Thesis for the Degree of Master

DNN based Real-time Pedestrian Tracking

DNN 기반 실시간 보행자 추적

June 2019

Department of Information
Communication Technology

Graduate School of Soongsil University

Nguyen Van Ngoc Nghia

Thesis for the Degree of Master

DNN based Real-time Pedestrian Tracking

DNN 기반 실시간 보행자 추적

June 2019

Department of Information

Communication Technology

Graduate School of Soongsil University

Nguyen Van Ngoc Nghia

Thesis for the Degree of Master

DNN based Real-time Pedestrian Tracking

A thesis supervisor: Prof. Sun-Tae Chung

Thesis submitted in partial fulfillment of the requirements for the Degree of Master

June 2019

Department of Information Communication Technology

Graduate School of Soongsil University

Nguyen Van Ngoc Nghia

To approve the submitted thesis for the
Degree of Master by Nguyen Van Ngoc Nghia

Thesis Committee

Chair	한영준	(signature)
Member	김강희	(signature)
Member	정선태	(signature)

June 2019

Graduate School of Soongsil University

ACKNOWLEDGEMENT

Firstly, I would like to thank my advisor Prof. Sun-Tae Chung. He has an enormous contribution to my thesis topic and during my studying in Soongsil University, from giving me ideas, advices, step by step to apply ideas, make improvements and correcting this thesis. He is always happy to support me even when he is very busy.

Secondly, I would like to express my gratitude to my thesis committee members: Prof. Han Young Jun and Prof. Kim Kang Hee for their advices, insightful comments. It helps me a lot in completing this thesis.

Besides that, I would like to thank my lab-mates for their ideas, fruitful discussion during lab meetings and they always happy to help me when I needed. We have a lot of fun and memories together both inside and outside laboratory. Those moments help me a lot in refreshing myself to keep my productivity.

Last but not the least, Thank my family, my friends for their continuous support, their encouragement. That means a lot to me to keep my motivation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

DNN based Real-time Pedestrian Tracking

Nguyen Van Ngoc, Nghia
Department of Information Communication Technology
Graduate School of Soongsil University

Even though so much progresses have been achieved in Multiple Objects Tracking (MOT), most of reported MOT methods are not still satisfactory for commercial embedded products like Pan-Tilt–Zoom (PTZ) camera. In this thesis, we propose a DNN-based real-time multiple pedestrian tracker for embedded environments. The proposed tracker is constructed based on TBD (Tracking By Detection). First, a new light-weight convolutional neural network (CNN)-based pedestrian detector is devised, which is designed to have dual path architecture over a light-weight MobileNet image classifier so as to detect pedestrians fast and even small size pedestrians as well. For further saving of processing time in tracking, the devised detector is applied for every other (2) frame, and Kalman filter (KF) is employed to predict positions of pedestrian in frames where the devised CNN-based detector is not applied. The facing direction orientation information is incorporated with IOU (Intersection Over Union) to enhance Hungarian based object association for tracking pedestrians while keeping similar computational cost. Through experiments

on Jetson TX2 board, it is verified that the devised pedestrian detector detects pedestrians even at small size fast and well when it is compared to many state-of-the-art detectors, and that the proposed tracker can track pedestrians fast and show accuracy performance comparably to many state-of-the-art tracking methods, which don't target for operating in embedded systems.

국문초록

DNN 기반 실시간 보행자 추적

응웬반응억냐
정보통신공학과
숭실대학교 대학원

다중 객체 추적(Multiple Object Tracking, 이하 MOT) 방법에 대해 많은 기술적 진전이 이루어졌지만, 보고된 대부분의 MOT 방법들은 PTZ 와 같은 상업적 임베디드 시스템에 사용되기에는 아직은 만족스럽지 않다.

본 논문에서는 딥러닝 네트워크 기반의 실시간 다중 보행자 추적 방법을 제안한다. 제안된 보행자 추적기는 검출 기반 추적 방법(TBD; Tracking By Detection) 에 기반하여 설계된다. 먼저, 새로운 가볍고, 작은 크기의 보행자도 검출할 수 있는 CNN(Convolutional Neural Network) 기반 다중 보행자 검출기를 고안하였다. 이 고안된 보행자 검출기는 경량 이미지 분류기인 MobileNet 의 기반 위에 작은 크기의; 보행자도 검출할 수있도록 2 중 패스 구조로 설계되었다. 추적시에, 추가적인 처리 시간의 절약을 위해 이러한 다중 보행자 검출기는 한 프레임 걸러 매 2 프레임 마다 적용되도록 하고, 검출기가 적용되지 않은 프레임에서의 다중 보행자 추적을 위해 칼만필터를

적용하였다. 프레임간의 객체의 연관을 위해, 객체간의 IOU(Intersection over Union) 정보에 더하여 보행자 방향 정보 (왼쪽, 오른쪽, 앞면, 뒷면 등)의 정보를 추가적으로 이용한 헝가리안 연관 알고리즘을 적용하였다. 보행자 방향 정보는 보행자 검출기의 출력 정보로 얻어지도록 CNN 을 설계하고 훈련하도록 설계되었고, 헝가리안 연관성 알고리즘 연산시에 별로 연산량을 추가하지 않는다.

엔비디아의 임베디드 컴퓨팅 보드인 Jetson TX2 에 구현 적용한 실험을 통해, 본 논문에서 제안한 보행자 추적방법이 최신 객체 검출기에 비해 작은 크기의 보행자 검출도 신속하고 잘 수행하며, 추적 정확도 성능에 있어서도 현재의 최신 다중 객체 추적 방법들에 비견할 정도의 성능을 보이면서 실시간으로 추적을 수행할 수 있음을 확인하였다.

CHAPTER 1 Introduction

1.1 Motivation

Object tracking such as pedestrian tracking is an important task in the field of computer vision applications such as traffic monitoring, visual surveillance, pedestrian-computer interactions, autonomous vehicle driving, biology and so on [1]. Many excellently performing object tracking methods are more concerned with tracking accuracy. Most of such methods are still computationally expensive for embedded systems.

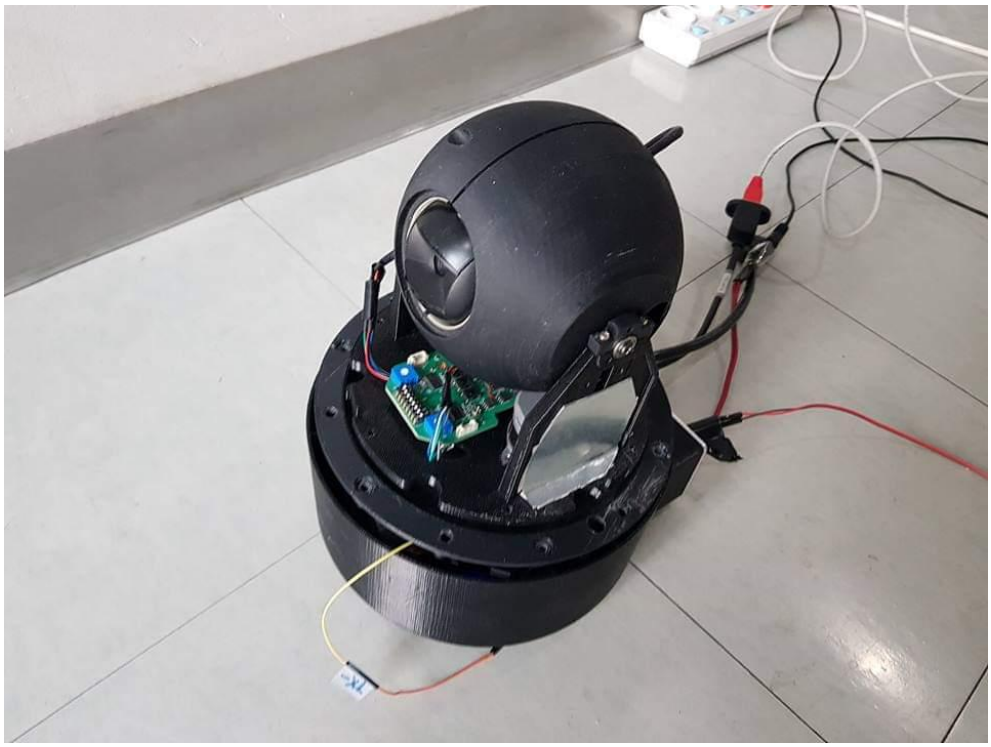
Because of the above reasons, I need to develop a system which is more suitable for our developing Pan–Tilt–Zoom (PTZ) camera with Jetson TX2 [2] as a main processor.

1.1.1 Jetson Tx2 board.

[Figure 1-1] Jetson Tx2 board

Jetson Tx2 is a popular and powerful embedded module for developing Artificial Intelligence applications. It's built around an NVIDIA Pascal™-family GPU and loaded with 8GB of memory and 59.7GB/s of memory bandwidth. It is equipped with HMP Dual Denver 2/2 MB L2 + Quad ARM® A57/2 MB L2, CPU @ 2GHz, 8GB of RAM. NVIDIA Pascal™, 256 CUDA cores. It features a variety of standard hardware interfaces that make it become easier to integrate into a wide range of products and form factors.

1.1.2 Pan–Tilt–Zoom camera



[Figure 1-2] Pan–Tilt–Zoom camera

PTZ camera is a camera which is able to rotate, zooming. The moving direction and zooming ratio of the PTZ camera can be controlled by software or hardware. PTZ camera is an important part of modern surveillance systems. It can direct the attention to suspicious events.

1.2 Research Objective

In this thesis, I propose a real-time multiple pedestrians tracking method with a good tracking accuracy for our developing PTZ camera with Jetson TX2 [2] as a main processor.

The proposed multiple pedestrians tracking method is based on Tracking-By-Detection (TBD) [3,4,5,6]. TBD based object tracking works in 2 main stages. First, object detection and next, association among detected objects between two consecutive frames.

For the detection part of the proposed tracking method, I design a light weight CNN-based pedestrian detector with a good detection performance of even small size pedestrians which produces pedestrian locations as bounding boxes and pose orientations of detected pedestrians. Since the lightly designed CNN-based pedestrian detector is not computationally light enough for embedded processing, I apply the detector and Kalman filter-based object predictor, alternatively. The possible prediction error (position and size of the object) in a frame will not accumulated since it is readjusted in the next frame by the CNN-based pedestrian detector, which is more reliable than the Kalman filter-based prediction. For association, I boost the accuracy of Hungarian algorithm [7] by incorporating pose

orientation information together with Intersection Over Union (IOU) into the association metrics, which does not increase association processing speed further. The moving direction information, which is adopted in LKDeep [6] is determined not to be adopted in association metrics since moving direction under PTZ environments takes time to compute exactly.

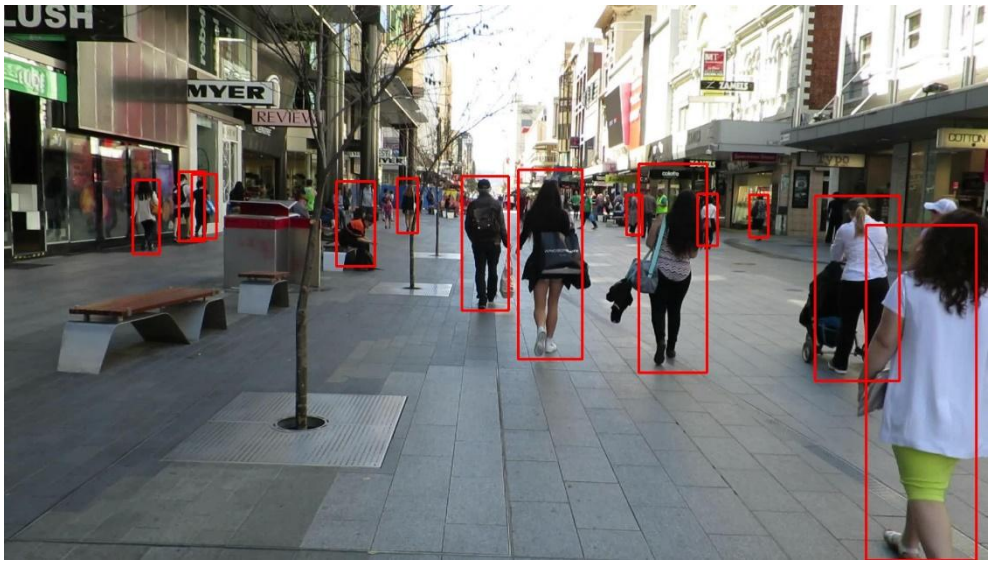
CHAPTER 2 Related works

The recent successfully object tracking methods are based on TBD due to significant improvements in object detection. In TBD, an object detector is first applied to find the target object bounding boxes, then an association rule is applied to associate the newly detected target objects in the current frame with the detected targets in the previous frame.

2.1 Object Detection

Recent object detection includes 2 main development trends, 2 stage detector such as Faster R- CNN [8], and single shot detector such as YOLOv2 [9] and SSD [10]. For the real-time processing purpose, single shot detector outperforms 2 stage detectors. On the other hand, single shot detectors perform worse in detecting small objects. Many proposed object detectors [11,12,13] including the third version of YOLO [14] (YOLOv3 [11]) improves the detection performance of small objects by employing several techniques or adopting new features. For example, YOLOv3 improves accuracy on small objects by designing a new backbone – darknet53 and making detection at 3 scales. All of those improved object detectors should pay more computational power for better detection performance. Thus, those improved object detectors are not suitable for real-time embedded applications. Tiny YOLOv3 is a faster version of YOLOv3, however its detection performance of small objects is sacrificed for faster processing. In this thesis, I design a new CNN-based pedestrian

detector which improves the detection performance of small size pedestrian without further processing burden.

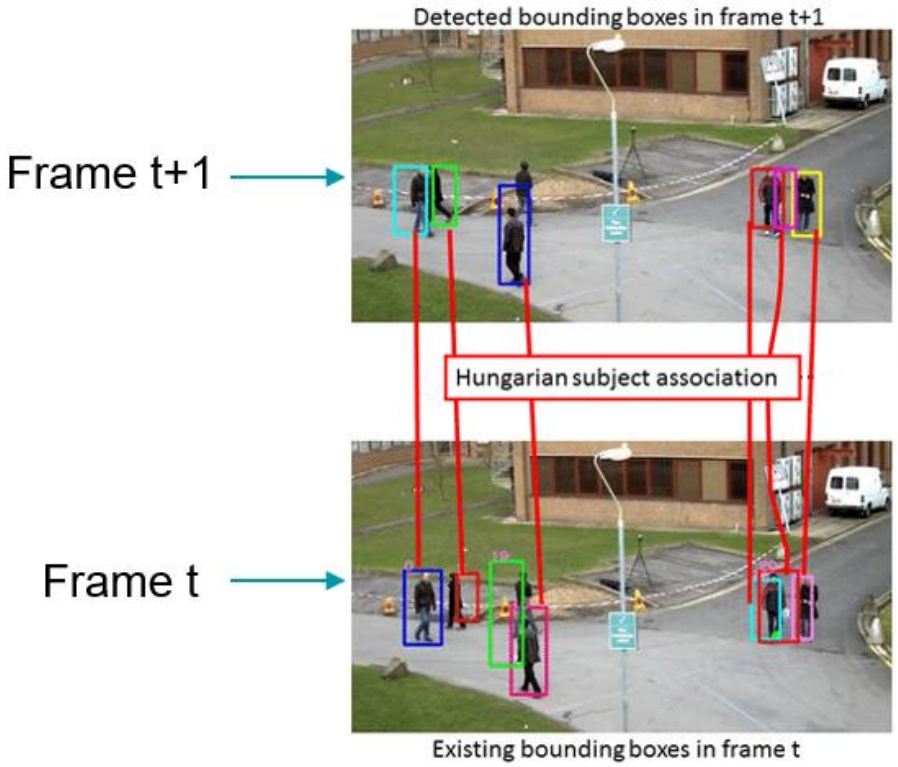


[Figure 2-1] Example result of people detection

2.2 Bounding Boxes Association

In TBD-based tracking, a conventional way to solve the association problem is to use Hungarian algorithm. Most of the previous TBD based methods using Hungarian for association merely depended on distance or the overlap ratio. Many of those methods such as SORT [3] and IOU Tracker [5] are only based on bounding box position. However, such association methods usually confuse when objects are overlapped, which leads to very much identity switches in crowd scenes. To alleviate occlusion problems, [6] employs a moving direction information more for Hungarian association. Together with the development of Deep Learning, the authors in Deep-SORT [4] introduce deep association metric. that uses not only the

location but also the appearance features. Similarly, [15] introduces a network to match a pair of object detections and use the similarity score for data association. However, calculating the appearance features and similarity score between objects are expensive and therefore may not be appropriate for the embedded environment.



[Figure 2-2] Bounding box association

The recent fast tracking algorithms merely depend on the precision of the detector. Simple Online and Real-time Tracking (SORT) [3] simply predicts the current position of previous detected people by KF. After obtaining the predicted location, SORT used Hungarian method on the IOU between predicted location and

detected location to do association. Overall, this association method is very similar to my association methods. The main different is in calculating cost metrics, SORT uses only IOU between current detected bounding boxes and the predicted KF of previous bounding boxes. This method only works correctly if the targets are sparse and there is no overlap between people. In the case of people crossed by others, SORT usually fail to find the correct identities to each people. This leads to identity switches problem in crowd scenes.

IOU Tracker [5] is another fast tracking method, this method heavily depends on the accuracy of the detector. The association rule simply iterates over all the detected targets, calculating the IOU with all the previous detected target and the choose the maximum value of IOU to match. In each frame the detector provides a list N people $D = \{D_{1,}, D_{2,}, \dots D_{N,}\}$ and a list of previous detected people $T = \{T_{1,}, T_{2}, \dots T_{M}\}$. IOU tracker iterates over D, for each D_i candidates, IOU Tracker finds the maximum IOU between D_i and T. Let say, D_i and T_j give the maximum IOU, IOU tracker decides D_i and T_j are matched. With simple association rule, association part of IOU tracker can reach up to thousands of frames per second. The speed and accuracy of this method depends heavily on the detector. Association only use IOU distance and ignore appearance features lead to a lot of identity switches. This algorithm also requires a reliable people detector.

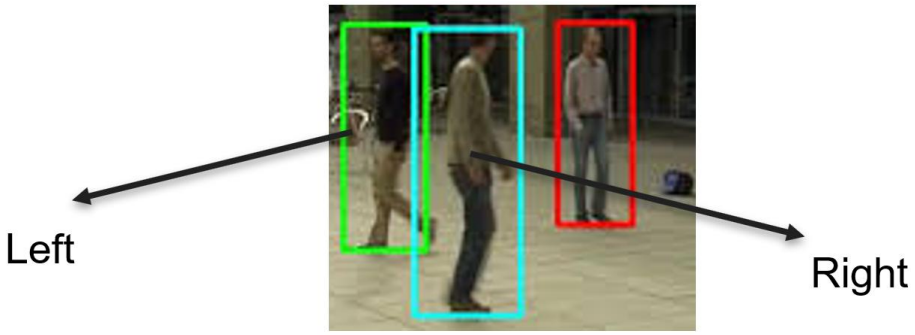
Deep-sort [4] uses a much more effective approach to solve association problem. It integrates motion and appearance features to make association decision.

Deep-sort uses an effective human re-identification Deep Learning network to extract appearance features. Since the network works well on re-identifying people, the quality of the extracted appearance features is very high. With high quality appearance features, Deep-sort works well to associate bounding boxes. However, in order to calculating the deep appearance features of a person, Deep-sort need to pass an image of that person through network. Extracting appearance features requires strong computing power especially when there are many people in the scene, since the network have to process many times.

LKDeep [4] is another fast tracker developed for PTZ camera. [4] customized the network of Tiny YOLOv2 for people detection. The customized network is fast. however, it is weak in detecting people, especially small people. In association [4] used Lucas-Kanade method to keep tracking a list of extracted feature points of each people and based on the tracked points to make associations. The more feature points are used to track, the more reliable in association. Therefore, when the number of people in frame become larger, such as crowd street, the set of feature points become very larger result in slow in extracting keypoints and association. [4] is more suitable at tracking in close distance view where people are scattered and the views are close, like in a small room.

In this thesis, I propose a new simple association algorithm employing pose orientation in addition to IOU for Hungarian association, which boosts the Hungarian association accuracy without sacrificing processing speed.

Pose orientation means where a person looks towards; front, back, right, and left. By incorporating this useful pose orientation information, the proposed association significantly reduces identity switches problem.



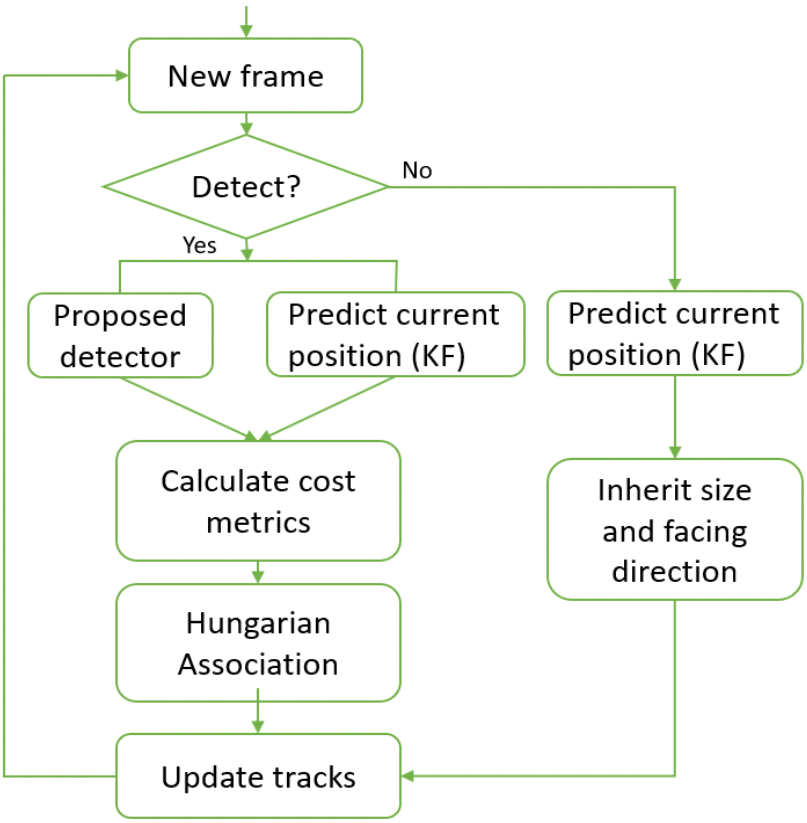
[Figure 2-3] Define pose orientation

CHAPTER 3 The Proposed Multiple Pedestrians Tracking method

3.1 Overall Working Architecture of the Proposed Multiple Pedestrians Tracking Method

Figure 3-1 describes the overall work-flow of our proposed multiple pedestrians tracking method. The proposed tracking method operates alternatively between detection mode and prediction mode. In detection mode, first detect pedestrians in the current frame and associate them with pedestrians in the previous frame. In the prediction mode, predicting is tracking. The proposed tracking method starts in the detection mode. In the detection mode, the pedestrians and their pose orientations are extracted as tight bounding boxes of pedestrians and as one of 4 status (front, back, left, right) by our developed CNN-based pedestrian detector. pose orientation of a pedestrian represents the direction where pedestrian looks towards in a scene. For tracking, association between pedestrians of the previous frame and those of current frame is accomplished by Hungarian algorithm where association cost metrics is made from IOU between the predicted bounding box and detected bound box of a pedestrian pose orientation. Association algorithm of the proposed tracking method is explained in detail later. The predicted bounding box of a pedestrian is obtained from translating the previous bounding box to a new position

by a motion vector, which is predicted by Kalman filter. Kalman filter contributes in improving object tracking performance as demonstrated in [16].



[Figure 3-1] The flowchart of the proposed multiple pedestrians tracking system

In the prediction mode, the tracking system applies Kalman filter to predict the motion vector of each bounding box towards the current frame. Motion vector indicates moving direction. If the system cannot predict the current positions, then it keeps the previous positions as the current positions. And, during prediction mode, size of bounding boxes and pose orientation are kept as the same as before. In the prediction mode, the predicted pedestrians are considered as tracked pedestrians.

The proposed tracking system keeps each pedestrian's information during consecutive 10 frames. After 10 frames, if the tracking system cannot find the corresponding updated locations, it considers the pedestrian is lost or disappear from the scene.

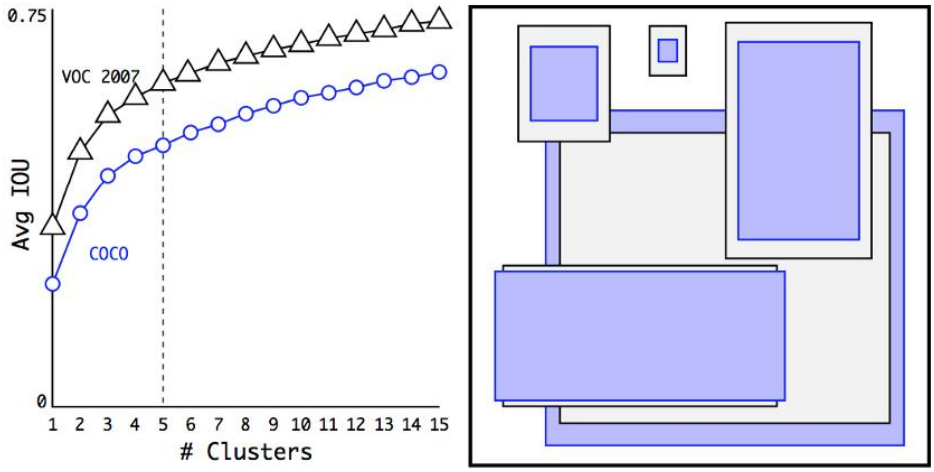
A newly detected pedestrian is kept to follow in next 3 frames. If the tracking system finds a match to it in that period, it is considered a new object and tracking of it starts. Our detector made error sometimes, if a newly detected person is instantly considered a new track.

[Figure 3-2] PTZ camera setup and detection result

3.2 Pedestrian detector

3.2.1 Anchor boxes

The first step in any object detection methods is to generate the potential bounding box candidates. These bounding boxes are vary and it depends on methods. Some hand-craft feature object detector like HOG or SIFT use a fixed size or some fix size bounding boxes and slide the fixed size window over the images to find the target objects. In modern Deep Neural Network object detector, the potential box candidates are generated in different ways. Fast-RCNN use a network called region proposal network to generate the box candidates from input images. This method works pretty well. However, it requires additional network to do it leads to drop in processing speed.



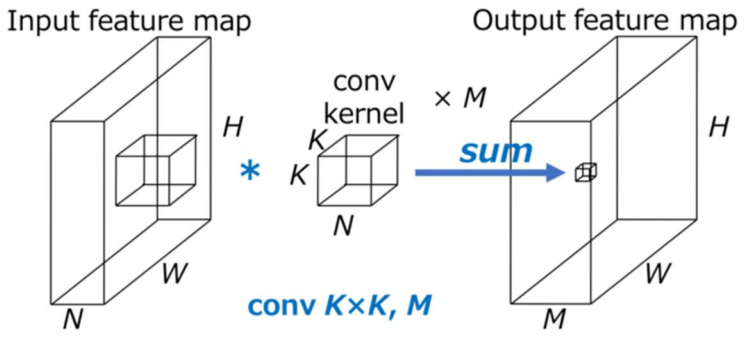
[Figure 3-3] Clustering box dimensions on VOC and COCO dataset. The left image shows the average IOU and the right image shows the relative centroids for VOC and COCO with 5 ratios. Source [9]

Some one-stage detectors like YOLO, SSD proposed the anchor concept. The sizes of potential box candidates are generated based on the training and testing dataset. In fact, the ration between width and height of an object are not vary too much, therefore, based on the dataset, I can choose a fixed ratio or a set of fixed ratios to represent the box candidates.

The popular and effective method to generate anchors box is using K-means clustering on the dataset. The number of ratios I need to train the network is equal to number of clusters in the K-means algorithm.

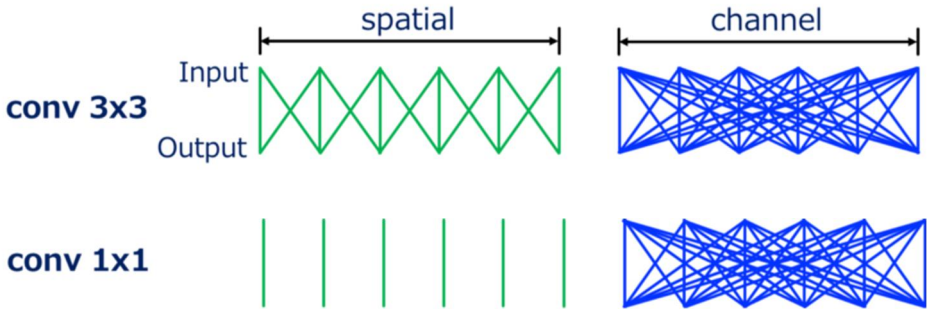
3.2.2 Depth-wise convolution layer

Consider an input features with shape WxHxN where WxH denotes the spatial size of the input feature map and N denotes the number of input channels. A convolution layer with size KxK is applied for this input feature map. After slide the window size of KxK over the input features map, the output feature map will have shape of SxHxM, where M denotes the output channels. The total computation cost is $WHNK^2M$. The computation cost can be divided into the spatial size of input and output feature map which is HxW, the size of convolution layer which is K^2 and the number of input and output channels NxM.



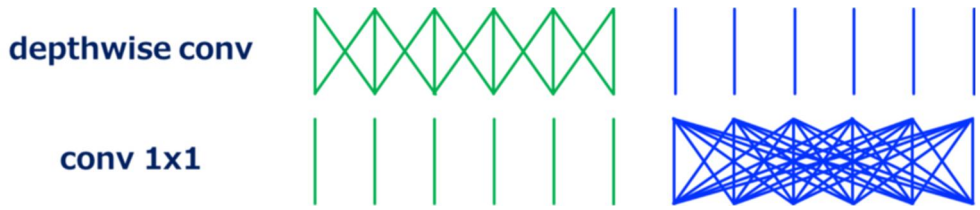
[Figure 3-4] Convolution operation on feature map [25].

The most popular convolution layer is a convolution layer with kernel size of 3x3. The computation cost of convolution layer 3x3 can be visualized in spatial and channel domain separately as in figure 3-5. In the spatial domain the input and output are locally connected, the number of connections in a node is equal to the size of the kernel. In the channel domain, it is fully connected.



[Figure 3-5] visualizing convolution 3x3 and convolution 1x1 in spatial and channel domain [25]

In depth-wise convolution layer. The computation is performed independently for each input channel. Clearly, it reduces computation cost as show in figure 3-6.



[Figure 3-6] Separable convolution layer in Mobilenet [25]

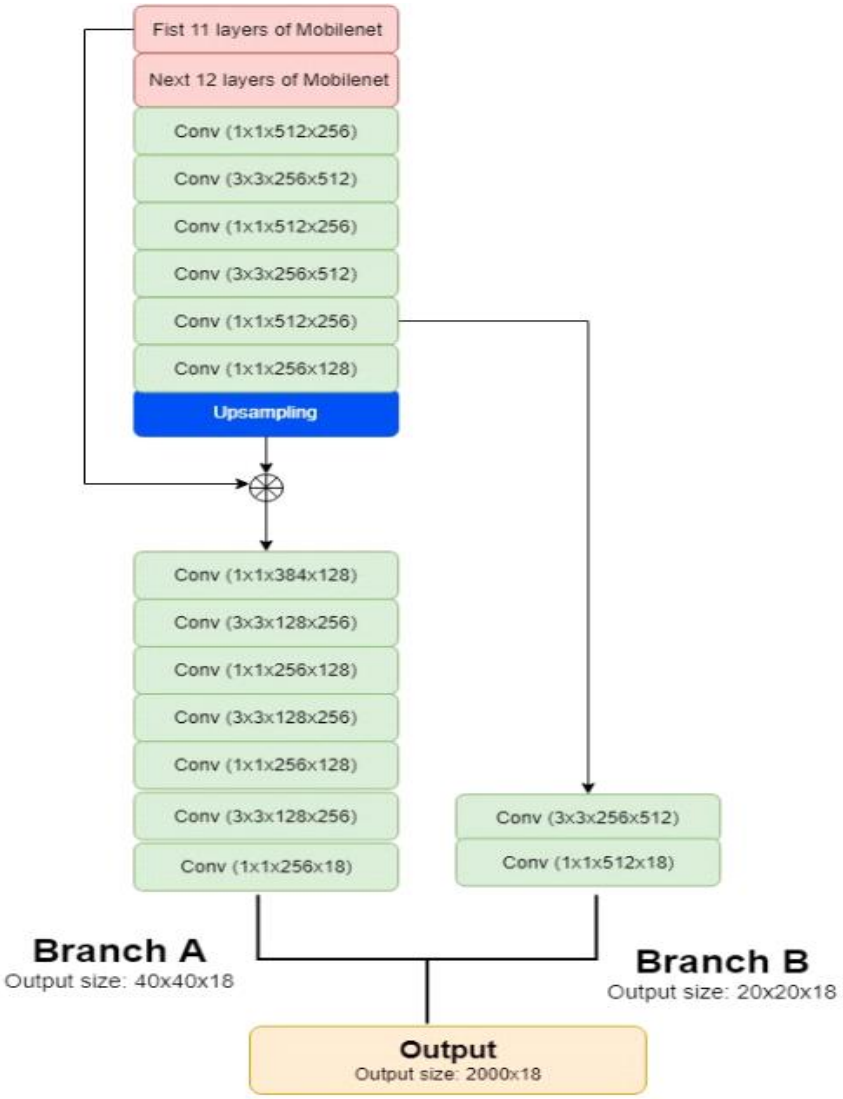
The 3x3 convolution layer is now replaced with a depth-wise convolution layer and a convolution layer 1x1 (separable convolution layer). From figure 3-5 and 3-6, after replacing 3x3 convolution layer by a depth-wise convolution layer and a 1x1 convolution layer, they are both calculating in the spatial and channel domain in the same way. The computation cost of the separable convolution layer is now $HWNK^2$ (depth-wise) + HWNM (1x1 convolution layer) = $HWN(K^2+M)$. Compare to $WHNK^2M$ of 3x3 convolution layer, the computation cost of separable convolution layer is smaller. Normally $M>>K^2$. For example K=3 and M>=32, the computation cost of separable convolution layer is 8 – 9 times smaller than the 3x3 convolution layer.

3.2.3 People detector network architecture

I design a CNN-based pedestrian detector which can detect small pedestrians as well as large pedestrians fast enough for real-time embedded applications. As a backbone of the proposed pedestrian detector network, I employ the first 23 layers of Mobilenet [17]. Mobilenet is well-known as a light weight deep neural network for mobile and embedded vision applications, which are based on a streamlined architecture that uses depth-wise separable convolutions.

I stack 5 more convolution layers to the top of the backbone to get more semantic information. Determination of the additional 5 convolution layers is done after optimality testing through experiments. After stacking 5 more convolution layers, I construct the rest of network into 2 branches; branch A responsible for detecting small size pedestrian and branch B responsible for detection of large size pedestrian. Moreover, inspired by Feature Pyramid Networks [18] that extract features from different layers of different scales, I design the detection network to use features from different layers for accurate detection.

For smaller size pedestrian detection, I need to detect at higher resolution, therefore, I use a convolution layer together with an upsampling layer to increase the resolution of the high-level structure and then I concatenate it with low-level structures features from 11th layer of backbone. I stack 7 more convolution layers to reduce the aliasing effect of upsampling and concatenating. Experiments show that 7 more convolution layers is optimal for branch A. In this branch our image features include low-level structures from 11th layer and high-level structure after upsampling layer. Even though, low-level structures that are not effective for accurate pedestrian detection, it keeps the features of small pedestrian. By combining both low-level and high-level structures features, the feature maps in this branch can describe small pedestrian features with rich semantic information. Therefore, branch A is effective for detecting small size pedestrian.



[Figure 3-7] Pedestrian Detector Network architecture

Figure 3-7 shows detail of our network architecture. The output of the Branch B for large size pedestrian detection is a matrix of size 20×20×18. 20x20 refers to the number of grid cells. Each cell contains 18 values, which divides into 2 bounding

boxes. Thus, each bounding box has 9 values consisting of 4 bounding coordinates, 4 pose orientations and 1 for confidence scores. Similarly, in the Branch A, the output size is 40×40×18. Branch A is used to detect small pedestrians, and therefore, I use higher resolution feature maps and increase the number of cells. The final output size is the combination of 2 branches with 2000 candidate bounding boxes (40×40+20×20=2000). Each bounding box is responsible to predict 18 values. Therefore, our final output size is 2000×18. Similarly to YOLOv3, I use the anchors to make predicting people. I use K-means clustering on our training set to find suitable anchor values for our dataset. The anchors I use in my experiments are (5×24), (9×50), (17×80), (32×155), (58×251), (114×402).

Figure 3-8 shows detection result of 2 branches on MOT16-12 sequence video, at frame 215. The result of branch A is on the right, it only detects the small pedestrian while branch B focus on detecting large pedestrian.

[Figure 3-8] Detection result of 2 branches on MOT16-12 sequence video, at frame 215.

3.2.4 Loss function

The loss function of the designed pedestrian detector network inherits from YOLO[14], The loss function penalizes four loss criteria. The detector network produces 2000 candidate bounding boxes. Each box is responsible for predicting 18 values, and then the detector network calculates loss for each cell. I obtain the final loss by summing up losses of every cells. For each box, the loss includes bounding box loss, pedestrian confidence loss, background loss and pose orientation loss.

Bounding box loss is same as YOLO as shown in 1st and 2nd term in figure 3-9. Human loss indicates the probability that a cell contains a person (3rd term in figure 3-9). Background loss is used to penalize when the background is detected as a person (4th term in figure 3-9. Where $(x_{ij}, y_{ij}, w_{ij}, h_{ij}, C_{ij})$ is the center location, width, height and confidence score of box j^{th} cell i^{th}, ($\widehat{x_{ij}}, \widehat{y_{ij}}, \widehat{w_{ij}}, \widehat{h_{ij}}, \widehat{C_{ij}}$) is the predicted center location, width, height and confidence score of corresponding cell. 1_{ij}^{obj} indicate human, 1_{ij}^{obj}, C_{ij} equal to 1 if the bounding box j^{th} of cell i^{th} contains a person and 0 in other case. 1_{ij}^{nobj} indicates background, $1_{ij}^{nobj} = 1$ only if the bounding box j^{th} of cell i^{th} doesn't contain a person.

I add one more term to penalize human pose orientation (facing direction), in this paper, I consider 4 main directions, facing to the left, right, front and back. Our pose orientation loss is expressed as shown in formula (1). Where, N is the number of directions, in this thesis I choose N=4. p_{ijk} is the ground truth probability

that a person in the bounding box j[th] of cell i[th] faced to k[th] direction. $\widehat{p_{ijk}}$ is the corresponding predicted probability.

$$\frac{1}{N}\sum_{k=1}^{N} 1_{ij}^{obj}[-(p_{ijk}\log(\widehat{p_{ijk}}) + (1-p_{ijk})\log(1-\widehat{p_{ijk}})] \quad (1)$$

My final loss combines all losses in 2000 cells as shown in Figure 3-9. Some constants α, λ, γ are weighting factors of each loss function. For example, if I want more precision on the location, I set a higher value for λ.

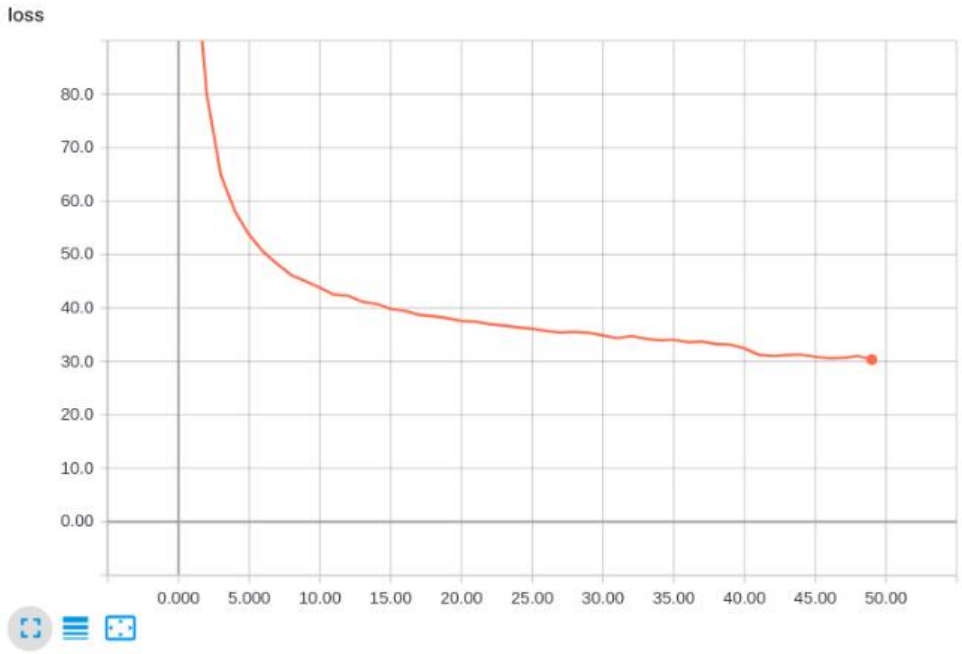
$$\begin{aligned}
Loss = {} & \lambda coord \sum_{i=1}^{2000}\sum_{j=1}^{B} 1_{ij}^{obj}[(x_i-\hat{x}_i)^2 + (y_i-\hat{y}_i)^2] \\
& + \lambda coord \sum_{i=1}^{2000}\sum_{j=1}^{B} 1_{ij}^{obj}[\left(\sqrt{w_i}-\sqrt{\widehat{w_i}}\right)^2 + \left(\sqrt{h_i}-\sqrt{\hat{h}}\right)^2] \\
& + \sum_{i=1}^{2000}\sum_{j=1}^{B} 1_{ij}^{obj}[-(C_i\log(\widehat{C_i}) + (1-C_i)\log(1-\widehat{C_i}))] \\
& + \alpha \sum_{i=1}^{2000}\sum_{j=1}^{B} 1_{ij}^{nobj}[-(C_i\log(\widehat{C_i}) + (1-C_i)\log(1-\widehat{C_i}))] \\
& + \gamma \sum_{i=1}^{2000}\sum_{j=1}^{B}\frac{1}{N}\sum_{k=1}^{N} 1_{ij}^{obj}[-(p_{ijk}\log(\widehat{p_{ijk}}) + (1-p_{ijk})\log(1-\widehat{p_{ijk}})]
\end{aligned}$$

[Figure 3-9] People detector loss function

where 1_{ij}^{obj} denotes if object appears in bounding box j[th] of cell i[th] and 1_{ij}^{nobj} denotes if object doesn't appear in bounding box j[th] of cell i[th].

3.2.5 Training process

I trained our network on our collected dataset for 50 epochs with Adam optimizer [24]. The dataset contains roundly 8000 images from a part of MOT16 [20] training set and from the internet. I need to relabel the dataset because there is no suitable dataset that provide pose orientation information. A PC with Intel™ Core™ i7-4770 CPU 3.40GHz, 16GB of RAM equipped with GeForce GTX Titan X Graphic Card is used to train the proposed pedestrian detector. The training process took roundly 1 day.



[Figure 3-10] Training loss after 50 epochs.

Since I have a limited dataset, in order to make the network learn better and avoid overfitting, I use an augmentation preprocessing step before feeding images to

the network. Augmentation helps the network learn from different aspects of the target objects by some transformation of the original images. Therefore, by using augmentation, I make sure that, I feed different images to the network in different epochs. In other words, my network doesn't see the same images in the training process.

3.2.6 Improving inference speed

During experiments, I realize that if I resize images by using CPU before feeding it into the model to do detection, the total running time is reduced about 5%-10%.

Fusing batch normalization with a convolution layer improves detection speed about 10-30%. Batch normalization together with dropout layer is popularly used to reduce overfitting and potentially improve the generalization of the model. Batch normalization is usually used after convolution layers. It also helps the training faster, make the network converge in a short amount of time. However, in inference step, the functionality of batch normalization is turn off and the approximated mean and variance are used instead. This can be implemented as a convolution layer or merged with the previous convolution layer.

3.3 Tracking Association

I propose an association algorithm that utilizes features of the association cost metrics of SORT [3] and Deep-SORT [4]. I first integrate IOU, pose orientation

all together to form a cost metrics. Then I apply Hungarian association method on that cost metrics. Figure 3-11 shows the integration.

Calculating Cost metric

Input: List of previous detected people $T = \{t_{1,}, t_{2,}, \dots t_N\}$

List of detected people: $D = \{d_{1,}, d_{2,}, \dots d_M\}$

Output: Cost metric $C = \{c_{i,j} | i = 1 \rightarrow N, j = 1 \rightarrow M\}$

Calculating:

For i **in** N:
  For j **in** M:
    $c_{i,j} = IOU(t_i, d_j)$
    If $c_{i,j} > 0$:
      If $t_i(fd) == d_j(fd)$:
        $c_{i,j} = min(1, c_{i,j} += 0.1)$
      Else:
        $c_{i,j} = max(0, c_{i,j} -= 0.1)$

Where IOU indicates the intersection over union.

$t_i(fd)$ and $d_j(fd)$ is pose orientation of predicted tracking object and pose orientation of detecting object

[Figure 3-11] Calculating cost metric.

The IOU between track t_i and detection d_j is calculated as bellow:

$$IOU(t_i, d_j) = \frac{Area(t_i) \cap Area(d_j)}{Area(t_i) \cup Area(d_j)} \quad (2)$$

Consider a frame with two people A and B, the cost metrics is $C = \begin{bmatrix} c_{A-1,A} & c_{A-1,B} \\ c_{B-1,A} & c_{B-1,B} \end{bmatrix}$, where $c_{A-1,B}$ is the matching weight between the predicted KF of person A and newly detected position of person B, $c_{A-1,A}$ is the matching weight between the predicted KF of person A and newly detected position of person A, similarly for $c_{B-1,A}, c_{B-1,B}$. Ideally, I expect the values of $c_{A-1,A}$ and $c_{B-1,B}$ as large as possible while the values of $c_{A-1,B}$ and $c_{B-1,A}$ go to zero. Most of the previously approaches use IOU distance as cost metric values. In this case, when person A and B are totally separated, the values of $c_{A-1,B}$ and $c_{B-1,A}$ are zero while the value of $c_{A-1,A}$ and $c_{B-1,B}$ is greater than zero. Therefore, Hungarian method correctly assign identities for person A and B. However, when A and B are overlapped, the values of $c_{A-1,B}$ and $c_{B-1,A}$ are greater than zero and sometimes it also greater than the values of $c_{A-1,A}$ and $c_{B-1,B}$, as a result, Hungarian method makes incorrect decision.

Besides IOU distance, appearance features are used in our cost metrics. When A and B is overlapped, I consider the pose orientation (facing direction) of newly detected position of A and B. Clearly, the facing direction of person A should be the same as the facing direction of the predicted KF of A. Therefore, if any predicted KF overlap and have the same pose orientation with newly detected people, the probability that those two candidates are matched is increase. I add up more confidence weight to the cost metric for this case. On the other hand, when predicted KF and newly detected people have different facing direction, the probability those two candidates are matched is low. I reduce the confidence weights by subtracting a

small value from the matching weight values. By modifying matching confidence weight from IOU distance metrics, I increase the accuracy of Hungarian method with less identity switches compare to the original IOU metrics.

CHAPTER 4 Experiments and Results

4.1 Experiment Environments

For the evaluation of the proposed pedestrian detector, I employ the well-known pedestrian dataset - INRIA person dataset [19] which includes roundly 1000 images. Furthermore, in order to evaluate the efficiency of the proposed pedestrian detector's detectability of small size pedestrians, I use MOT16 [20] pedestrian detection test set which includes 7 challenge videos with total 5919 image frames, 759 tracks and 182,326 target bounding boxes. Video frame rate vary from 14 frame per second to 30 frame per second, frame size is 640×480 (MOT16-06 sequence) and 1920×1080 (the remaining sequences). Pedestrian density varies from 8.1 to 45.3 pedestrian per frame.

For evaluation of the proposed multiple pedestrians tracking method, I utilize MOT16 benchmark [20]. And compare the proposed tracking method with other the state-of-the-art tracking methods, SORT[3], Deep-SORT[4], IOU-Tracker [5] which have been reported in MOT 2016 Benchmarks [20]. MOT16 evaluation dataset include 7 challenge videos (MOT16-01, MOT16-03, MOT16-06, MOT16-07, MOT16-08, MOT16-12, MOT-16-14). I compare proposed tracking method with other state-of-the-art tracking methods using proposed detector on 2D MOT 2015 benchmark[20]. The evaluation dataset includes 11 sequence videos (TUD-

Stadtmitte, TUD-Campus, PETS09-S2L1, ETH-Sunnyday, ADL-Rundle-6, ADL-Rundle-8, ETH-Bahnhof, ETH-Pedcross2, KITTI-13, KITTI-17, Venice-2) with total 5500 image frames, 39905 target bounding boxes. Video frame rate vary from 7 to 30 frames per second, frame size from 640×480 to 1920×1080. The pedestrian density varies from 2.2 to 11.9 pedestrian per frames.

I use a PC with Intel™ Core™ i7-4770, CPU 3.40GHz, 16GB of RAM equipped GeForce GTX Titan X Graphic Card to train the proposed pedestrian detector. The embedded system for experiments is a Jetson TX2 board [2] with HMP Dual Denver 2/2 MB L2, Quad ARM A57/2 MB L2, CPU 2GHz, 8GB of RAM. NVIDIA Pascal™, 256 CUDA cores.

4.2 Experimental Metrics

4.2.1 Experimental Metrics for People Detector

To evaluate pedestrian detection performance, I use Average Precision (AP), precision and recall.

Precision is used to measure how accuracy is the prediction. That is the percentage of the correctly detected positive objects. It is defined as below:

$$Precision = \frac{TP}{TP + FP}$$

Where TP (True Positive) is the total number of correctly detected targets. FP (False Positive) is the total number if incorrectly detected targets.

Recall is used to measure how good the system finds all the positives. It is defined as:

$$Recall = \frac{TP}{TP + FN}$$

Where FN (False Negative) is the total number of undetected targets.

AP (Average Precision) is computed as the average of maximum precision at 11 recall levels (it is divided by 11 to find the average):

$$\frac{1}{11} x \sum_{r = [0.0:1.0]} AP_r = \frac{1}{11} x \sum_{r = [0.0:1.0]} P_{interp}(r)$$

Where $P_{interp}(r) = max_{(s>r)}(P(s))$. For the more detail, reader may refer to [21].

4.2.2 Experimental Metrics for Tracking Association

For evaluating tracking performance, I use performance metrics suggested by MOT benchmark [20] that include MOTA, MOTP, MT, ML, FP, FN, ID SW, Frag. If the tracked objects are an actual target objects, then the tracked object is called true positive, if not, then the tracked objects are called false positive. If the actual targets are fail to track, then it is called a false negative. FP represents the total case of false positives, and FN represents the total case of false negatives. ID-SW (Identity Switches) counts the total number of mismatched objects in frames. If a tracked object doesn't match its actual ground truth target, it is an ID switched. MT (Mostly Tracked target) means the percentage of ground-truth trajectories that are covered by a tracking method for at least 80% of their respective life-span. ML (Mostly Lost targets) is the percentage of ground-truth trajectories that are covered

by a tracking method for at most 20% of their respective life-span. Frag is the total amount of times a trajectory is fragmented (that is interrupted during tracking), and speed is processing speed (in frames per second (fps) excluding the detector) on the benchmark.

MOTA (Multiple Object Tracking Accuracy) measure three error sources: missed targets, false positives and identity switches and is defined as:

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + ID\_SW_t)}{\sum_t (GT_t)}$$

where t is the frame index and GT is the total number of ground truth objects.

MOTP (Multiple Object Tracking Precision) is used to measure the misalignment between the annotated (ground truth) and the tracked target bounding boxes and is defined as:

$$MOTP = \frac{\sum_{t,i} (d_{t,i})}{\sum_t (C_t)}$$

where C_t refers to the number of matches at frame time t and $d_{t,i}$ is the IOU ratio (3) between tracked object and its ground truth bounding boxes at frame time t. MOTP is used to evaluate the average overlap between all correctly matched targets and their respective objects and it is ranges between t_d = 50% and 100%.

For the detail description of each metric, readers may refer to [6, 20]. For object tracking point of view, MOTA is considered to be more important than the MOTP.

4.3 Experimental Results

4.3.1 Performance of Pedestrian detector

Experiment results about performance of the proposed pedestrian (object) detector against INRIA and MOT16 dataset are summarized in Table 4-1.

[Table 4-1] Performance comparison between state-of-the-art pedestrian detector and proposed pedestrian detector

	AP against INRIA	AP against MOT16	MOT16 Precision	MOT16 Recall	Speed (FPS) (On Jetson TX2)
Proposed Pedestrian Detector	0.66	0.49	55.5	56.4	12.5 FPS
Tiny YOLOv3	0.51	0.27	32.9	35.4	14 FPS
YOLOv3 JTA [20]	-	0.62	84.7	66.2	1 FPS
YOLOV2[20]	-	0.46	84.3	69.9	3 FPS
Faster R-CNN[20]	-	0.72	89.8	77.3	10 times slower than YOLOv2
DPM[20]	-	0.61	64.8	68.1	-

(*) In [9], the author claimed that YOLOv2 is about 10 times faster than Faster R-CNN.

Where YOLOv3 JTA is the YOLOv3 trained on JTA dataset, YOLOv2 is the version 2 of YOLO, and DPM is Deformable Part Model [23]. The results for the proposed

one and Tiny YOLOv3 in Table 1 have been obtained from experiments on Jetson TX2 board and AP of YOLO JTA, YOLOv2, Faster R-CNN, and DPM against MOT16 are taken from MOT website [20], but speed is measured on Jetson TX2 board.

[Figure 4-1] Example result on MOT16-03 sequence #frame 158

Table 4-1 shows that the proposed pedestrian detector performs significantly better than Tiny YOLOv3 against MOT16 testing set, which contain many small size pedestrians. YOLOv3 performs more reliable in pedestrian detection over the proposed one. However, YOLOv3 is very slow on an embedded computing device, Jetson TX2 board. It takes around 1 second to process a frame. For the proposed tracking method, I apply the detector every other frame, which makes the entire processing of the proposed tracking method operate in real-time even for embedded systems.

4.3.2 Performance of The Proposed Multiple Pedestrian Tracker

Experimental results on performance of the proposed tracker against MOT16 benchmark are summarized in Table 4-2. I compare my result with some open source state-of-the-art tracking algorithms such as SORT[3], Deep-SORT[4]. and IOU Tracker[5], and our previous tracker, LKDeep [6], SORT[3], Deep-SORT[4], and IOU tracker[5] are not the best among the stat-of-art tracking algorithms, but are chosen since they released open source codes.

The results of LKDeep in Table 4-2 is taken from [6]. The results of other open source trackers in Table 2 is taken from [20], and the arrow ↑ indicates the higher score is better, and arrow ↓ indicates the lower score is better. IOU tracker, SORT, and Deep-SORT utilizes Faster R-CNN detection data provided by MOT16 challenge, which produces more precise detection than single shot detector like the

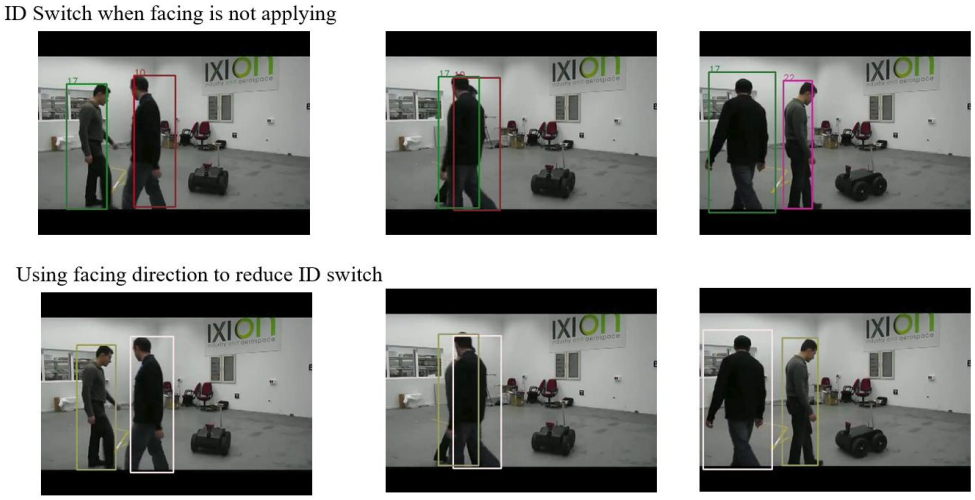
proposed detector, but is slow. LKDeep, which was our previous tracker, utilizes DPM detection data provided by MOT16. And, Proposed Tracker v2 is the same as proposed tracker except that pedestrian detector is applied for every frame.

[Table 4-2] Comparing tracking results on MOT16 challenge Benchmarks

	MOTA↑	MOTP↑	MT↑	ML↓	FP↓	FN↓	ID SW↓	Frag↓	Association speed ↑	Computing Environment
Proposed Tracker(**)	41.58	74.0	10.7	42.4	7569	97738	1345	1566	148	Jetson TX2
Proposed Tracker v2(*)	42.3	74.2	11.7	41.2	7590	96425	1207	2533	84.6	Jetson TX2
SORT[3] (*)	59.8	79.6	25.4	22.7	8698	63245	1423	1835	59.5	2.6 GHz, 1 Core
Deep-SORT [4](*)	61.4	79.1	32.8	18.2	12852	56668	781	642	17.4	2.6 GHz, 1 Core
IOU Tracker [5](*)	57.1	77.1	23.6	32.9	5702	70278	2167	1839	3004	3.4 GHz, 1 Core
LKDeep[6](*)	32.3	76.4	5.7	62.1	1193	121333	953	943	32	3.4GHz, CPU

(*) indicates that a detector in the tracker is applied for every frame, and (**) indicates that a detector is applied for every other frame.

From experimental results in Table 4-2, one can notice the following facts. First, the proposed tracker achieves comparable performance overall compared to our previous tracker, but with significant association speed-up, which makes the proposed tracker more suitable for embedded application. Second, the proposed tracker shows much better association speed compared to other open source trackers except IOU tracker. Even though IOU tracker shows very high association speed due to very simple association algorithm, it cannot work in real-time since it utilizes Faster R-CNN detector which is slow as shown in Table 4-1.



[Figure 4-2] Using facing direction to reduce identity switch problem

In order to have a fair comparison, I evaluated the open source tracking methods, SORT tracker, Deep-SORT tracker and IOU tracker by replacing their detectors by our designed pedestrian detector against 2D MOT 2015 benchmark. Table 4-3 summarizes the experimental results from replacement of our pedestrian

detector, which are produced by Multiple Object Tracking Challenge Development Kit [22].

[Table 4-3] Comparing tracking results on the same detector on 2D MOT 2015 Benchmark.

	MOTA↑	MOTP↑	MT↑	ML↓	FP↓	FN↓	ID SW↓	Frag↓	FPS↑
Proposed Tracker (**)	37.1	73.8	129	163	8044	16575	496	1018	16.7
Proposed Tracker v2 (*)	39.1	74.7	134	152	7588	16268	431	1000	11.4
SORT (*)	38.2	74.4	111	156	7043	17077	528	847	11.4
SORT v2 (**)	36.4	73.9	109	182	7200	17608	576	869	16.7
Deep-SORT(*)	38.1	74.4	165	139	9342	15065	285	851	6.7
IOU Tracker(*)	36.1	74.6	174	122	9588	14776	1117	1130	12.4

((*) Detector is applied for every frame, (**) Detector is applied for every other frame.)

As stated in 4.2, from tracking point of view, MOTA (Multiple Object Tracking Accuracy) is the most important performance metrics to evaluate a tracking method. Proposed tracker v2 where pedestrian detector is applied for every frame produces higher MOTA and MOTP compared to other trackers. Table 4-3 shows that by incorporating pose orientation into association cost metrics, the proposed tracker has less ID SW (Identity Switches) of 431 compared to 528 of SORT while keeping almost the same processing time. Our association also can keep tracking the

target for longer period with the number MT is 129 compared to 111 of SORT. IOU tracker do a simple and less accuracy association, so that leads to a lot of ID SW (Identity Switches) problem with 1117 identity switches. With high quality deep appearance features more for association, Deep-SORT is the best at keep tracking target, with ID SW of 285. It also keeps tracking the target for long period with the highest MT 165. However, extracting deep appearance features takes a lot of time which makes Deep-SORT becomes the slowest tracker in Table 4-3.

The proposed tracker reduces processing time by applying pedestrian detection every other frame. From Table 4-3, one can see that by applying detection for every other frame I made a big improvement with respect to processing speed without sacrificing too much tracking accuracy. The proposed tracker is the fastest among trackers in Table 4-3. I also tested SORT v2 where our designed detector is applied for every other frame. Accuracy performance of SORT v2 deteriorates even though speed performance improves. Overall, the proposed tracker can be considered as more suitable than other trackers listed in Table 4-3 for embedded applications such as multiple pedestrians tracking under PTZ camera.

CHAPTER 5 Conclusions

In this thesis, I proposed a real-time multiple pedestrians tracking method for embedded applications such as embedded surveillance. Compared to the state-of-the-art multiple objects tracking methods, which show excellent tracking accuracy performance, the proposed tracker traded off accuracy performance against speed performance, but operates in real-time and performs accurately good enough for some embedded applications, which is shown through comparison experiments on Jetson TX2 embedded board.

Through experiments on Nvidia's embedded computing board, Jetson TX2 and comparisons to performance results of the sate-of-the-art object detectors and trackers, it is shown that the designed pedestrian detector detects fast and performs well even for small size pedestrian detection compared to many state-of-the-art object detectors, and that the proposed tracking method can operate in real-time for embedded systems like PTZ camera equipped with Jetson TX2 as a main processor and perform comparably to performances of many state-of-the-art tracking methods.

REFERENCES

[1] H. Yanga, L. Shaoa, F. Zhenga, L. Wangd, and Z. Songa, “Recent Advances and Trends in Visual Tracking: A Review,” Journal of Neu rocomputing, Vol. 74, No. 18, pp. 3823-3831, 2011.

[2] Jetson TX2, https://elinux.org/Jetson_TX2 (accessed Feb 12, 2019).

[3] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple Online and Realtime Tracking,” Proceeding of IEEE International Conference on Image Processing, pp. 3464-3468, 2016.

[4] N. Wojke, A. Bewley, and D. Paulus, “Simple Online and Realtime Tracking with a Deep Association Metric,” Proceeding of IEEE International Conference on Image Processing, pp. 3645-3649, 2017.

[5] E. Bochinski, V. Eiselein, and T. Sikora. “High- Speed Tracking-by-Detection Without Using Image Information,” Proceeding of International Workshop on Traffic and Street Surveillance for Safety and Security at IEEE Advanced Video and Signal-based Sureillance, pp. 1-8, 2017.

[6] Q.D. Vu, T.B. Nguyen, and S.T. Chung, “Simple Online Multiple Pedestrian Tracking Based on LK Feature Tracker and Detection for Embedded Surveillance,” Journal of Korea Multimedia Society, Vol. 20, No. 6, pp. 893-910, 2017.

[7] Hungarian Algorithm, https://en.wikipedia.org/ wiki/Hungarian_algorithm (accessed Feb., 12, 2019).

[8] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks,” Proceedings of the 28th International Conference on Neural Information Processing Systems - Vol. 1, pp. 91-99, 2015.

[9] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” Proceeding of 2017 IEEE Conference on Computer Vision and Pattern Recognition, pp. 6517-6525, 2017.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.Y. Fu, et al., “SSD: Single Shot MultiBox Detector,” Proceeding of European Conference on Computer Vision, pp. 1-17, 2016.

[11] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” arXiv:1804.02767, 2018.

[12] S. Bell, C.L. Zitnick, K. Bala, and R. Girshick. “Inside-outside Net: Detecting Objects in Context with Skip Pooling and Recurrent Neural Networks,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2874-2883, 2016.

[13] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, S. Yan, et al., “Perceptual Generative Adversarial Networks for Small Object Detection,” Proceeding of 2017 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1222-1230, 2017.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once Unified, Real- time Object Detection,” Proceeding of 2016 IEEE Conference on Computer Vision and Pattern Recognition, pp. 779-788, 2016.

[15] L. Leal-Taix, C.C. Ferrer, and K. Schindler, “Learning by Tracking: Siamese CNN for Robust Target Association,” Proceeding of 2016 IEEE Computer Vision and Pattern Recognition Conference Workshops, pp. 418-425, 2016.

[16] D. Kim, J. Park, and C. Lee “Object-tracking System Using Combination of CAMshift and Kalman Filter Algorithm,” Journal of Korea Multimedia Society, Vol. 16, No. 5, pp. 619-628, 2013.

[17] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, et al., “Mobile Nets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” arXiv:1704.04861, 2017.

[18] T.Y Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, S. Belongie, et al., “Feature Pyramid Networks for Object Detection,” Proceeding of 2017 IEEE Conference on Computer Vision and Pattern Recognition, pp. 936-944, 2017.

[19] INRIA Person Dataset, http://pascal.inrialpes. fr/data/pedestrian/ (accessed May 22, 2019).

[20] MOT Challenge, https://motchallenge.net/ (accessed May 22, 2019).

[21] M. Everingham, L. Gool, C.K. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes (VOC) Challenge,” Journal of Computer Vision. Vol. 88, Issue 2, pp. 303-338, 2010.

[22] Multiple Object Tracking Challenge Development Kit, https://bitbucket.org/amilan/motchallenge-devkit/ (accessed May 22, 2019).

[23] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, “Object Detection with Discriminatively Trained Part Based Models,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 9, pp. 1627-1645, 2010.

[24] Diederik P. Kingma, Jimmy Ba, "Adam: A Method for Stochastic Optimization", 3rd International Conference for Learning Representations, https://arxiv.org/abs/1412.6980, 2015.

[25] Why MobileNet and Its Variants (e.g. ShuffleNet) Are Fast, https://medium.com/@yu4u/why-mobilenet-and-its-variants-e-g-shufflenet-are-fast-1c7048b9618d (accessed May 22, 2019).