64 \times 32 \times 16$
Residual 7	$3 \times 3/1$	$64 \times 32 \times 16$
Residual 8	$3 \times 3/2$	$128 \times 16 \times 8$
Residual 9	$3 \times 3/1$	$128 \times 16 \times 8$
Dense 10		128
Batch and ℓ_2 normalization		128

Figure 16: Overview of the CNN architecture in Deep SORT

In the above diagram showing the layers of the classifier, we can see a layer labelled "Dense 10". The "Dense 10" layer will be the appearance descriptor for the clip

under consideration. Once training the classifier is complete, all the clips of the detected bounding boxes from the image are sent to this network to achieve the "128 X 1" dimensional feature vector.

The cost function is as follows:

$D = \text{Lambda} * D_k + (1 - \text{Lambda}) * D_a$

Where,

D_k= Mahalanobis distance
D_a = Cosine distance between the appearance descriptors
Lambda = weighting factor.

2.3 Existing Works

The current state of the art visual surveillance technique is a combination of YOLOv3 and DeepSORT.

In the YOLO+DeepSORT implementation, visual object tracking is performed on videos by training a YOLOv3 detector on the COCO data set. The dataset comprises 800 images that contain objects of 6 distinct classes. A YOLOv3 detector performs the object detection is done followed by a DeepSORT tracker performing tracking of the objects that fall under the 6 specific classes. The authors used the Python language to implement the system. The classes upon which they trained the system were: Human, Car, Bus, Truck, Bike and Motorbike.

- The code was executed in Google Colab where the models ran through 320 epochs.
- Using the LabelImg tool, the 800 images were annotated one by one
- The Pytorch library was also used on the data set during training.
- Frames were named using the YOLO format.
- Out of the 800 images, 200 images were utilised for validation.
- Upon annotation, a .txt extension was added to the images

After finishing labelling, the next step is to place the images and annotations/labels in a directory. The information can either be sent as parameters or hardcoded within the main file with the help of the PyTorch library.

After training, the final output is a weights file. The YOLOv3 object detector uses this weight file. The model accepts the input video and then produces a total number of frames. The frames are then processed by the object detector.

YOLOv3 generates 27 bounding boxes. The bounding boxes will have a class ID and confidence values assigned to them. The detected bounding box is later furthered to the DeepSORT tracker. DeepSORT assigns each bounding box a unique identification number while tracking.

3 Proposed Solution

3.1 Skeleton of Proposed System

The skeleton of the proposed system can be divided into two parts - Detection and Tracking. In the detection stage, we take video footage as input and generate bounding boxes which represent the detected objects in the video frames. We then pass these bounding boxes to the tracking module which associates a unique ID with each detection that enables us to track objects across frames. We perform detection using DETR and tracking is implemented using Deep SORT.

3.2 Flowchart

The diagram below describes the model flowchart. The input is a video feed, where the frames follow an aspect ratio between 0.5 to 2, and no dimension can be greater than 1600. The CNN backbone used is the Resnet 50 which consists of 50 convolution layers. The transformer encoder consists of 1 multi-headed attention layer, 2 linear layers, 3 dropout layers and 2 normalization layers. The decoder consists of 2 multi-headed attention layer, 2 linear layers, 4 dropout layers and 3 normalization layers.The output is passed to a FFN. The result is filtered to detect only humans, and bounding boxes are generated. These bounding boxes are then passed onto the Deep SORT architecture. This tracking module uses the Kalman filter and a CNN model as described previously to generate bounding boxes with unique IDs.

4 Result Analysis

In order to compare the results of our proposed solution with the existing solution, we ran both the models on the same dataset, and we observed some significant improvements. We classify the two category of metrics as - missed detections and identity switches.

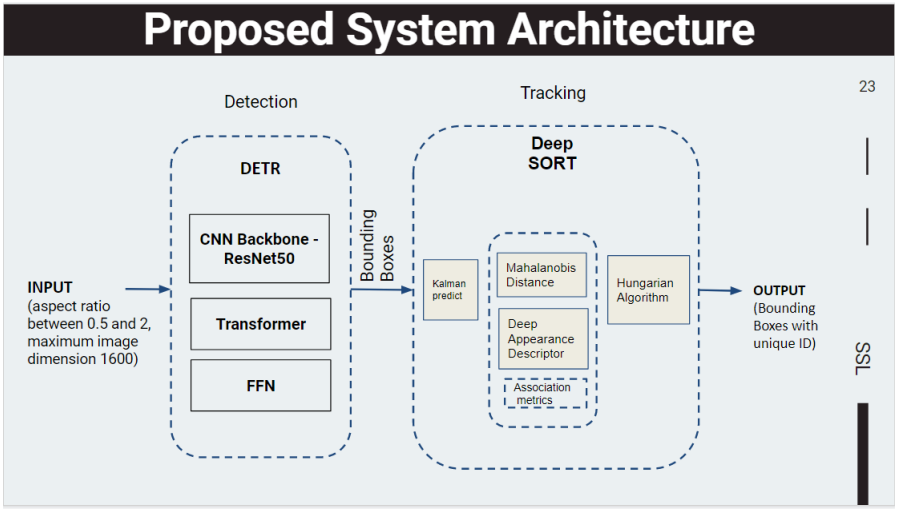


Figure 17: Flowchart of Our Proposed System

4.1 Missed Detections

The first major improvement we observed is the degree of missed detections. Our proposed solution is able to better handle occlusions and other shortcomings of the existing solution. While the YOLO + Deep SORT model missed quite a few objects, our proposed model is more robust. This can be observed in the figure below.

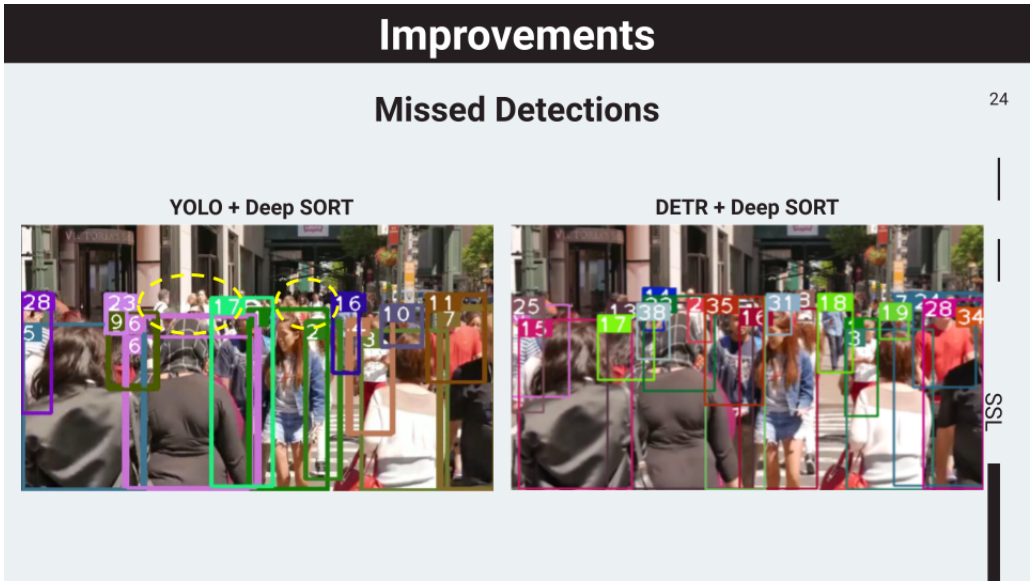


Figure 18: Missed Detections

4.2 Identity Switches

The next area improvement we observed is the number of identity switches. The YOLO + Deep SORT model experiences a lot of identity switches. It assigns the same ID to multiple people and also assigns the same person multiple IDs. This is prominent if there are occlusions or when the person exits the frame. The DETR + Deep SORT model we proposed reduces the number of identity switches. An example is given below.

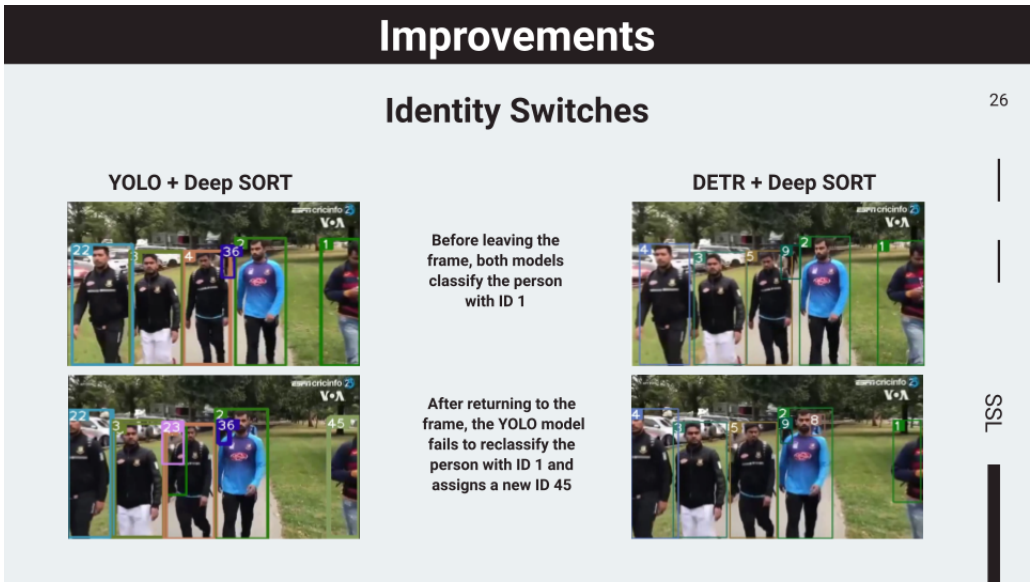
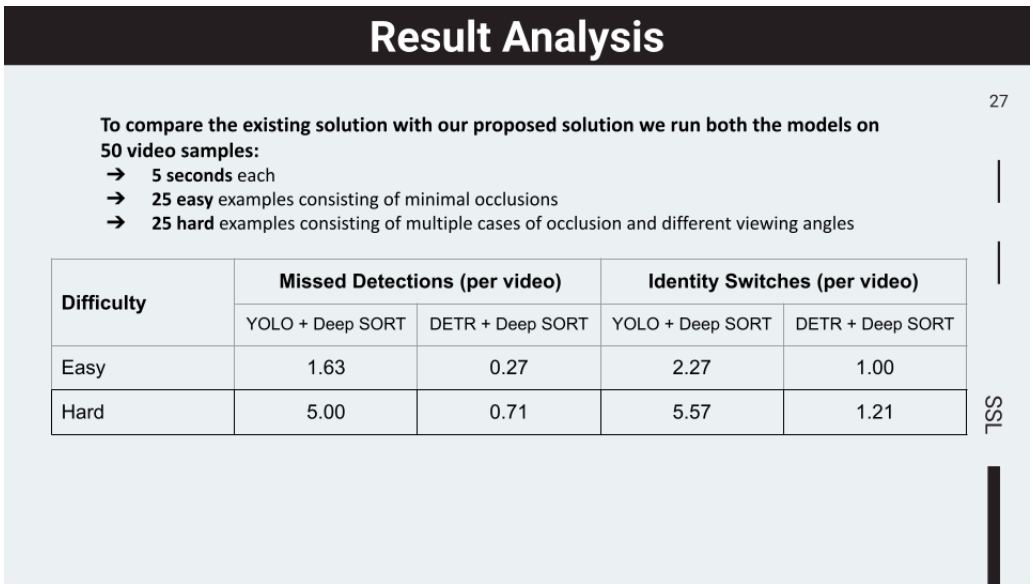


Figure 19: Identity Switches

4.3 Result Overview

We classified two metrics for result analysis - Missed Detections which is defined as the average number of missed detections per video and Identity Switches which is defined as the average number of identity switches per video. We compared the results on our curated dataset which consists of 50 videos, each 5 seconds long. It comprises of 25 easy and 25 hard examples. Easy examples are defined as those with minimal occlusions or obstructions on which both the models are expected to perform well. Hard examples have multiple cases of occlusions, obstructions and different viewing angles to make prediction harder. As we can see from the

figure below, the performance of our model is better than the existing solution. For easy examples the differences are not as drastic. The difference in performance becomes significantly prominent when tested on the hard examples. Our model consistently outperforms the existing solution in both metrics.



Difficulty	Missed Detections (per video)		Identity Switches (per video)	
	YOLO + Deep SORT	DETR + Deep SORT	YOLO + Deep SORT	DETR + Deep SORT
Easy	1.63	0.27	2.27	1.00
Hard	5.00	0.71	5.57	1.21

Figure 20: Result Analysis

5 Conclusion and Future Works

5.1 Conclusion

This 3 step system aims to provide a more visual, fast and accurate surveillance system that incorporates the use of newer technology and has the potential for mass use. Our contribution here is solving some of the key problems such as occlusion handling, the need for specified target detection and different lighting conditions by introducing DETR into the system model. It has rooms for growth and further works and can act as a base for such endeavours.

5.2 Future Works

Improving the DETR Architecture

The system is developed around the baseline DETR architecture. Further improvements can be made to this architecture to improve the performance of our system.

Predicting Paths We can further train our model to give predictions regarding the path taken by an individual when that data is not available. This will be especially useful when considering regions outside the surveillance area or in dealing with missing surveillance footage.

Preventing Privacy Breach The primary concern with surveillance systems is the implications of a privacy breach. Since we are dealing with sensitive information, it is essential that this data is protected. This may be achieved by creating a distributed architecture. We can apply federated learning to obtain the gradients and then upload that model to a blockchain. This ensures that the model only has access to the gradients and not the actual data.

References

[1] A. Mangawati, Mohana, M. Leesan and H. V. R. Aradhya, "Object Tracking Algorithms for Video Surveillance Applications," 2018 International Conference on Communication and Signal Processing (ICCSP), Chennai, 2018, pp. 0667-0671, doi: 10.1109/ICCSP.2018.8524260 (2018).

[2] N. Wojke, A. Bewley and D. Paulus, "Simple online and realtime tracking with a deep association metric," 2017 IEEE International Conference on Image Processing (ICIP), Beijing, 2017, pp. 3645-3649, doi: 10.1109/ICIP.2017.8296962 (2017).

[3] Bathija, A. and Sharma, G.. Visual Object Detection and Tracking using YOLO and SORT. Int. J. Eng. Res. Technol.(IJERT), 8, pp.705-708 (2019).

[4] Liu, W., et al. "SSD: Single shot multibox detector. arXiv 2016." arXiv preprint arXiv:1512.02325 (2020).

[5] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement. arXiv 2018." arXiv preprint arXiv:1804.02767 (2018): 1-6.

[6] Huang, Zhaojin, et al. "Mask scoring r-cnn." Proceedings of the IEEE conference on computer vision and pattern recognition. 2019.

[7] Lin, Tsung-Yi, et al. "Feature pyramid networks for object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[8] Cai, Zhaowei, and Nuno Vasconcelos. "Cascade r-cnn: Delving into high quality object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[9] Carion, Nicolas, et al. "End-to-End Object Detection with Transformers." arXiv preprint arXiv:2005.12872 (2020).

[10] Lin, T. Y., et al. "Focal loss for dense object detection. arXiv 2017." arXiv preprint arXiv:1708.02002 (2002).