DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2017

Tracking of Humans in Video Stream Using LSTM Recurrent Neural Network

MASOUMEH POORMEHDI GHAEMMAGHAMI

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF COMPUTER SCIENCE AND COMMUNICATION

Tracking of Humans in Video Stream Using LSTM Recurrent Neural Network

MASOUMEH POORMEHDI GHAEMMAGHAMI

Master in Machine Learning
Date: August 10, 2017
Supervisor: Mårten Björkman
Examiner: Danica Kragic Jensfelt
School of Computer Science and Communication

Abstract

In this master thesis, the problem of tracking humans in video streams by using Deep Learning is examined. We use spatially supervised recurrent convolutional neural networks for visual human tracking. In this method, the recurrent convolutional network uses both the history of locations and the visual features from the deep neural networks. This method is used for tracking, based on the detection results. We concatenate the location of detected bounding boxes with high-level visual features produced by convolutional networks and then predict the tracking bounding box for next frames. Because a video contain continuous frames, we decide to have a method which uses the information from history of frames to have a robust tracking in different visually challenging cases such as occlusion, motion blur, fast movement, etc. Long Short-Term Memory (LSTM) is a kind of recurrent convolutional neural network and useful for our purpose. Instead of using binary classification which is commonly used in deep learning based tracking methods, we use a regression for direct prediction of the tracking locations. Our purpose is to test our method on real videos which is recorded by head-mounted camera. So our test videos are very challenging and contain different cases of fast movements, motion blur, occlusions, etc. Considering the limitation of the training data-set which is spatially imbalanced, we have a problem for tracking the humans who are in the corners of the image but in other challenging cases, the proposed tracking method worked well.

Sammanfattning

I detta examensarbete undersöks problemet att spåra människor i videoströmmar genom att använda deep learning. Spårningen utförs genom att använda ett recurrent convolutional neural network. Input till nätverket består av visuella features extraherade med hjälp av ett convolutional neural network, samt av detektionsresultat från tidigare frames. Vi väljer att använda oss av historiska detektioner för att skapa en metod som är robust mot olika utmanande situationer, som t.ex. snabba rörelser, rörelseoskärpa och ocklusion. Long Short-Term Memory (LSTM) är ett recurrent convolutional neural network som är användbart för detta ändamål. Istället för att använda binära klassificering, vilket är vanligt i många deep learning-baserade tracking-metoder, så använder vi oss av regression för att direkt förutse positionen av de spårade subjekten. Vårt syfte är att testa vår metod på videor som spelats in med hjälp av en huvudmonterad kamera. På grund av begränsningar i våra träningsdataset som är spatiellt oblanserade har vi problem att spåra människor som befinner sig i utkanten av bildområdet, men i andra utmanande fall lyckades spårningen bra.

Contents

Chapter 1

Introduction

1.1 Background

In the past two decades, the problem of object detection, localization and tracking received significant attention in different research areas. This coincides with the rising demand for information about objects location and identity, which stems from applications in various fields, such as manufacturing, military, business management, surveillance and security, transport and logistics, medical care, traffic management, childcare, performance analysis in sports and sports medicine. Also human detection and tracking can be widely used in many applications, including people counting and security surveillance in public scenes [1], [2], [3]. Different methods have been used for this purpose. In some research such as [4], combination of Kalman filter prediction and mean shift tracking is used. Tree-structured probabilistic model for human tracking is used in [5]. Recently, Neural Networks such as radial basis function (RBF) neural network [6] and CNN [1], [2], [4], have become more popular for image processing purposes. CNNs have recently been applied to various computer vision tasks such as image classification [7], semantic segmentation [8], object detection [9], and many others. Such great success of CNNs lead it to be used mostly with distinguished performance in visual applications. Using CNN for tracking has a limitation which is related to training data. Tracking requires more data in order to set a sufficient variety and it is difficult to collect a large enough amount of training data for video processing applications and training algorithms. Several recent tracking algorithms [10] have mentioned the data leakage issue by transferring pretrained CNNs on a large-scale dataset such as ImageNet [11]. Online training videos can be used to train entirely from scratch online during test time and teach the tracker to handle complex challenges in the condition such as rotations, changes in viewpoint, lighting changes with no offline training being performed, but these tracking methods have a problem of too slow speed. Also, such trackers have lower performance compared with offline training methods because they cannot take advantage of the large number of videos for improving their performance [12].

1.2 Research Question

In this Master's Thesis project, Deep Learning for tracking human in a video stream will be implemented. The question that will be examined is how to adapt a Deep Learning based method for object detection to the application of human tracking and improve it in terms of tracking accuracy and computational cost and speed of the tracking. The assumption is

that humans should be possible to be re-tracked, if tracking is temporarily lost due to the human leaving the field of view. We implement and test different structures and architectures for improving overall performance of the results. The system should be able to track people at several distances and poses. Tracking algorithms suffer from sequence-specific challenges including occlusion, deformation, lighting condition change, motion blur, etc. So finding the best algorithm that could be invariant to these conditions is our goal. The method that we propose is trying to do a robust tracking.

1.3 Related work

Visual tracking, as a fundamental problem in computer vision, has found wide applications. Though many methods have been proposed in literature, it still remains a difficult task in unconstrained environments due to several factors such as viewpoint change, illumination variations, pose and scale variations, motion blur and so on. Most trackers can be divided into two sub-models. One is a motion model and the other is an appearance model. The motion model aims at estimating the posteriori probability of the target states. For motion models, particle filtering is the dominant approach which can be formulated as a sequential Monte Carlo importance sampling method for estimating the latent state variables of a dynamical system based on a sequence of observations [13]. Appearance model plays a central role in modern tracking systems. Based on appearance model, most trackers can be divided into generative tracker or discriminative tracker or hybrid of them. Discriminative models formulate tracking as an online classification problem which distinguish the target from the background. Discriminative trackers utilize the information from both the object and the background. Ensemble tracking is proposed in [14] where each base tracker is a least-square classifier which works directly on the image pixels. In [15], online boosting is proposed to online update discriminative features. Semi-supervised extensions of online boosting is addressed in [16] to prevent drift. To alleviate the "butterfly effect" [1] caused by inaccurate update of the classifier, multiple instance learning is employed in [17] which puts all ambiguous positive and negative samples into bags to learn a discriminative model for tracking. The concept of "superpixel" is proposed in [18] which facilitates the tracker to distinguish between the target and background. The methods which were read during the literature review have been classified into three classes; Fast Tracking, Robust Tracking and Robust & Fast Tracking. Some of these tracking methods which use deep learning, have been summarized in each category.

1.3.1 Fast Tracking

Tracking Human by using Kalman Filter and Mean Shift Methods

This method [4] is used for pedestrian detection and tracking by using a single night-vision video camera installed on the vehicle. They used a support vector machine (SVM) for tracking phase and combination of Kalman filter prediction and mean shift tracking for tracking phase. They performed their experiments in the special case with a fixed camera and stationary background. So it seems that this method will probably not generalize well to the motion blur situation with a non-stationary background and moving camera.

[1] The butterfly effect means that small causes can have large effects.

"Offline Human Tracking Using Spatial and Temporal Features of Image "

This method [1] estimates the location and the scale of an object given its previous location, scale, as well as current and previous image frames. In the proposed method, a creative shift-variant CNN architecture is designed to prevent drift problem when the distracting objects are similar to the target in cluttered environment. The CNNs learn both spatial and temporal features jointly from image pairs of two adjacent frames during offline training. The main idea of this method is based on this fact that the location and the appearance of an object in the previous frame should assist us to detect the object in the current frame, by using these additional pieces of information for detection. They effectively turn an object detector into a tracker that is capable of estimating the location and the scale of the target given its previous location, size, as well as current and previous image frames. The target location in the second image patch on the basis of spatial and temporal structures of these two patches was found. The spatial structures indicate the appearances, while the temporal structures capture the motion information. The structure of the proposed method is shown in Figure 1.1.

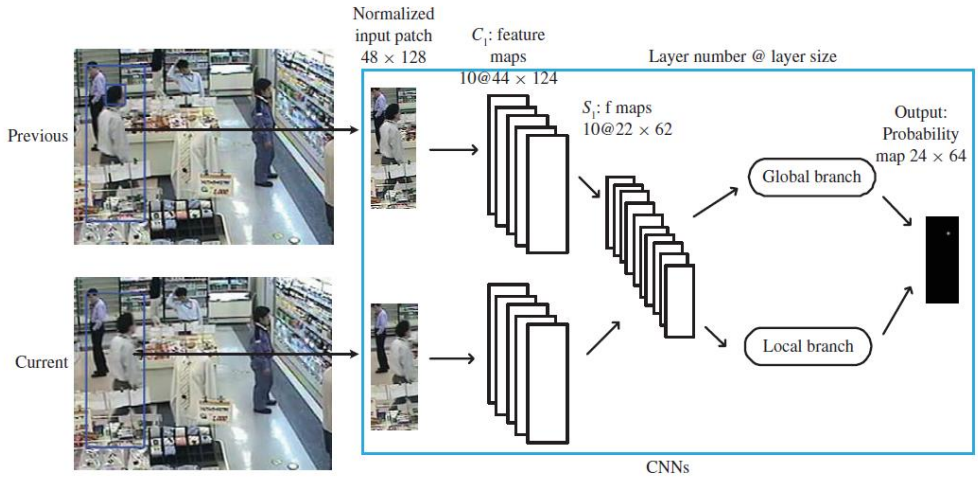

Figure 1.1: Architecture of tracking method which uses spatial and temporal features of images. (From Jialue Fan, Wei X, 2010)

"Visual tracking with Simple convolutional neural network"

In this Method [19], the visual tracking problem in a discriminant manner is addressed where a simple convolutional neural network (CNN) is employed to extract discriminant features and simultaneously classify the object from the background. This simple CNN model is demonstrated in Figure 1.2. This simple CNN consists of 5 layers in total. The input are the image patches cropped from the positive class (which comes from the object) as well as the negative class (which stands for the background). The second layer is the convolutional layer. Other layers are pooling layer, normalization layer, fully connected and softmax layers. This method is fast enough but not robust enough in different challenging environment such as motion blur [19].

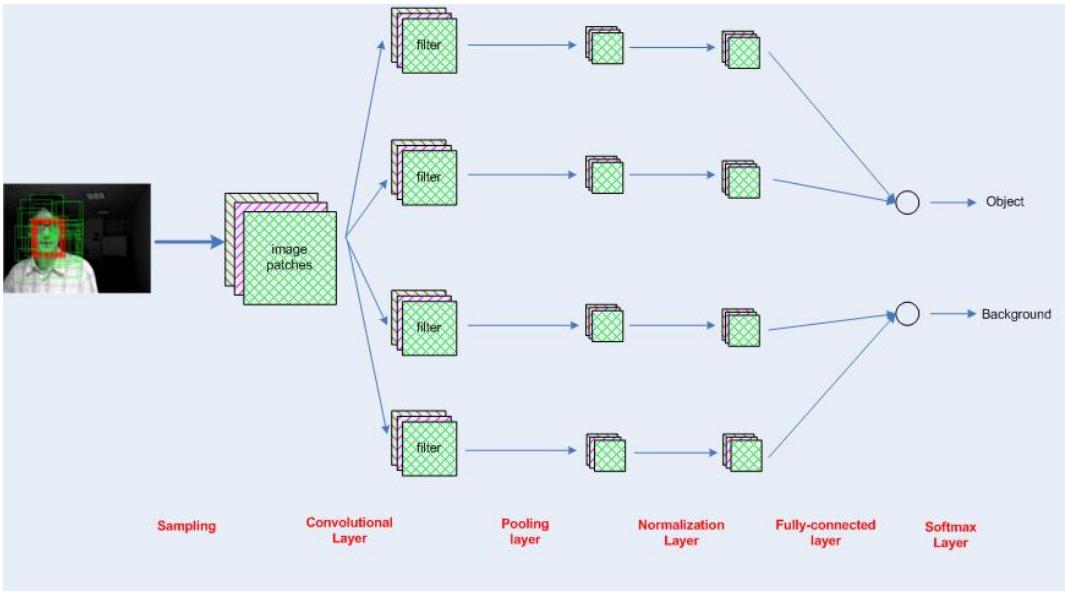

Figure 1.2: Basic structure of CNN for visual tracking. (From Le Zhang and Ponnuthurai Nagaratnam Suganthan, 2015)

"Learning to Track at 100 FPS with Deep Regression Networks"[20]

In this method a simple feed-forward network with offline training is proposed for tracking. The tracker learns a generic relationship between object motion and appearance and can be used to track novel objects that do not appear in the training set. This method is named (GO-TURN) Generic Object Tracking Using Regression Networks. A neural network for tracking in an entirely offline manner was trained. At test time, when tracking novel objects, frozen network weights are used, and no online fine-tuning required; so this method is very fast and it could track generic objects at 100 fps. The proposed method structure is shown in Figure 1.3. In this model, target object and search region are used as input and each input into a sequence of convolutional layers. The output of these convolutional layers is a set of features that are high-level representation of the image and fed through a number of fully connected layers. Fully connected layers compare the features from the target object with features in the current frame to find where the target object has moved.

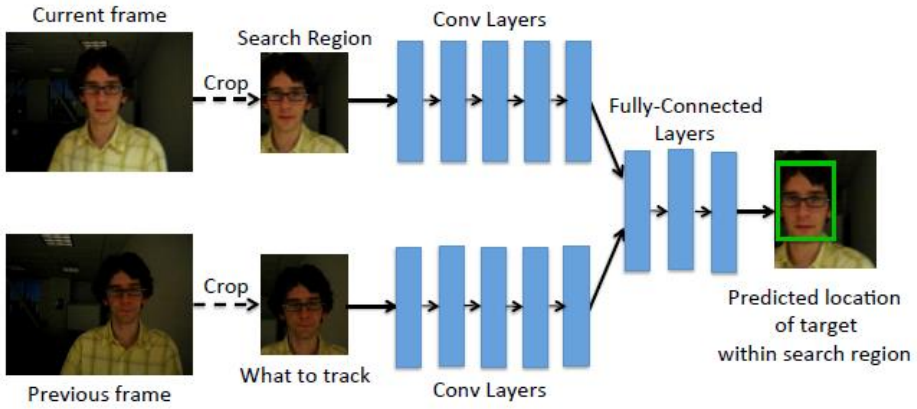

Figure 1.3: Architecture of GOTURN. (From David Held, Sebastian Thrun, and Silvio Savares, 2016)

1.3.2 Robust Tracking

"Learning Multi-Domain Convolutional Neural Networks for Visual Tracking"[21]

In this method, Multi-Domain Convolutional Neural Networks (MDNet) as a tracking algorithm based on the discriminatively trained Convolutional Neural Network (CNN) is used. A large set of videos with tracking ground-truths is used for pretraining of the CNN to obtain a generic target representation. The network has shared layers and multiple branches of domain-specific layers. Domains are related to individual training sequences and each branch is used for binary classification to find target in each domain. Each domain in the network is trained iteratively for generic target representations in the shared layers. During the tracking of a target in a new sequence, a new network would be built by combining the shared layers in the pretrained CNN with a new binary classification layer, which is updated online. Online tracking works by evaluating the candidate windows randomly sampled around the previous target state. This method produces very robust tracking and was the winner of The VOT2015 Challenge. The structure of this method is shown in Fig 1.4.

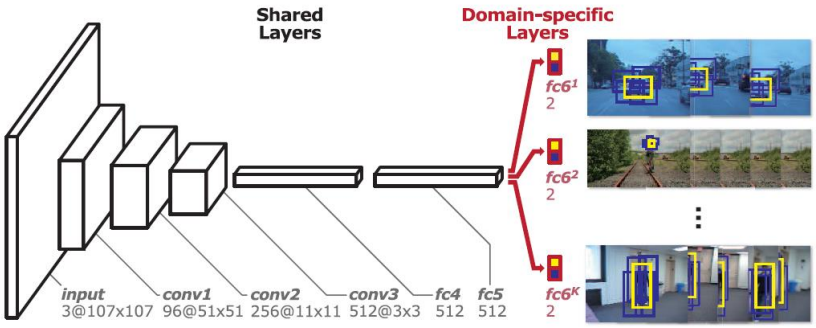

Figure 1.4: Architecture of MDNet. (From Hyeonseob Nam and Bohyung Han, 2016)

"STCT: Sequentially Training Convolutional Networks for Visual Tracking"[22]

In this method, a sequential training for convolutional neural networks and transfer pre-trained deep features with application of online tracking are used. CNN is assumed as an ensemble with each channel of the output feature map as a kind of individual base learner. For avoiding overtraining, each base learner is trained by using different loss criteria. For having an acceptable online ensemble, all the base learners are sequentially sampled into the ensemble via importance sampling. For further improving the robustness of each base learner, convolutional layers are trained with random binary masks, which are used as a regularization to make each base learner focus on different input features. The structure of the proposed method is shown in Figure 1.5.

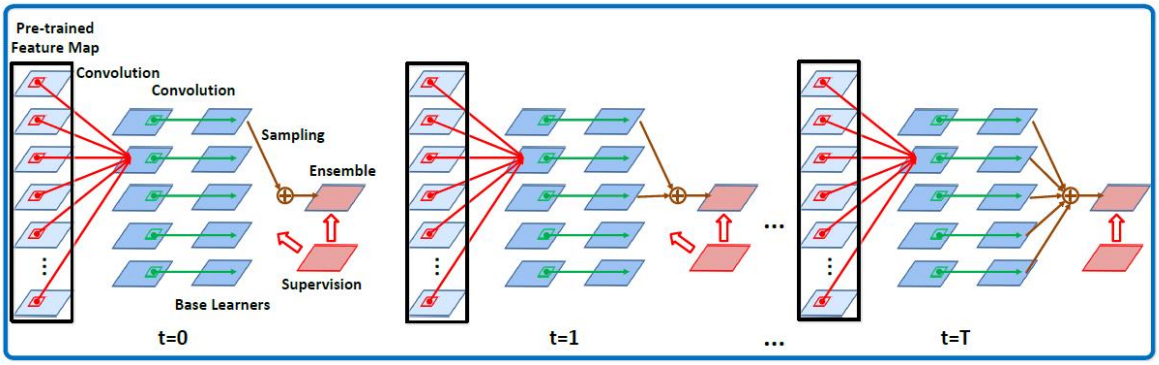

Figure 1.5: Proposed sequential training method for CNNs (STCT). (From Lijun Wang, Wanli Ouyang, Xiaogang Wang, 2016)

1.3.3 Robust & Fast Tracking

"Using Recurrent Convolutional Neural Networks for Visual Object Tracking"[23]

In this method, Recurrent convolutional network uses both the history of locations and the distinctive visual features which are learned by the deep neural networks. Both high-level visual features produced by convolutional networks and region information for tracking are used in this method. The structure of the tracking procedures is illustrated in Figure 1.6. YOLO(one successful detection method) is chosen to collect rich and robust visual features and preliminary location inferences; Then LSTM is used in the next stage as it is spatially deep and appropriate for sequence processing. This method is named ROLO (Recurrent YOLO). The proposed model is a deep neural network that uses raw video frames as input and its output is the coordinates of a bounding box of an object being tracked in each frame.

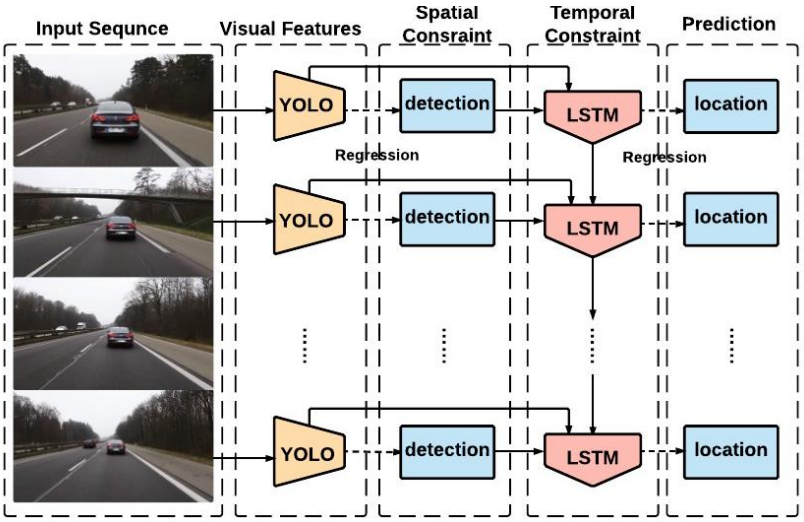

Figure 1.6: Structure of ROLO. (From Guanghan Ning, Zhi Zhang, Chen Huang, Zhihai He, 2016)

1.4 Evaluation and Datasets

1.4.1 Evaluation

Performance of the trackers could be evaluated by several evaluation criteria. One widely used evaluation metric for object tracking is the center location error, which calculates the average Euclidean distance between the center locations of the tracked targets and the labeled ground-truth positions of all the frames. Another commonly used evaluation metric is the overlap score [24]. The average overlap score (AOS) can be used as the performance measure. In addition, the overlap scores can be used for determining whether an algorithm successfully tracks a target object in one frame, by testing whether S is larger than a certain threshold t_0 (for example, 0.5).

1.4.2 Datasets

Some standard dataset such as Object Tracking Benchmark (OTB) [25] and VOT2014 [26] are very famous for tracking tasks and the tracking problem is shown up as a sub-task in these two computer vision challenges, OTB and VOT. These two challenges conveniently provide researchers with valuable datasets as well as annotated ground truth, and detailed evaluation methods, which makes the methods comparable with each other. The OTB evaluation is based on two metrics: center location error and bounding box overlap ratio. Some state-of the art trackers including MUSTer [27], CNN-SVM [10], MEEM [28], TGPR [29], DSST [30] and KCF [31], SCM [32] and Struck [12] participate in these challenges and their results are available for comparing our results with them. Also we used a database which is recorded by Tobii Pro Glasses 2 and annotated by ourselves and used as well as VOT & OTB. In this report, we name the data which is recorded by Tobii Pro Glasses 2, *Office & Supermarket* data, because mostly recorded in the supermarket and office. Figure 1.7 shows Tobii Pro Glasses 2 which has a camera on front of the Glasses. The Office & Supermarket data was recorded by this camera.

Figure 1.7: Tobii Pro Glasses 2

1.5 Limitations

As mentioned in the previous section, the thesis will address problems observed at using head mounted camera for recording video and tracking human in that video. So the proposed method needs to works well on the Office & Supermarket data. Office & Supermarket data contains more challenging cases than VOT & OTB. Videos were recorded with the Tobii Pro Glasses 2 so the camera was moving as well as the human which should be tracked. The up and down and fast movements of head of recorder affects the performance of the tracking process. Also there are lots of situation of motion blur, occlusion and change of illumination which we tried to handle.

1.6 Organization of the Report

This Master's Thesis project report is organized as follows: Chapter 1 provides an introduction, details the motivation for the project, the problems addressed, and a brief overview of previous methods. Chapter 2 focused on the theory and background behind the concepts explored in this thesis in the areas of Convolutional and Recurrent neural networks and LSTM. Chapter 3 explains in detail the method used to implement and perform the experiments. The details of the data set and the structure and parameters of the proposed network are addressed. Chapter 4 contains the results of the experiments, and a detailed discussion of the findings. Chapter 5 concludes the project report by summarizing the results and giving suggestions for future works.

Chapter 2

Theory and Background

2.1 Convolutional Neural Networks

Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that recently have a great application in visual analysis and machine learning. ConvNets have been successful in classification, segmentation, detection and tracking problems.

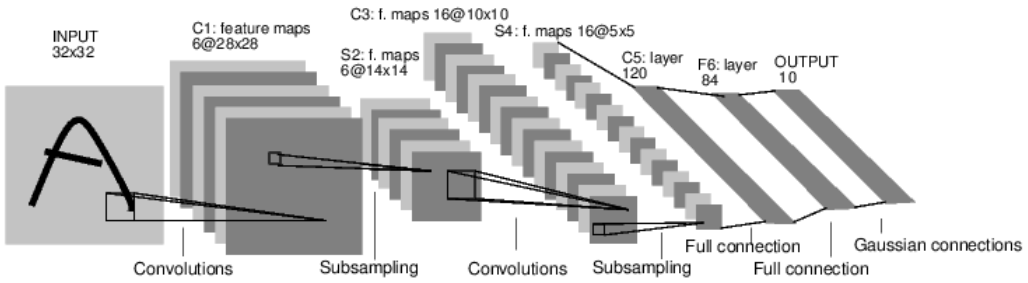

Figure 2.1: LeNet neural network structure. (From LeCun, Yann and Bottou 1998)

CNN has four main steps: convolution, subsampling, activation and fully connectedness. The most popular and first implementation of the CNN is the LeNet, which was introduced by Yann LeCun in 1998 [33]. Figure 2.1 illustrates the LeNet structure.

First step in CNN is convolution. The main idea of using convolution in first layers is extracting features from the input image. There are some filters that act as feature detectors from the original input image. In other words, convolution is a process where input signal is labeled by the network based on what it has learned in the past. If the network decided that the input signal looks like previous cat images that it has learned previously, the "cat" reference signal will be convolved with the input signal. The resulting output signal is then passed on to the next layer.

In the second step, subsampling, for reducing the sensitivity of the filters to noise and variations, the inputs from the convolution layer are smoothed. Subsampling also reduces the dimensionality of each feature map but save the most important information. This smoothing process is named subsampling or downsampling or pooling. Subsampling can be achieved by different methods of max, average, sum etc.

Third step is activation. The activation layer controls how the signal flows from one layer to the next layer. In different structures of CNNs, a wide variety of complex activation functions could be chosen to model signal propagation. One of the most famous function is

(ReLU), which is known for its faster training speed. The ReLu has the mathematical form of:

$$f(x) = max(0, x)$$

The forth step is fully connected. The last layers in the most Convolutional network are fully connected. It means that neurons of previous layers are connected to every neuron in next layers. The output from the convolutional and pooling layers contain high-level features of the input image. Features of fully connected layers are used by softmax layer and the input image is classified into different classes based on the training data. Also fully-connected layers help to learn non-linear combinations of mentioned features. Combinations of those features might be better for classification or other application of CNN.

2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are popular models that have shown great promise in many works which need the information of history such as language processing or video processing. The main idea behind RNNs is to use sequential information. In other neural networks, all inputs and outputs are independent of each other. But for many tasks it does not work well. For example, If you want to predict the next word in a sentence, it is better to know which words came before it. RNNs are called recurrent because they perform the same task for every element of a sequence, and the output depends on the previous computations. Also we can say that RNNs have a "memory" which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps . Here is what a typical RNN looks like:

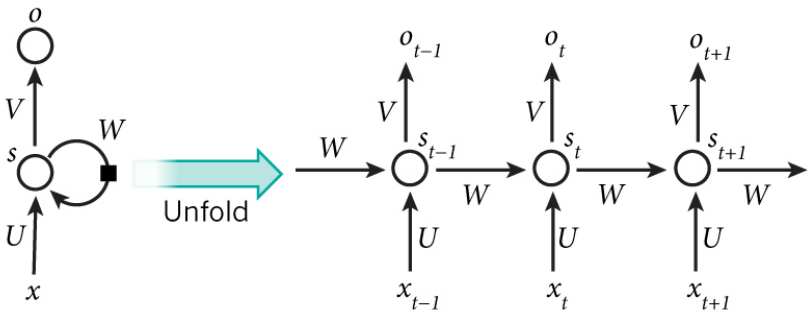

Figure 2.2: recurrent neural network and the unfolding in time of the computation involved in its forward computation

Figure 2.2 shows a RNN is unrolled or unfolded into a full network. With unrolling we simply mean that we write out the network for the full sequence. For example, if we want to use 5 sequences of a video, the network would be unrolled into a 5-layer neural network, one layer for each sequence. More details about the parameters in figure and the formulas of RNN are as follows:

x_t is the input at time step t. For example, x_1 could be a vector corresponding to the second sequence of a video. s_t is the hidden state at time step t. It's the "memory" of the network. s_t is calculated based on the previous hidden state and the input at the current

step:

$$s_t = f(Ux_t + Ws_{t-1})$$

The function f usually is a nonlinearity such as tanh or ReLU. s_{-1}, which is required to calculate the first hidden state, is usually initialized to zero. o_t is the output at step t. For example, if we wanted to predict the the position of human in the next time stamp in a video it would be a vector of probabilities.

$$o_t = \text{softmax}(Vs_t)$$

Training a RNN is similar to training other neural network. Backpropagation algorithm is used here, but with a little change. As the parameters are shared by all time steps in the network, the gradient at each output is calculated not only based on the calculations of the current time step, but also on the previous time steps. For example, in order to calculate the gradient at t=6 we would need to backpropagate 5 steps and sum up the gradients. This method in named Backpropagation Through Time (BPTT).

RNNs have shown great success in many tasks. But the most commonly used type of RNNs are LSTMs, which are much better at capturing long-term dependencies than vanilla RNNs are. LSTM network will be explained in the next section.

2.3 Long short-term memory (LSTM) Network

LSTM is a special kind of recurrent neural network which works for many tasks, much better than the traditional version [34]. One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame or predict future frame. But RNN does not have long memory for doing a perfect prediction. Sometimes, we only need to look at recent information to perform the present task, but some times we need to have long term memory for our prediction. Long Short Term Memory networks (LSTMs) are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997) [35].

In a standard recurrent neural network, in the gradient back-propagation phase, the gradient signal is multiplied by the weight matrix of recurrent hidden layer, a large number of times (related to the number of timesteps). This is the reason of importance of magnitude of weights in the transition matrix.

If the weights in this matrix are small (smaller than 1.0), it causes vanishing gradients because the gradients become so small and learning becomes very slow or stops working. So the task of learning long-term dependencies in the data would be impossible. The vanishing gradient problem is illustrated in Figure 2.3. On the other hand, if the weights in this matrix are large (larger than 1.0), it could cause exploding gradients, it means that the gradients become so large that it can cause learning to diverge.

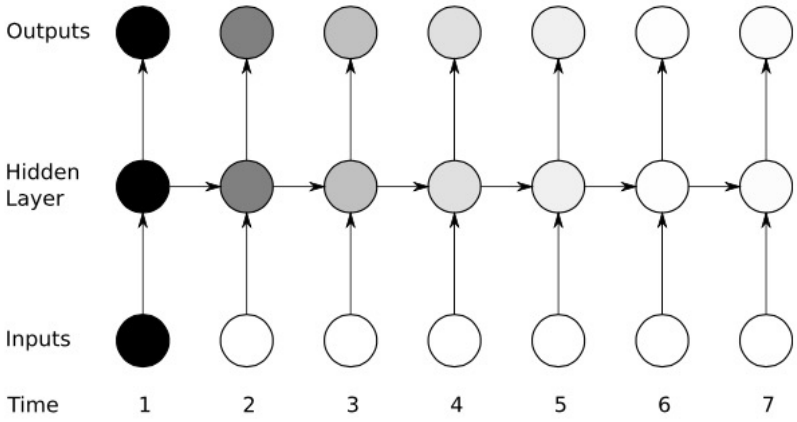

Figure 2.3: The vanishing gradient problem for RNN. The shading of the nodes in the unfolded network shows their sensitivity to the inputs at time one (the darker the shade, the greater the sensitivity). The sensitivity decrease over time as new inputs overwrite the activations of the hidden layer, and the network 'forgets' the first inputs. (From Alex Graves, 2012)[36]

These problems of RNN are the main motivation for designing the LSTM model which have a memory cell which is shown in Figure 2.4. A memory cell has four main elements: an input gate, a neuron with a self-recurrent connection (a connection to itself), a forget gate and an output gate. The weight of the self-recurrent connection is 1.0 and ensures that the state of a memory cell can remain without change in different timestep. The input gate can let incoming signal change the state of the memory cell or block it. Also, the output gate can let the state of the memory cell change other neurons or prevent it. The forget gate can let the cell to remember or forget its previous state, as needed. In the following paragraphs, these elements will be discussed further.

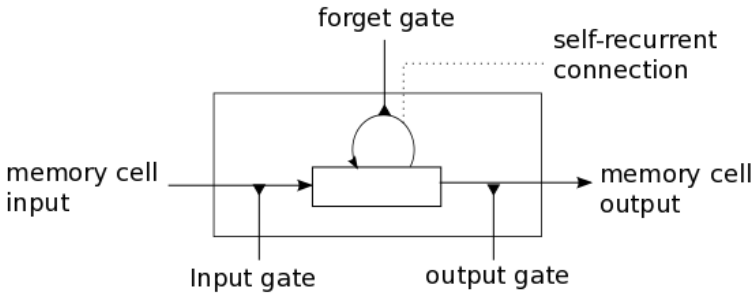

Figure 2.4: LSTM memory cell, (From Kyunghyun Cho Pierre Luc Carrier, 2017)

Gradient information is preserved by LSTM. Figure 2.5 illustrates preservation of gradient information by LSTM. As it was showed in Figure 2.3 the shading of the nodes shows their sensitivity to the inputs at time one; in the LSTM, the black nodes are maximally sensitive and the white nodes are completely insensitive. The input, forget, and output gates are illustrated below, to the left and above the hidden layer respectively. All gates are either entirely open ('O') or closed ('—'). The memory cell 'remembers' the first input until the forget gate is open and the input gate is closed. The sensitivity of the output layer can be switched on and off by the output gate without affecting the cell [36].

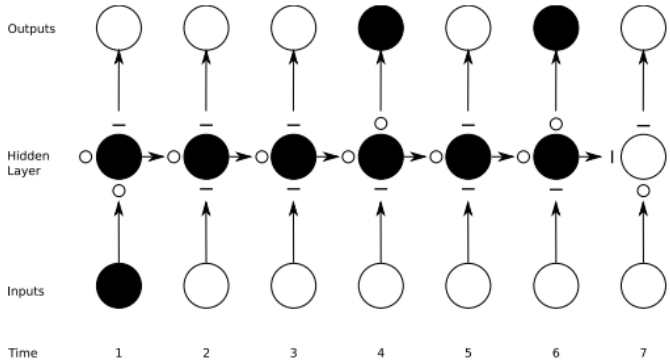

Figure 2.5: Preservation of gradient information by LSTM.(From Alex Graves, 2012)

All recurrent neural networks have the form of a chain of repeating modules of neural network. In traditional RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

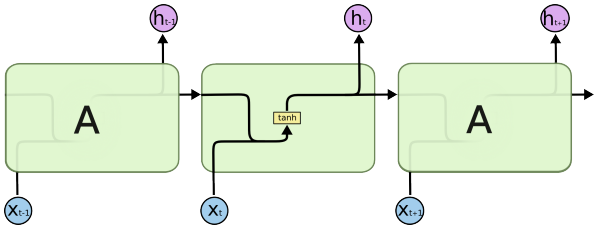

Figure 2.6: The repeating module in a standard RNN contains a single layer (From Christopher Olah, 2015)

LSTMs also have similar chain structure, but the repeating module is a bit different. Instead of having a single neural network layer, there are four, interacting in a very special way.

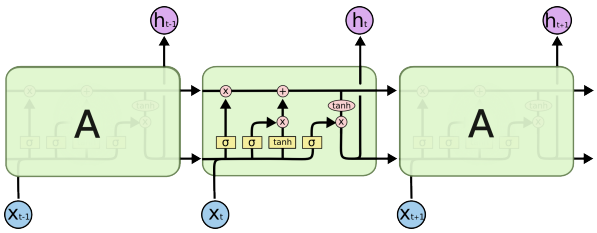

Figure 2.7: The repeating module in an LSTM contains four interacting layers (From Christopher Olah, 2015)

In the Figure 2.7, each line includes an entire vector, from the output of one node to the inputs of others. The pink circles represent operations, such as vector addition, and the yellow boxes are learned neural network layers. Lines merging shows concatenation, and a line forking address its content being copied and the copies going to different locations. This information is showed in the following figure.

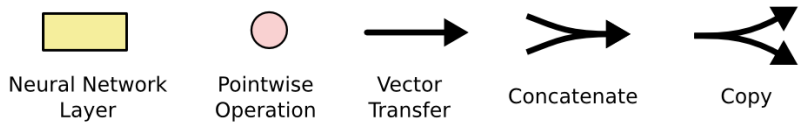

Figure 2.8: Information about different parts of LSTM diagram (From Christopher Olah, 2015)

The one important factor in LSTMs is the cell state, the horizontal line running through the top of the diagram shows cell state. The cell state is similar to conveyor belt. It connects the entire chain, with a linear interactions and information flow along it without change.

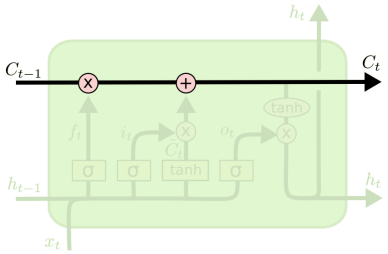

Figure 2.9: cell state in the LSTM,(From Christopher Olah, 2015)

It is possible to add or remove information to the cell state by gates. Gates are combination of a sigmoid neural net layer and a multiplication operation.

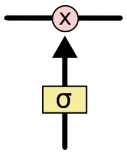

Figure 2.10: Gate in the LSTM, (From Christopher Olah, 2015)

An LSTM has three of these gates, to protect and control the cell state. As a new input comes and input gate i_t is activated, new information will be accumulated to the cell. Also, if the forget gate f_t was on, the past cell status c_{t-1} could be "forgotten". The output gate o_t controls whether the latest cell output c_t propagated to the final state h_t or not. The LSTM architecture uses memory cells to store and use information of the history, to discover long-range temporal relations. Nonlinear sigmoid $\sigma = (1 + e^{-x})^{-1}$, outputs numbers between zero and one. Value of zero means "let nothing through," while a value of one means "let everything through!"[34]. The formula of input gate i_t, forget gate f_t, output gate o_t and final state h_t is define as following:

$$i_t = \sigma(W_{x_i} x_t + W_{hi} h_{t-1} + b_i)$$
$$f_t = \sigma(W_{x_f} x_t + W_{hf} h_{t-1} + b_f)$$
$$o_t = \sigma(W_{x_o} x_t + W_{ho} h_{t-1} + b_o)$$
$$g_t = \sigma(W_{x_c} x_t + W_{hc} h_{t-1} + b_c)$$
$$h_t = o_t * tanh(c_t)$$

$$c_t = f_t * c_{t-1} + i_t * g_t$$

The main difference of LSTM with classical RNNs is the use of the these gating functions i_t, f_t, o_t , which explained previously, and indicate the input, forget, and output gate at time t respectively. Weight parameters W_{x_i}, W_{h_i}, W_{h_f}, W_{h_o},W_{x_f}, W_{h_c}, W_{x_o} and W_{x_c}, connect the different inputs and gates with the memory cells and outputs and biases b_i, b_f, b_c and b_o. The cell state c_t is updated with a fraction of the previous cell state c_{t-1} that is controlled by f_t [37].

Chapter 3

Methodology

3.1 The Data Set

In recent years, Different benchmark datasets have been developed for various vision problems. Some standard datasets such as Object Tracking Benchmark (OTB) [25] and VOT2014 [26] are very famous for Tracking task. These two challenges conveniently provide researchers with valuable dataset as well as annotated ground truth. The dataset contains different challenging visual situations. Although these data contain different challenging visual situations such as motion blur, occlusion, we need to test the proposed method in more challenging data. For having more challenging data, we need to make our own data in addition to use of the standard Tracking dataset. We chose all subsets of the OTB100 and VOT2014 which are related to human and also recorded some videos by Tobii Pro Glasses 2 and annotated them by help of hand-made software and named it Office & Supermarket data which was recorded by a head mounted camera mostly in the office and supermarket. The mentioned camera was moving as well as human that is in the video.

The whole dataset that we used in this project contain 9 attributes which represents the challenging aspects in visual tracking. These attributes are explained in table 3.1

Attributes	DESCRIPTION
Illumination Variation	the illumination in the target region is significantly changed.
Scale Variation	the ratio of the bounding boxes of the first frame and the current frame is out of the range 2
Occlusion	the target is partially or fully occluded.
Deformation	non-rigid object deformation
Motion Blur	the target region is blurred due to the motion of target or camera.
Fast Motion	the motion of the ground truth is larger than 20 pixels .
In-Plane Rotation	the target rotates in the image plane.
Out-of-Plane Rotation	the target rotates out of the image plane.
Out-of-View	some portion of the target leaves the view.
Background Clutters	the background near the target has the similar color or texture as the target.
Low Resolution	the number of pixels inside the ground-truth bounding box is less than 400

Table 3.1: Different attributes of dataset

The first frame of the target object is shown in Figure 3.1 for all sequences that used in this project.

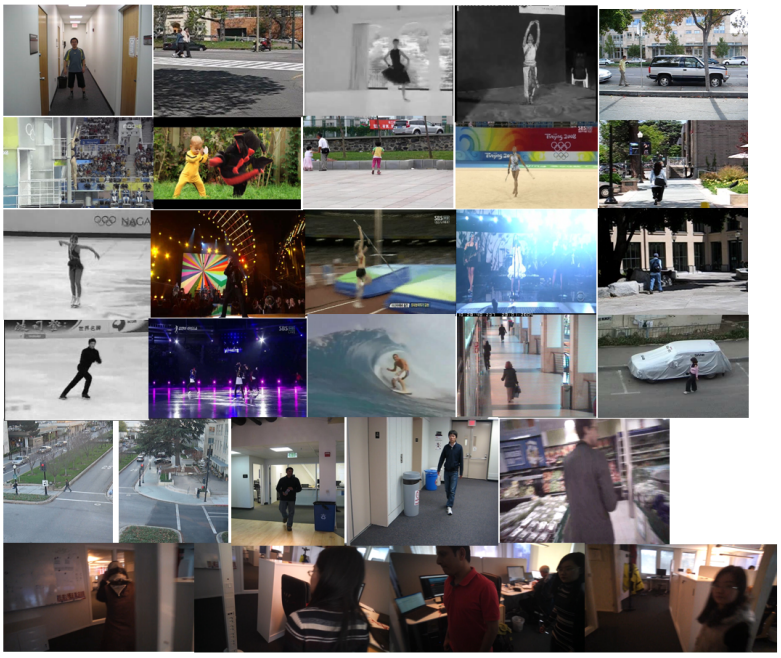

Figure 3.1: First frame of the target object for sequences that used in this project

3.2 Proposed Method

Tracking algorithms suffer from sequence-specific challenges including occlusion, deformation, lighting condition change, motion blur, etc. So finding the best algorithm that could be

invariant to these conditions is our goal.

3.2.1 General overview of proposed method

The method that we propose is trying to do a robust tracking that have a good results in the different challenges including occlusion, deformation, lighting condition change, motion blur, etc. The general and simplified overview of proposed method is shown in Figure 3.2

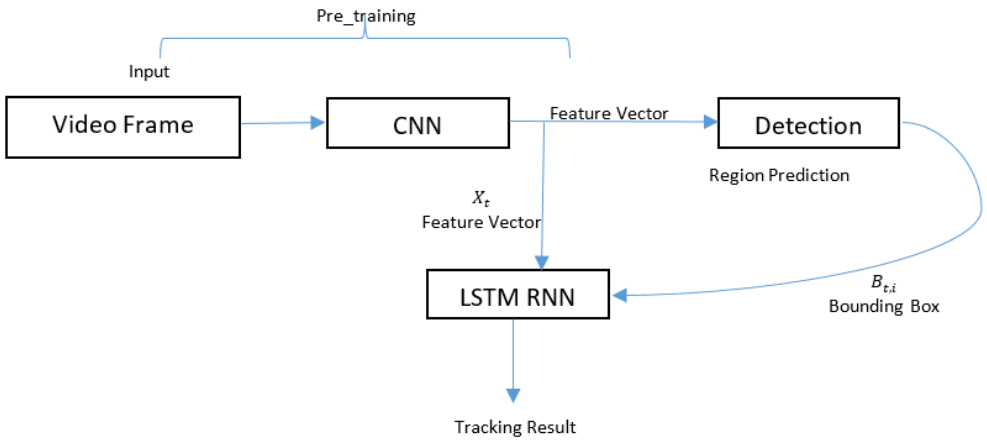

Figure 3.2: General and simplified overview of proposed method

The key motivation behind this method is that tracking failures problem can often be effectively solved by learning from historical visual information. The recurrent convolutional model is "doubly deep" because it uses both the history of locations and robust visual features of past frames. Long Short Term Memory (LSTM) unit is used as the tracking module. LSTM RNNs, uses memory cells to store and output information, allowing it to better discover long-range temporal relations. This method extends the neural network learning and analysis into the spatial and temporal domain.

The proposed model contain a deep neural network for which the input is raw video frames and it returns the coordinates of a bounding box of an object which was tracked in each frame. Tracking probability is calculated as following:

$$p(B_1, B_2, ..., B_T|\ X_1, X_2, ..., X_T) = \prod_{t=1}^{T} p(B_t|\ B_{<t}, X_{<t})$$

B_t and X_t are the location of an human and an input frame, respectively, at time t. $X_{<t}$ is a history of input frames and $B_{<t}$ is a history of previous locations of human before time t.

A traditional CNN for general feature learning is used for weight pre-training. The feature map of the whole image is produced by using convolutional neural network which takes a video frame as its input. By using this feature vector, the bounding box of detection block is produced. By using both bounding box of the detection process and feature vector as inputs of LSTM network, the tracking results is achieved. For detection part, we use YOLO and

SSD detection which have good detection results. The performance of the detection part has a strong effect on the performance of the tracking because the detection bounding box is used as a target of tracking and used as a reference for finding the tracking box. Also having robust visual features is important for having accurate tracking results.

Network Training of the Tracking Module

We use the LSTM RNNs for the training of the tracking module. In this module, LSTM has two streams of data, which are the feature vector from the convolutional layers and the detection information $B_{t,i}$ from the detection block. At each time-step t, a feature vector X_t is extracted. X_t and $B_{t,i}$ and also output of states from the last time-step S_{t-1} are inputs of LSTM network. Mean Squared Error (MSE) is used for training:

$$L_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(B_{target} - B_{prediction})^2$$

n indicates the number of training samples in a batch, $B_{prediction}$ is the tracking prediction and B_{target} is the target ground truth value. We use the Gradient Descent method for stochastic optimization.

3.2.2 YOLO detection + LSTM

Although accuracy is important in visual tracking systems, another important factor is the speed of the tracking system. Most of the tracking methods which use deep learning such as ConvNets are computationally expensive. Applying this kind of networks to each frame for visual tracking will result in high computational complexity. Recently, YOLO detection is proposed in [38]. They changed object detection from classification to regression problem to spatially separated bounding boxes and associated class probabilities. The speed of baseline YOLO model is 45 fps. A smaller version of the network, Fast YOLO, processes at 155 fps with state-of-the-art object detection performance. The structure of YOLO is illustrated in Figure 3.3

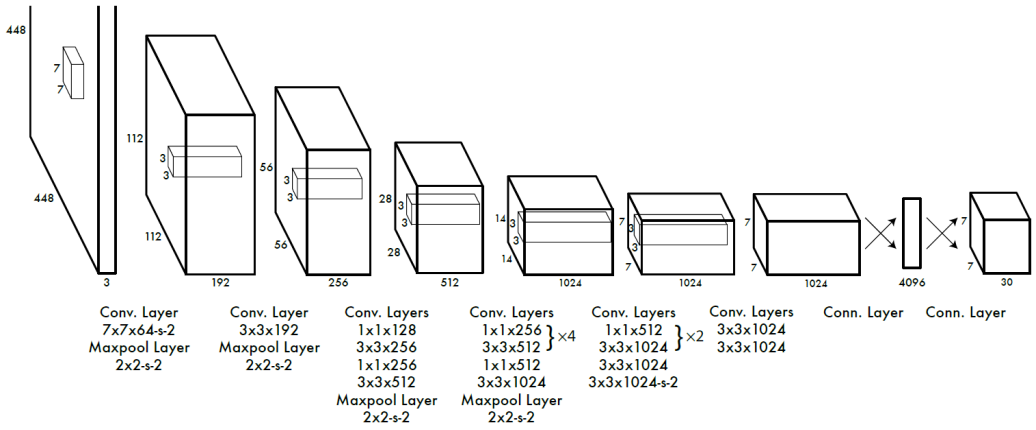

Figure 3.3: Structure of YOLO Detection. (From Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhad, 2015)

This detection network has 24 convolutional layers followed by 2 fully connected layers. The convolutional weights were learned with ImageNet data of 1000 classes such that the

network has a generalized understanding of most different visual objects. The output of the first fully connected layer is a feature vector of size 4096 which is a dense representation of the mid-level visual features. we use this feature vector as an input of LSTM network. We used the pre-trained weights of YOLO to generate visual features. After convolutional layers, YOLO have fully connected layers to regress feature representation into region predictions. These predictions are encoded as an $S \times S \times (B \times 5 + C)$ tensor. This means that the image is divided into $S \times S$ splits. Each split has B predicted bounding boxes. Each bounding box has 5 location parameters including x, y, w, h and its confidence level c. C is the class label related to each bounding box. In this project, we follow the YOLO architecture and use the S=7, B= 2 and C=20 [1]. We just use the information related to position of the object and nullify the confidence level and class label. It is explained in the section 5.2.1 that nullifying the confidence level and class label may not give us optimal result and in the future works we will change it and using the information of both confidence level and class label would help us to improve the results.

$$B_t = (0, x, y, w, h, 0)$$

Here x, y shows the coordinates of the bounding box center relative to the width and the height of the image, respectively. Parameters are Converted to [0,1] to make it easier for regression and then they are concatenated with the 4096-dimensional visual features and finally fed into the tracking module.

The structure of YOLO + LSTM is illustrated in Figure 3.4

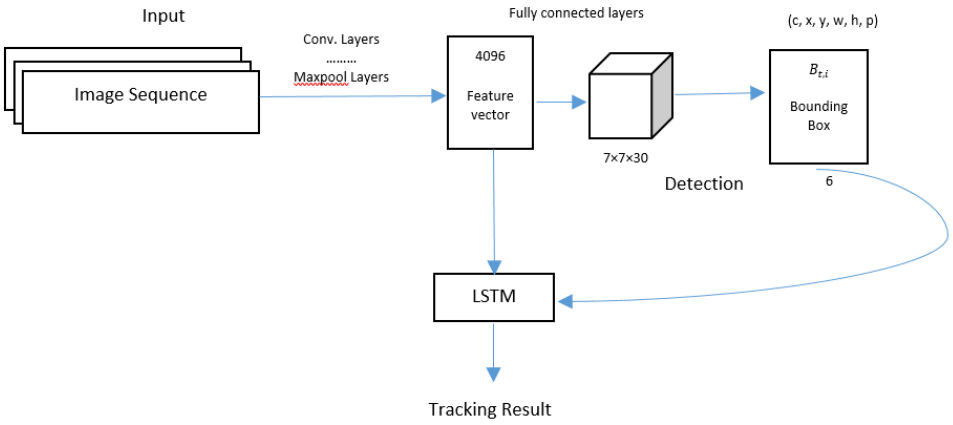

Figure 3.4: Structure of YOLO Detection + LSTM

In one frame, YOLO may output multiple detections. As we do single human tracking, we need to have one bounding box in each frame. So we design two methods for getting one bounding box in each frame. In the first method we just select the bounding box with

[1] YOLO has 20 classes contain "aeroplane", "bicycle", "bird", "boat", "bottle", "bus", "car", "cat", "chair", "cow", "diningtable", "dog", "horse", "motorbike", "person", "pottedplant", "sheep", "sofa", "train","tvmonitor". We are interested to find and track class "person". Class label determines the class of detected object.

largest confidence level. In the second method, we define a cost matrix that is computed as the intersection-over-union (IOU) distance between the current detection and the previous tracking result. For detection of the first frame, we make a decision based on the IOU distance between the detection boxes and the ground truth. Also a minimum IOU is defined to reject the detection which their IOU are less than IOU min. In Figure 3.5 method of choosing one bounding box from multiple detection bounding boxes is illustrated. The difference of the method ROLO which was mentioned in section 1.3.3 is in using these two methods for choosing one bounding box from multiple bounding boxes. In ROLO, the single bounding box was chosen by comparing the detection bounding boxes with ground truth in each frame. The box with more overlap with ground truth was chosen. But in real experiment, ground truth is not available so we propose these methods instead.

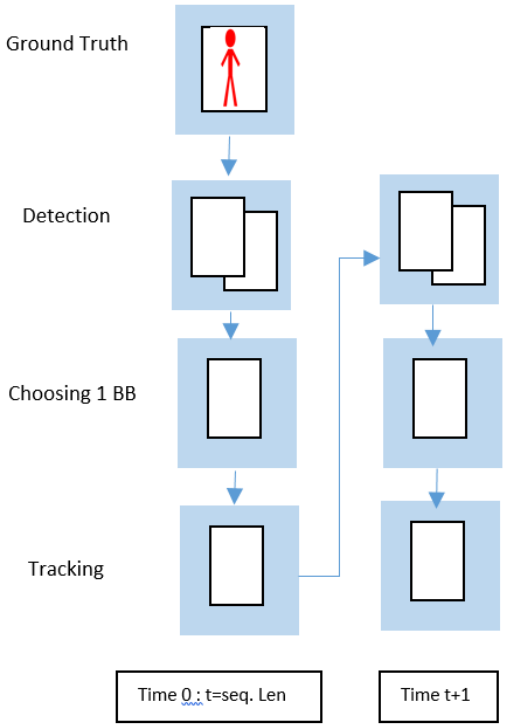

Figure 3.5: Method of choosing one bounding box from multiple detection bounding boxes

3.2.3 SSD detection + LSTM

As mentioned in previous section, the accuracy of bounding box detection has a great effect in the accuracy of tracking results. Since SSD is one of the detection methods which has a acceptable accuracy, we decided to use it as a detection part of our algorithm. One of the structural difference of SSD and YOLO is that SSD does not have fully connected layers. So we have to map the a convolutional layer to a feature vector for feeding in to a LSTM network.

The Structure of SSD detection is illustrated in Figure 3.6

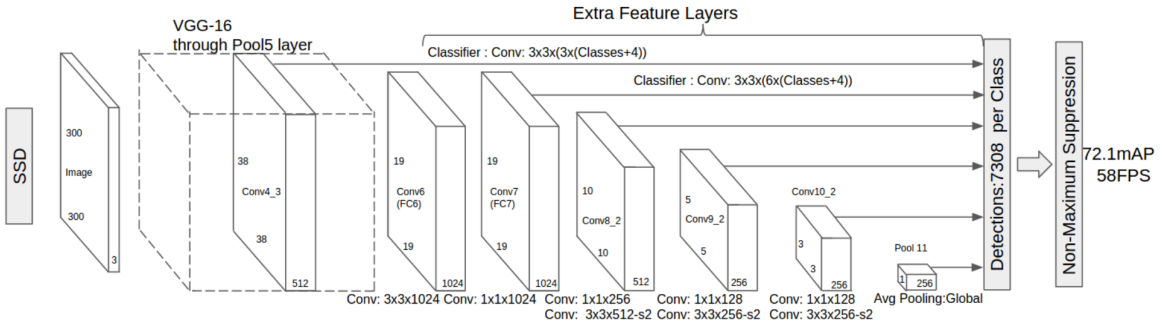

Figure 3.6: Structure of SSD Detection. (From Wei Liu, Dragomir Anguelov, Dumitru Erhan, 2016)

The SSD method is based on a feed-forward convolutional network and results in a fixed number of bounding boxes and scores of object class in each boxes [39]. It's structure has two parts. First one is based on a standard architecture used for high quality image classification and the second part is related to producing detection boxes. The detection of the SSD has two main features. First, it has multi-scale feature maps for detection. It means that it has convolutional feature layers at the end of the base network where the size decrease progressively and help to predict the detection boxes at multiple scales. Second is convolutional predictors for detection. It means that each convolutional layer could predict certain number of detection boxes by using convolutional filters. Comparing with YOLO, YOLO have fully connected layer instead of a convolutional filter for this purpose.

Another question that occur here is which convolutional layer is more suitable and contain robust visual features. It seams that layers 6 or 7 would have more features related to the positions of objects since other layers have low level features comparing to layer 6 and 7. We design different experiments to find the best layer for our work. Convolutional layers 6 and 7 have the same dimension of $19 \times 19 \times 1024$. This is equal to 369664 features. Our GPU does not have the ability to manage this size of data as an input of LSTM network. We chose to use 13 channels from 1024 channels. So the size of the feature vector is 4693 ($19 \times 19 \times 13 = 4693$). Other convolutional layer are not so suitable for our work. For example convolutional layer 4 has a dimension of $38 \times 38 \times 512$, We choose 4 channels of 512 channels. The values of three channels are zero, which can indicate dead filters, and can be a symptom of high learning rates. So this layer does not have enough features for our work. Also convolutional layer 8 structure is $10 \times 10 \times 512$, We chose 40 channels of 512 channels. From these 40 channels, just 14 of them were not dead filter and not zero. Finally we choose convolutional layer 6 and 7 for our experiments. These layers also have some dead filters but have more visual features for feeding the LSTM. The structure of this method is illustrated in Figure 3.7

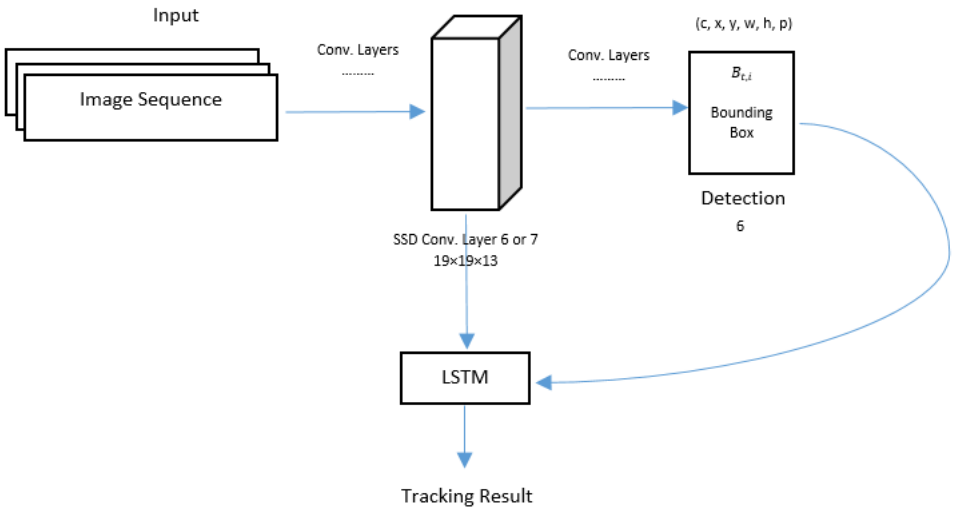

Figure 3.7: Structure of using SSD detection + LSTM

For training process of this method, we have the pre-trained weights able to generate visual features of convolutional layers 6 and 7 and also detection boxes. The model that was used was trained with PASCAL VOC data and fine-tuned by Office & Supermarket Data. After producing the feature vector and bounding boxes, we train the tracking network which contains the LSTM network. As SSD produces multiple boxes and we only need one box for tracking single person, we chose the bounding box with max confidence level for using as input for LSTM box.

3.2.4 SSD detection +YOLO Features + LSTM

As we thought that YOLO features are more robust for using as an input of LSTM network and because SSD convolutional layers contain some dead filter, It seems that it is not so suitable as an input for LSTM networks and they have not enough information for using as a feature vector. So we design another experiment which uses YOLO feature vector and relative SSD detection boxes and mix them up for using as an input for LSTM network. The structure of this method is illustrated in Figure 3.8

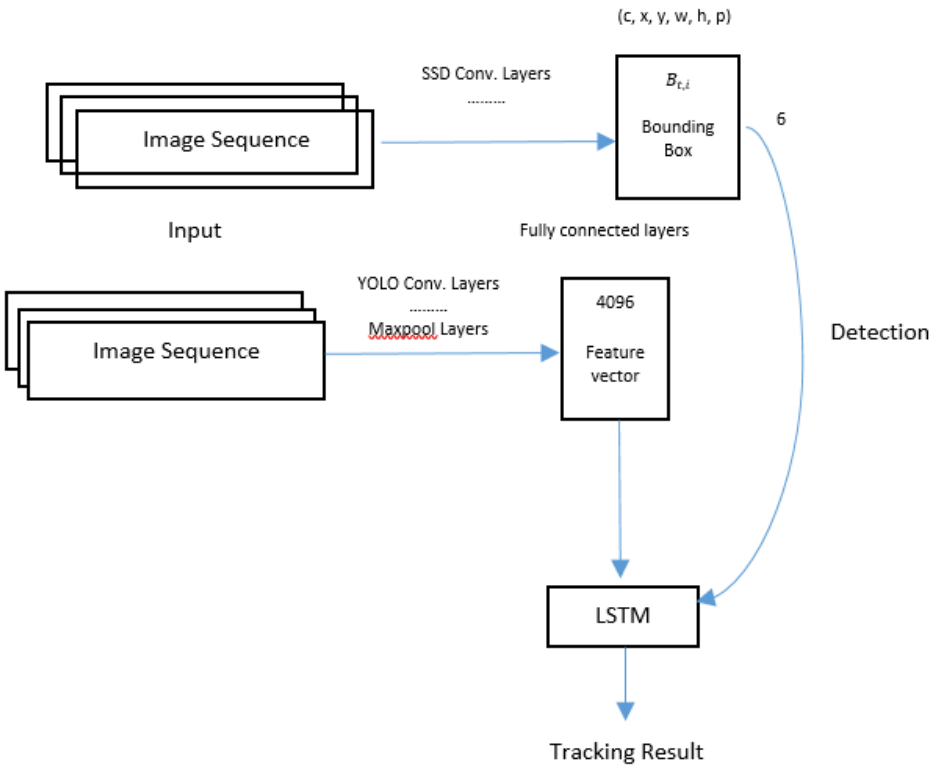

Figure 3.8: Structure of SSD detection +YOLO Featurs + LSTM

For the training process of this method, we have the both pre-trained weights for YOLO and for SSD to generate visual features of YOLO and detection boxes of SSD. After producing the feature vector and bounding boxes, we train the tracking network which contains the LSTM network.

3.3 Implementation Details

We use TensorFlow for implementing three layer LSTM network. For training we used 25 videos from OTB100 and VOT2014 with different sequence numbers and also two long Office & Supermarket video with 2000 sequences. Also we fine-tuned [2] our model to Office & Supermarket videos. The most important parameters used in training network is indicated in Table 3.2

All samples from the training set (27 videos containing around 20, 000 frames) are examined in with batch size which was set to 16, and training was performed for 27*500 iterations with learning rate parameters shown in Table 3.2. The dimension of input data for different propose structure is indicated in the Table 3.3. The experiments are conducted on Windows

[2]Fine tune means: First pretrain a deep net on a large-scale dataset. Then, given a new dataset, and start with these pretrained weights when training on new task.

Step number	Batch number	Learning rate	Iteration
6	16	0.00001	20000

Table 3.2: Parameters of network

Proposed Method	Number of Feature	Number of Prediction	Number of input
YOLO + LSTM	4096	6	4102
SSD + LSTM	$(19 \times 19 \times 13 = 4693)^*$	6	4699
YOLO + SSD + LSTM	4096	6	4102

Table 3.3: Dimension of input data for different propose structure. * 13 filter is chosen from 1024 filter. The real size is $19 \times 19 \times 1024$

10, with Intel(R) Xeon(R) CPU E5-1650 v3 and 16,0 GB RAMand GPU GeForce GTX 1080.

Chapter 4

Results and Discussion

It is difficult to say that which of the qualitative or quantitative evaluation is better for tracking problem. Also it is challenging task to measure the accuracy of a tracking method with quantitative metrics. Many factors such as position accuracy, robustness, tracking speed, memory requirement can be considered. Even in one frame with the tracking output and ground-truth object state, there can be several metrics to measure accuracy [25]. Some times tracking results show that tracker does not loose the object but the size of the bounding box is smaller than ground truth. If we average the evaluated values of all frames in an image sequence, the evaluation may not be fair because a tracker may have lost the target in the beginning or in the spatial sequences but could have tracked the target successfully in other frames.

4.1 Quantitative Results

4.1.1 Tracking Performance

One of the quantitative evaluation methods that is used for evaluating our tracking is Precision plot. It is determined by the center location error, which computes the average euclidean distance between the center locations of the tracked targets and the ground truth positions of all the frames. When a tracking algorithm loses track of a target object, the output location can be random, and thus, the average error value does not measure the tracking performance correctly [17]. If the estimated locations are within a given threshold distance of the ground-truth would be evaluated, it would be more accurate evaluation [17], [40].

Another commonly used evaluation metric is the overlap score [24]. The overlap score is defined as $S = \frac{r_t \cap r_0}{r_t \cup r_0}$, where r_t is tracked bounding box and r_0 is the ground-truth bounding box.

Also, $\cap$ and $\cup$ represent the intersection and union operators, respectively. The average overlap score (AOS) can be used as the performance measure. Also, the overlap scores can be used for determining whether an algorithm successfully tracks a target object in one frame, by counting the cases that S is larger than a different thresholds and plot these results to show the success rate within a different given thresholds.

For temporal robustness evaluation, each tracking algorithm is evaluated several times from different starting frames in the video stream. So, each time, every algorithms are evaluated from a particular starting frame until the end of the video stream. The average of

tracking results (success rate) is calculated and TRE (Temporal Robustness Evaluation) score is generated.

So we evaluate our proposed tracking with the quantitative evaluation methods: AOS and plot of Success rate and TRE.

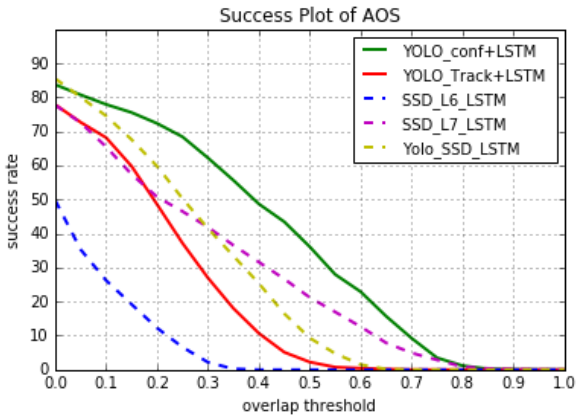

Figure 4.1: Success rate for Office & Supermarket Data with multiple Human in the video

Figure 4.1 shows the success rate for Office & Supermarket Data with multiple Human in the video with 2000 frames. It is clear that the results of method $YOLO_{conf} + LSTM$ have better result than others. $YOLO_{conf} + LSTM$ is related to the method that for each frame, one box of detection was chosen based on max confidence level. $YOLO_{Track} + LSTM$ is related to the method that for each frame, one box of detection box was chosen based on the previous tracking results. YOLO + SSD + LSTM which contain the YOLO features and SSD bounding box information gives the second best results. Figure 4.2 shows the success rate of TRE for Office & Supermarket Data with multiple humans in the video where each tracking algorithm is evaluated 20 times from different starting frames in the video stream and the average is calculated and plotted. Threshold zero is not exactly zero and it is equal to 0.0001. It is arranged to be a bit more than zero to see the percentage of wrong tracking bounding boxes. For example in Figure 4.2 by reading the value from the graph at overlap threshold=0, it is clear that in 12 percent of frames, there is wrong tracking boxes and there is not any overlap between ground truth and tracking box.

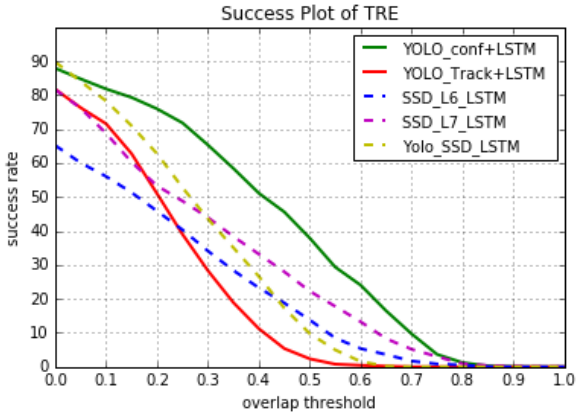

Figure 4.2: Success plot of TRE for Office & Supermarket Data with multiple human in the video

Proposed Method	Thr=0	Thr=0.1	Thr=0.2	Thr=0.4	Thr=0.6	Thr=0.8
$\text{YOLO}_{conf} + LSTM$	0.83	0.77	0.72	0.48	0.22	0.09
$\text{YOLO}_{Track} + LSTM$	0.77	0.68	0.48	0.18	0.07	0.0
$\text{SSD}_{L7} + LSTM$	0.77	0.65	0.50	0.41	0.21	0.0
$\text{SSD}_{L6} + LSTM$	0.49	0.35	0.12	0.02	0.0	0.0
YOLO + SSD + LSTM	0.85	0.74	0.59	0.25	0.15	0.0

Table 4.1: Success rate in different Threshold for Office & Supermarket Data with multiple human in the video

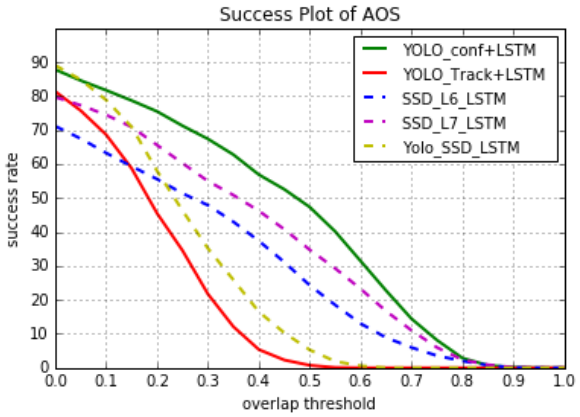

Figure 4.3: Success rate for general Office & Supermarket Data

Proposed Method	Office & Supermarket Data with multiple human in the video	General Office & Supermarket data	VOT data
$YOLO_{conf}$ + LSTM	0.40	0.41	0.56
$YOLO_{Track}$ + LSTM	0.20	0.18	0.29
SSD_{L7} + LSTM	0.26	0.35	0.37
SSD_{L6} + LSTM	0.20	0.28	0.31
YOLO + SSD + LSTM	0.25	0.24	0.32

Table 4.2: AOS (Average Overlap Score) for Different Data

Table 4.3 shows a summary of Average Overlap Score (AOS) for 8 top trackers plus YOLO + LSTM for OTB data-set and Office & Supermarket. The results of the other trackers are reported from [23]. YOLO + LSTM achieves the best performance for some of the test video sequences.

Sequences	YOLO LSTM	STRUCK [12]	CXT [41]	TLD [42]	OAB [43]	CSK [40]	RS [44]	VTD [45]	CNN SVM [10]
Human2	0.56	0.65	0.24	0.39	0.61	0.16	0.52	0.18	0.61
Human9	0.35	0.06	0.06	0.15	0.17	0.22	0.60	0.24	0.35
Gym	0.58	0.35	0.42	0.27	0.06	0.21	0.41	0.36	0.43
Human8	0.35	0.12	0.10	0.12	0.09	0.17	0.33	0.24	0.45
Skater	0.60	0.55	0.58	0.32	0.48	0.43	0.57	0.47	0.57
BlurBody	0.50	0.69	0.66	0.39	0.67	0.38	0.27	0.23	0.60
Woman	0.63	0.69	0.20	0.12	0.46	0.19	0.35	0.14	0.73
Tobii	0.41	NA	NA	NA	NA	NA	NA	NA	NA

Table 4.3: Summary of Average Overlap Score (AOS) for 9 top trackers for OTB data-set. (Results of other tracker used from Guanghan Ning, Zhi Zhang, Chen Huang, Zhihai He, 2016)

4.1.2 Parameter Sensitivity

Step number indicates the number of previous frames which are used each time for a prediction by LSTM. In previous experiments, we used 6 for step number. For finding the best step number which has good effect on the performance and speed of the tracking, we perform some experiments and results show that the optimum number for Step number is 6. The result of effect of the Step number on the performance of the tracking which was done on the model YOLO + LSTM is shown in Figure 4.4. As it is clear from figure, the performance of having 6 frames as a history of LSTM is better than others. So we choose step number equal to 6 in all our experiments.

Figure 4.5 illustrates the speed of tracking in different step number given to LSTM.

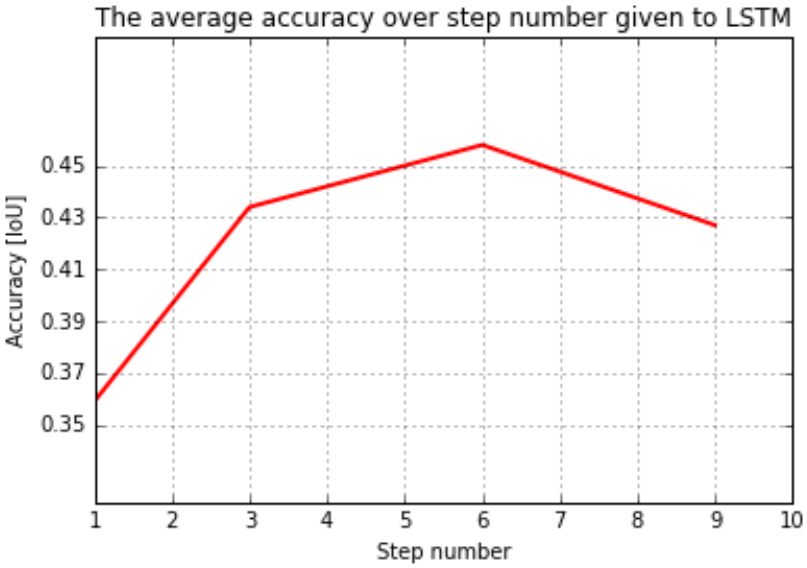

Figure 4.4: The average accuracy over the numbers of step given to LSTM

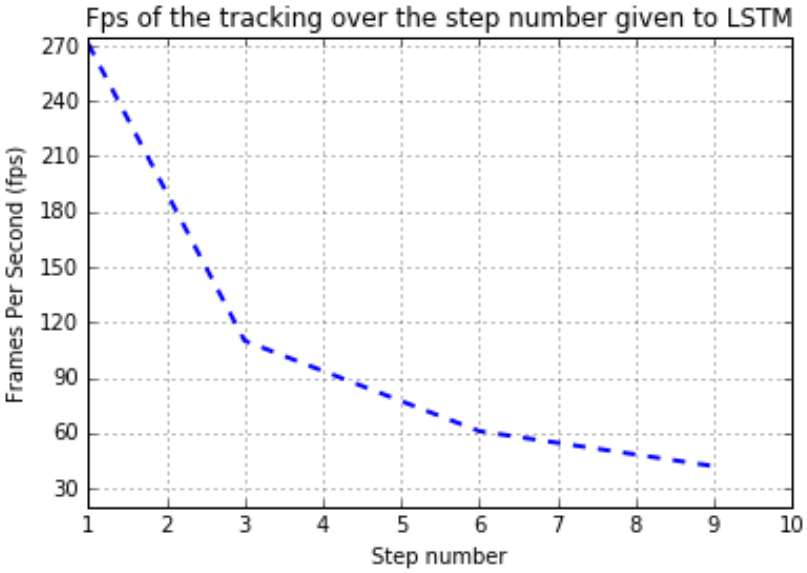

Figure 4.5: Fps of the tracking over the numbers of step given to LSTM

4.2 Qualitative Results

Qualitative evaluation methods could be more comprehensible for human to see the results in different condition and judge about the performance of the tracking.

We try to show the qualitative results in different challenging conditions such as occlusion, motion blur, illumination variation, wrong detection information, rotation, shape change and fast motion and discuss about them.

Figures number 4.6 to 4.12 are related to the results of method $YOLO_{conf} + LSTM$ and Office & Supermarket dataset. Green bounding box shows tracking results and blue bounding boxes show detection results.

Figure 4.6 shows the case where the target person is occluded by other human. The green bounding boxes which shows the tracking results, could follow the target human. Detection results in some frames contain both persons.

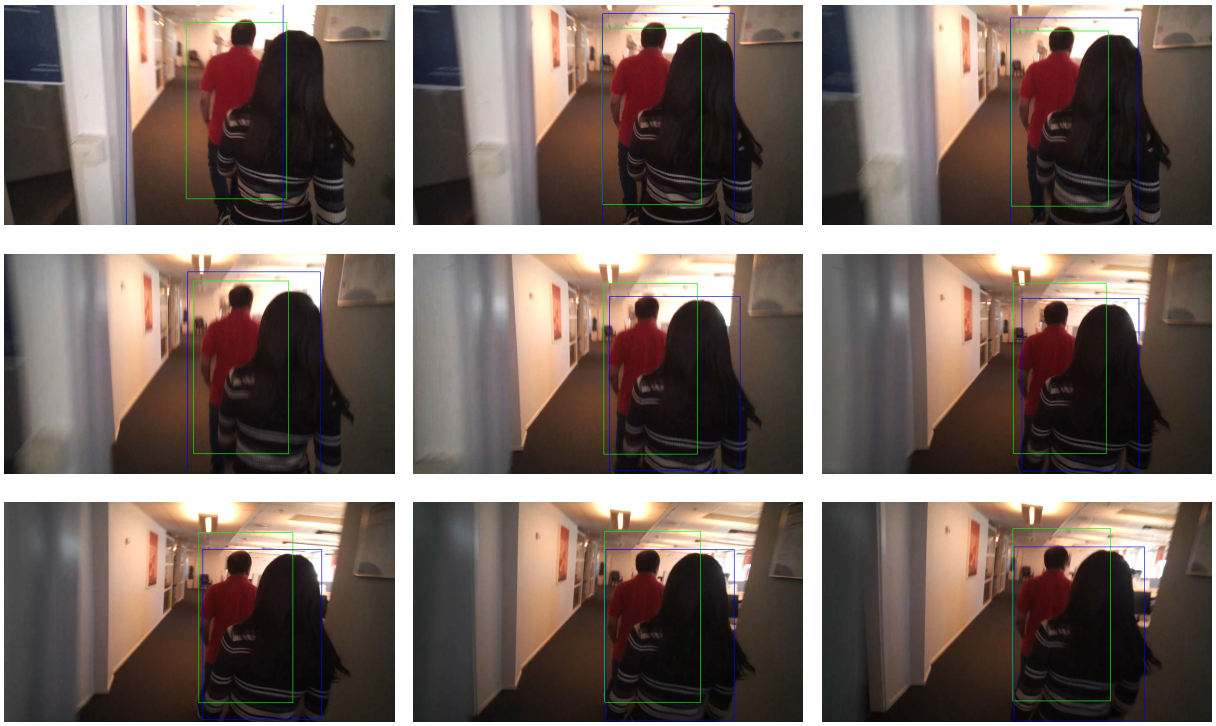

Figure 4.6: Examples of continuous frames of Office & Supermarket dataset, green bounding box shows tracking results and blue bounding boxes show detection results in the case of partial occlusion

Figure 4.7 shows the case where the detection switch to another person and has wrong position but tracking continue to follow the the target person. It is because LSTM uses the information from the history of the frames so the failure of the detection in some frames could not deceive the tracking from the following the target.

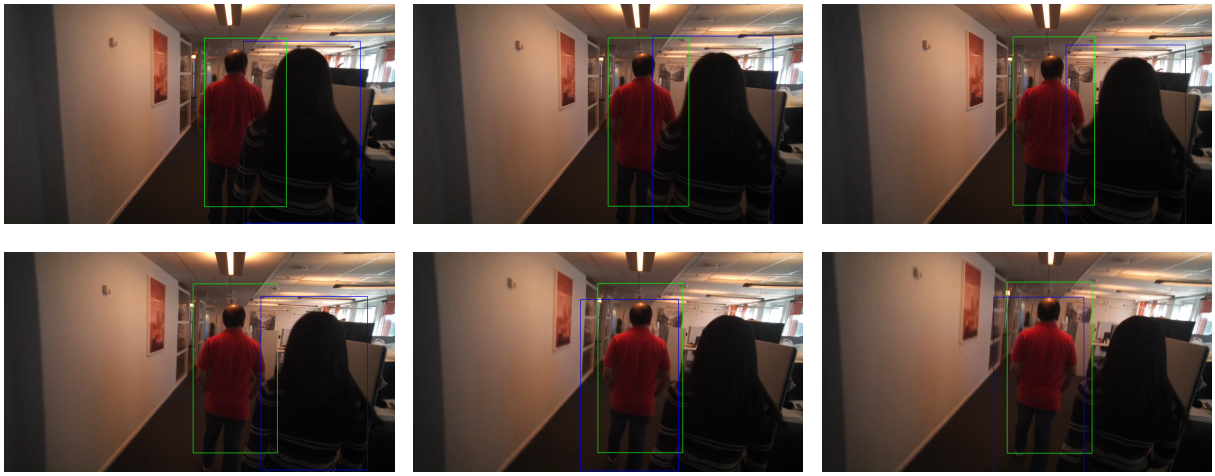

Figure 4.7: Examples of continuous frames of Office & Supermarket dataset, green bounding box shows tracking results and blue bounding boxes show detection results in the case of wrong detection

Figures 4.8 and 4.9 show the case that there is both motion blur and illumination change. It is clear that the tracking results, could follow the target human in these challenging situation.

Figure 4.8: Example of continuous frames of Office & Supermarket dataset, green bounding box shows tracking results and blue bounding boxes show detection results in the case of motion blur and illumination change

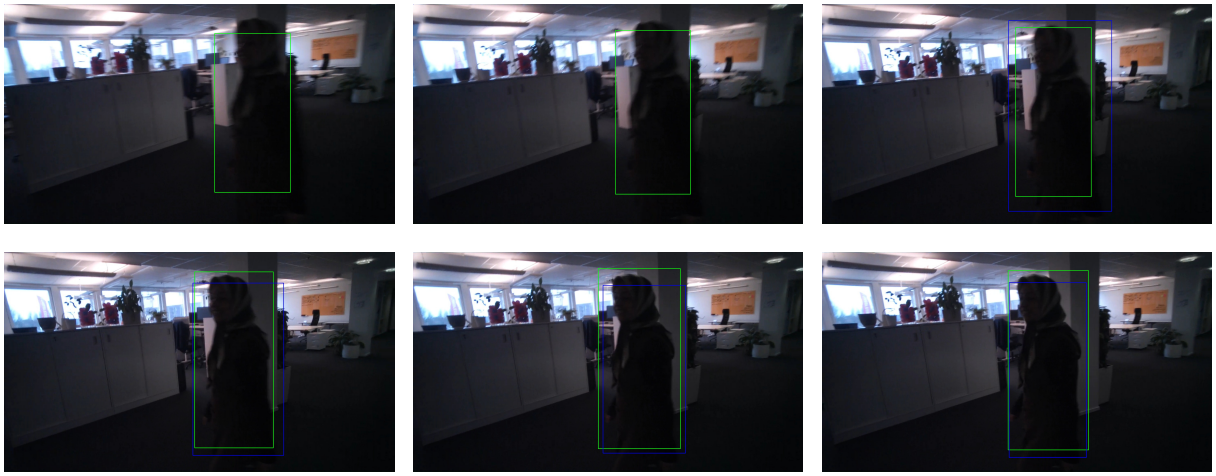

Figure 4.9: Example of continuous frames of Office & Supermarket dataset, green bounding box shows tracking results and blue bounding boxes show detection results in the case of motion blur and illumination change

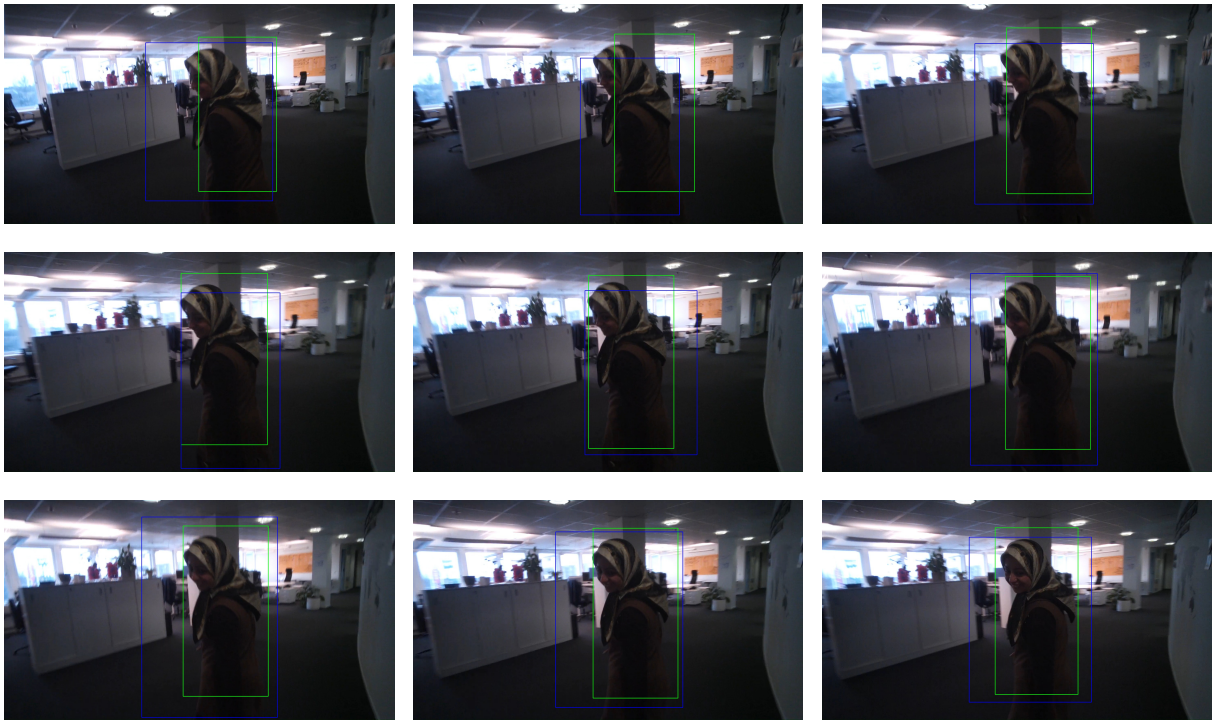

Figure 4.10: Example of continuous frames of Office & Supermarket dataset, green bounding box shows tracking results and blue bounding boxes show detection results in the case of human turning

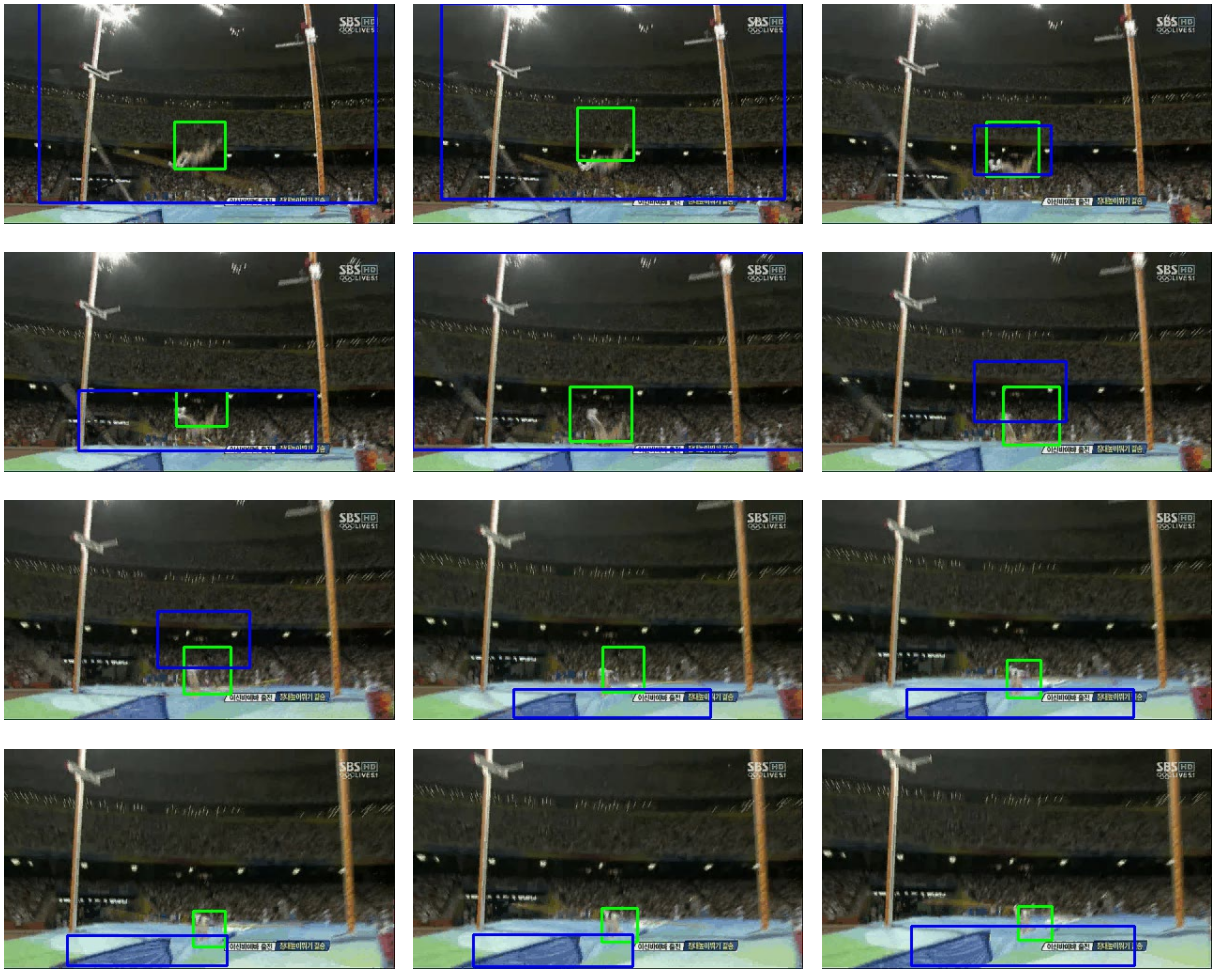

Figure 4.11: Example of continuous frames of VOT data, green bounding box shows tracking results and blue bounding boxes show detection results in the case of deformation of body and motion blur and very small size of human in the video

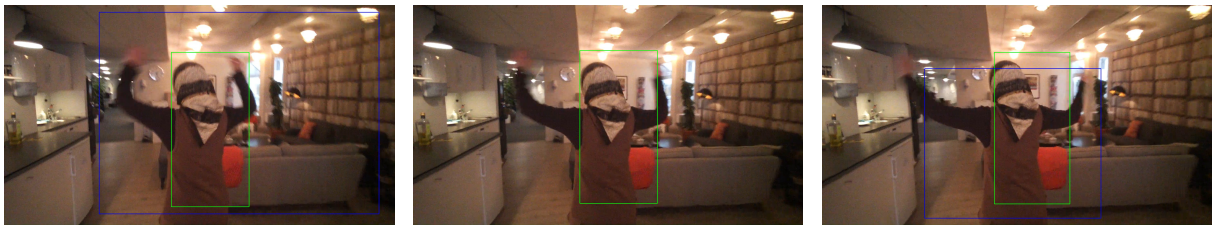

Figure 4.12: Example of continuous frames of Office & Supermarket dataset, green bounding box shows tracking results and blue bounding boxes show detection results in the case of deformation of body

Figures 4.10, 4.11 and 4.12 show the case that there is turning and deformation of the human body. It is clear that the tracking results could follow the target human but when the human opens her arms and her body's shape is changed, tracking could follow the human but the size of the bounding box does not cover her arms in Figure 4.12. These kinds of cases affect the quantitative results and decrease the success rate because AOS is based on the overlap

of the bounding box and tracking bounding box. In this case the main purpose of tracking which is following the target is achieved but the whole human body is not covered. In Figure 4.11 although the body of the human has rotation of 360 degree, the tracker successfully could follow the human.

For a case where the human is in the left or right corner of image, proposed method couldn't follow the person well. Figures 4.13 shows an example for this case. In this example it is clear that the false detection bounding boxes have also deceiving effect on the tracking result, but the main reason for this problem is that tracking object in the training data are mostly in the center of the images. It is not statistically balanced in different positions. So the lack of database which has a statistically balanced made this problem for us.

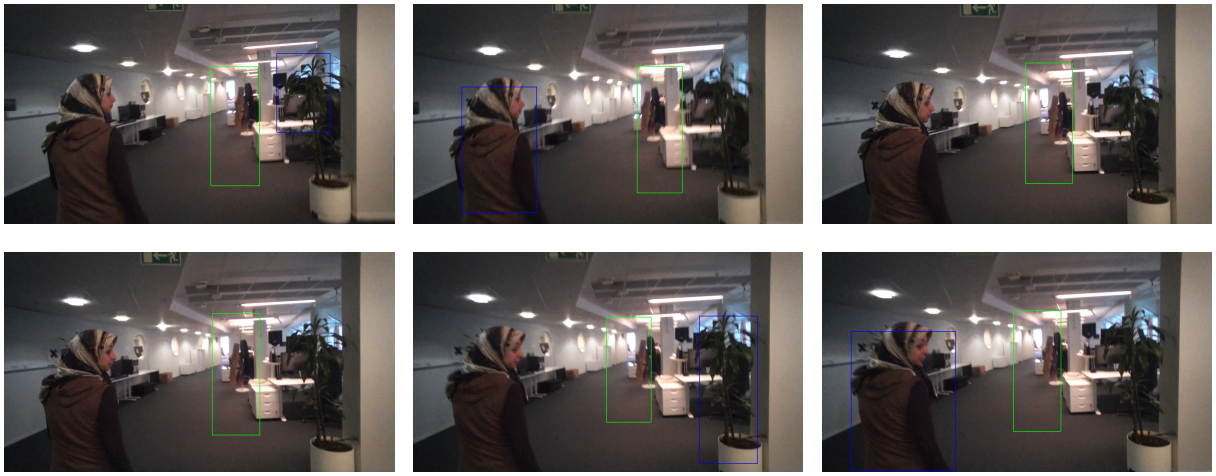

Figure 4.13: Example of continuous frames of Office & Supermarket dataset, green bounding box shows tracking results and blue bounding boxes show detection results in the case of human in the corner of the image

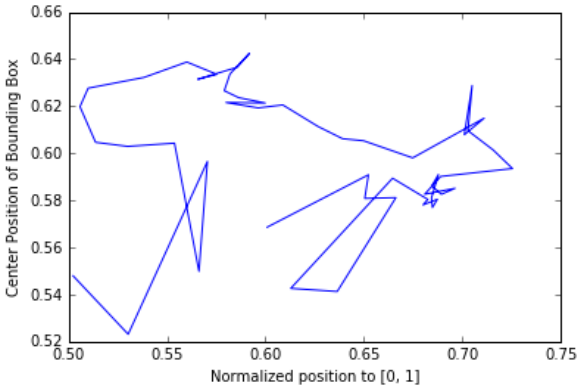

Figure 4.14: Trace of the center of the tracking bounding box for 50 frames of a video

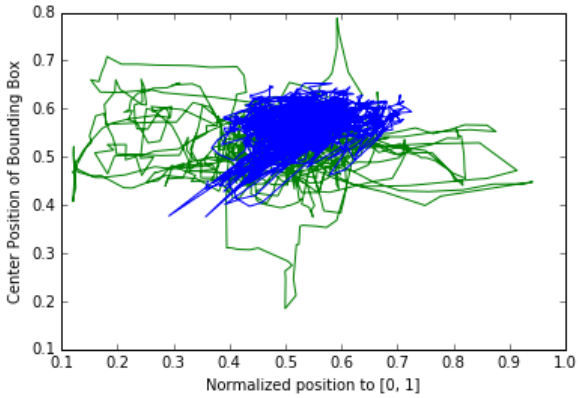

Figure 4.15: Trace of the center of the tracking bounding box for 2000 frames of a video. Blue is related to tracking and Green is for ground truth.

Figures 4.14 and 4.15 show the trace of the center of the tracking bounding box for 50 and 2000 frames of a video respectively. Figures 4.14 shows that tracking could trace the human in different places but in 4.15 it is clear that mostly the place of the box is in the center of the image. Although in the majority of frames of ground truth, bounding box is in the center, there is some bounding boxes in the corners which the tracking couldn't find them. The reason of this problem is related to lack of spatially balanced training data base which have statistically balanced in place of object in the different frames. In most of the available data and also the training data which is used for this project, tracking object are mostly in the center of the images.

Chapter 5

Conclusion

This thesis has been proposed with the aim of developing a model for having robust human tracking. We proposed various models to achieve better results. Different approaches have been tried for different tasks during the development of the proposed algorithms implementation which was discussed in the previous chapters.

5.1 Summary of Findings

We have developed a method of spatially supervised recurrent convolutional neural networks for visual human tracking. This method extends the deep neural network learning. We change it in the way of using history of video for increasing the tracking performance and track human in the challenging dataset Office & Supermarket. This dataset have been recorded with a head mounted camera and is more challenging than VOT or OTB. Using information about previous frames helps to track human in the cases of motion blur, occlusion, change of illumination etc. Also, the proposed method could track the human after disappearing for short time and appearing again in the video. In the case where the human is in the corner of the image, proposed method could not successfully follow him. The reason is spatially imbalanced training data. LSTM + YOLO uses the robust image features as well as their spatial location supervision. This method is spatially deep, as it is capable of using the visual features and location of detecting bonding boxes. It is also temporally deep by using temporal features as well as their possible locations. As a result this method can effectively track in the occlusion and severe motion blur. We also examined other combinations of trackers such as SSD with LSTM network. It seems that YOLO's visual features are more suitable for using as a input of LSTM network and have more compatible information for our work.

5.2 Future Work

Although the proposed human tracking could follow human in many of the challenging conditions, its performance could be improved by doing some more changes in the proposed method. The limitations which reduce the robustness of the method and some suggestions for its improvement are explained in this chapter.

5.2.1 Limitation and suggestion for improvement

Using multiple object detection for single object tracking

One part of the inputs of proposed tracking methods is detecting bounding boxes of a human. So the accuracy of tracking is affected by the accuracy of detection. Since these detectors outputs more than one bounding boxes in each frame, we proposed two methods for choosing one bounding box from multiple bounding box, the first is based on max confidence level and second is based on the previous frame tracking results. These methods were explained in the previous chapters.

So in two cases, the input of LSTM network is not correct and it would deceive the network. First is the case that detection has a false result and second is the case that choosing a wrong bounding box from several detection bounding box.

As LSTM uses the history of some previous frames, these mentioned problems does not affect it seriously. So robust tracking would need to expect that output from the detector is not perfect and it would need to handle false positions of detection. However having more accurate detection boxes would help to improve tracking performance in the future and having robust tracker. We have some suggestions in this regard.

1. Using a more accurate detection method instead of YOLO and SSD.
2. Choosing the detection bounding box based on the both confidence level and class of object which was detected. It means that it is better, first check the class label and filter the non-human bounding boxes and then use the max confidence level and choose the best bounding box. Also we can determine a threshold for confidence level to filter the small confidence levels.
3. Using other creative methods for choosing one bounding box from multiple bounding boxes.

Problems of feature vectors of SSD

As mentioned before, the accuracy of detection affects the accuracy of tracking. So choosing a detection method with a high performance is preferable. It seems that SSD outperforms YOLO [39]. But in addition to using bounding boxes, it is essential to have a rich and robust visual feature vector to feed the LSTM. Unlike YOLO, SSD does not have any fully connected layers, so we have to use features from convolutional layers. Layers 6 and 7 which was chosen to use as a input of LSTM, have a dimension of $19 \times 19 \times 1024 = 369664$. Since this size of dimension is too big to feed the LSTM network, 13 filters were chosen from these 1024 filters. As a result the feature vector of the SSD detector is not as rich as YOLO's feature vector. It would be useful to find a way to reduce the dimension of features of SSD. Having worthy features is essential to improve the tracking performance. One suggestion could be using PCA for reducing the dimension of SSD's features. It would be implemented in future work for achieving better results.

Imbalanced training data

Lack of spatially balanced training data base which is statistically balanced in place of object in the different frame is the other limitation for our works. In most of the available data for tracking, human is in the center of the image. Since our testing data recorded from a head-mounted camera in the real world and the camera also has some stride movement, there are some wrong results in the case where the human is in the corner of the image. So having

a data that is statistically balanced in the position and scale of human could help to improve the the tracking results effectively.

5.2.2 More development

One of the important steps in the development of the proposed method is generalizing this single human tracking to multiple human tracking. The second step is identifying humans in the multiple human tracking and assign an id for each of the humans in the video. In these cases if a new person appears in the video, the tracker could track him, and it would also be able to find a person who has disappeared in some frames and appeared again and give him the same id as before.

5.3 Ethical and societal implication

We are living in the technology and information age which every thing including research, economic activities, etc could affect human privacy. Especially the Deep Learning research area, which is related to human essentially needs big databases which may affect human privacy. Gathering, storing, manipulating data are creating ethical dilemmas.

In our work, for training and testing process, the VOT, OTB and Office & Supermarket data are used. Office & Supermarket data is recorded by our team. So we are responsible for it. The problem is that usually we can not get permission from every single person in the video. How can we do these types of studies which are related to human behaviour, without violating their personal integrity?

Another main problem is the final usage of the proposed method which would be used in any product that needs human tracking. Could it affect the privacy of humans? For answering this question it is essential to explain shortly the application of the proposed method in the real applications.

As mentioned before, the purpose of using this head mounted camera is mostly behavioral studies and research and this kind of camera would not be used by common people. One example of using this glasses is using it for human computer interaction experiments which is mostly used in the universities. Another application of this product is related to study human behavior for economic reasons and this kind of usage is mostly done in the shopping malls. As mentioned before, it may not be feasible to get the permission of every single person that may appear in a video. But there are some suggestion for respecting human privacy. First is making a equipped lab for each study case; this suggestion seems expensive and not applicable for most cases. Second suggestion is anonymization. It means that, blur people's faces so that they can not be recognized in every publication and every single image that is ever posted on the Internet or any public forum. So it is possible to apply a face recognition and blurring techniques to the video before doing any publication.

So, the duty to respect human privacy absolutely is one of the most important consideration of users of this kind of technology. People should feel comfortable with all the data that is collected and their faces should not be recognizable and identities should not be logged anywhere without permissions of them. Anonymization of data should always be done to the greatest extent possible.

Bibliography

[1] Jialue Fan, Wei Xu, Ying Wu, and Yihong Gong. Human tracking using convolutional neural networks. *IEEE Transactions on Neural Networks*, 21(10):1610–1623, 2010.

[2] Tao Zhao and Ramakant Nevatia. Tracking multiple humans in complex situations. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1208–1221, 2004.

[3] Ralf Plänkers and Pascal Fua. Tracking and modeling people in video sequences. *Computer Vision and Image Understanding*, 81(3):285–302, 2001.

[4] Fengliang Xu, Xia Liu, and Kikuo Fujimura. Pedestrian detection and tracking with night vision. *IEEE Transactions on Intelligent Transportation Systems*, 6(1):63–71, 2005.

[5] Sergey Ioffe and David Forsyth. Human tracking with mixtures of trees. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 690–695. IEEE, 2001.

[6] Fan Yang and Michel Paindavoine. Implementation of an rbf neural network on embedded systems: real-time face tracking and identity verification. *IEEE Transactions on Neural Networks*, 14(5):1162–1175, 2003.

[7] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[8] Seunghoon Hong, Hyeonwoo Noh, and Bohyung Han. Decoupled deep neural network for semi-supervised semantic segmentation. In *Advances in Neural Information Processing Systems*, pages 1495–1503, 2015.

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[10] Seunghoon Hong, Tackgeun You, Suha Kwak, and Bohyung Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *ICML*, pages 597–606, 2015.

[11] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115 (3):211–252, 2015.

[12] Sam Hare, Stuart Golodetz, Amir Saffari, Vibhav Vineet, Ming-Ming Cheng, Stephen L Hicks, and Philip HS Torr. Struck: Structured output tracking with kernels. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2096–2109, 2016.

[13] Jun Feng Li. *Sequential Monte Carlo methods for multiple target tracking*. PhD thesis, University of Cambridge, 2008.

[14] Shai Avidan. Ensemble tracking. *IEEE transactions on pattern analysis and machine intelligence*, 29(2), 2007.

[15] Helmut Grabner and Horst Bischof. On-line boosting and vision. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 260–267. IEEE, 2006.

[16] Helmut Grabner, Christian Leistner, and Horst Bischof. Semi-supervised on-line boosting for robust tracking. In *European conference on computer vision*, pages 234–247. Springer, 2008.

[17] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Robust object tracking with online multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1619–1632, 2011.

[18] Fan Yang, Huchuan Lu, and Ming-Hsuan Yang. Robust superpixel tracking. *IEEE Transactions on Image Processing*, 23(4):1639–1651, 2014.

[19] Le Zhang and Ponnuthurai Nagaratnam Suganthan. Visual tracking with convolutional neural network. In *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*, pages 2072–2077. IEEE, 2015.

[20] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.

[21] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4293–4302, 2016.

[22] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Stct: Sequentially training convolutional networks for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1373–1381, 2016.

[23] Guanghan Ning, Zhi Zhang, Chen Huang, Zhihai He, Xiaobo Ren, and Haohong Wang. Spatially supervised recurrent convolutional neural networks for visual object tracking. *arXiv preprint arXiv:1607.05781*, 2016.

[24] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[25] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.

[26] Matej Kristan, Roman Pflugfelder, Ales Leonardis, Jiri Matas, Fatih Porikli, Luka Cehovin, Georg Nebehay, Gustavo Fernandez, Tomas Vojir, Adam Gatt, et al. The visual object tracking vot2013 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 98–111, 2013.

[27] Zhibin Hong, Zhe Chen, Chaohui Wang, Xue Mei, Danil Prokhorov, and Dacheng Tao. Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 749–758, 2015.

[28] Jianming Zhang, Shugao Ma, and Stan Sclaroff. Meem: robust tracking via multiple experts using entropy minimization. In *European Conference on Computer Vision*, pages 188–203. Springer, 2014.

[29] Jin Gao, Haibin Ling, Weiming Hu, and Junliang Xing. Transfer learning based visual tracking with gaussian processes regression. In *European Conference on Computer Vision*, pages 188–203. Springer, 2014.

[30] Martin Danelljan, Gustav Häger, Fahad Khan, and Michael Felsberg. Accurate scale estimation for robust visual tracking. In *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press, 2014.

[31] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[32] Wei Zhong, Huchuan Lu, and Ming-Hsuan Yang. Robust object tracking via sparsity-based collaborative model. In *Computer vision and pattern recognition (CVPR), 2012 IEEE Conference on*, pages 1838–1845. IEEE, 2012.

[33] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[34] Christopher Olah. Understanding lstm networks. *Avalible: http://colah.github.io/posts/2015-08-Understanding-LSTMs/fn1*, 2015.

[35] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[36] Alex Graves. Supervised sequence labelling. In *Supervised Sequence Labelling with Recurrent Neural Networks*, pages 5–13. Springer, 2012.

[37] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In *Advances in Neural Information Processing Systems*, pages 3882–3890, 2016.

[38] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[39] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[40] Joao Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Exploiting the circulant structure of tracking-by-detection with kernels. *Computer Vision–ECCV 2012*, pages 702–715, 2012.

[41] Thang Ba Dinh, Nam Vo, and Gérard Medioni. Context tracker: Exploring supporters and distracters in unconstrained environments. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1177–1184. IEEE, 2011.

[42] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2012.

[43] Helmut Grabner, Michael Grabner, and Horst Bischof. Real-time tracking via on-line boosting. In *Bmvc*, volume 1, page 6, 2006.

[44] Hirotaka Niitsuma. A trainable object-tracking method using equivalent retinotopical sampling and fisher kernel. 2003.

[45] Junseok Kwon and Kyoung Mu Lee. Visual tracking decomposition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1269–1276. IEEE, 2010.