Real-time Online Human Tracking with a Stereo Camera for Person-Following Robots

Bao Xin Chen

A Thesis submitted to

the Faculty of Graduate Studies

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical Engineering and Computer Science

York University

Toronto, Ontario

September, 2019

Abstract

Person-Following Robots have been studied for multiple decades now. Recently, person-following robots have relied on various sensors (e.g., radar, infrared, laser, ultrasonic, etc). However, these technologies lack the use of the most reliable information from visible colors (visible light cameras) for high-level perception; therefore, many of them are not stable when the robot is placed under complex environments (e.g., crowded scenes, occlusion, target disappearance, etc.). In this thesis, we are presenting three different approaches to track a human target for person-following robots in challenging situations (e.g., partial and full occlusions, appearance changes, pose changes, illumination changes, or distractor wearing the similar clothes, etc.) with a stereo depth camera. The newest tracker (SiamMDH, a Siamese convolutional neural network based tracker with temporary appearance model) implemented in this work achieves 98.92% accuracy with location error threshold 50 pixels and 92.94% success rate with IoU threshold 0.5 on our extensive

person-following dataset.

All human participants based studies were sanctioned by the Office of Research Ethics at York University (Ethics Approval Certificate Id: STU 2017 - 065).

Acknowledgments

I acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC), the NSERC Strategic Network for Field Robotics (NCFRN), and the Canada Research Chairs Program through grants to my supervisor Professor John K. Tsotsos. I would like the thank the Graduate Program funding and scholarship from York University. I am truly grateful to my committee members, Prof. Michael Brown and Prof. George Z.H Zhu for reviewing my thesis.

I would like to thank Raghavender Sahdev and his brother Sidharth Sahdev for helping in the process of dataset preparation and making the videos for this work. And, Raghavender is the co-author of my two papers on person-following robots. And, I also would like to thank Dekun Wu provided hardware for testing.

Finally and most importantly, I would like to thank my parents and my sisters' unconditional support.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI	artificial intelligent
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRV	Conference on Computer and Robot Vision
fps	frames per second
GPU	Graphical Processing Unit
HSV	hue-saturation-value
ICVS	International Conference on Computer Vision Systems
NCC	Normalized Cross Correlation
OAB	Online Ada-Boosting Tracker
RGB	image with Red, Green, and Blue channels
RGBD	image with Red, Green, Blue, and Depth channels

RGBSD	image with Red, Green, Blue, and Stereo Depth channels
RPN	Region Proposal Network
SD	Stereo Depth channel
SiamMask	Siamese Mask tracker
SiamMD	Siamese Mask tracker with Depth
SiamMDH	Siamese Mask tracker with Depth and Appearance
SLAM	Simultaneous localization and mapping
SOAB	Selected Online Ada-Boosting
VGA	Video Graphics Array

Chapter 1

Introduction

1.1 Motivation

There has been tremendous progress in computer vision leading to many useful practical applications, such as Object detection [1] [2], Object tracking [3] [4], Image classification [5] [6], Image Segmentation [2] [7], Body Pose Estimation [8] [9], Style Transfer [10] [11], etc. As well, there are many robotic applications using computer vision technique, including Simultaneous localization and mapping (SLAM) using visual feature matching [12] [13], Person Following Robots using object tracking [14] [15], Fallen Person Detection using human pose estimation [16] [17], Place Recognition using visual features [18] [19], Place Classification for cognitive robots [20], Visual Search [21] [22], Visual Odometry [23], [24], Free Space Estimation [25], etc. But, there are still many open problems in computer vision.

Among these, Human Tracking is one of them, and drew my interest when We were developing an artificial intelligent (AI) program for a person following robot. There are two fundamental tasks in a person-following robot system ([14], [15]): 1) Tracking the target. 2) Following the target.

Person following robots need a robust and real-time algorithm to solve the tracking problem in a complex environment under unexpected circumstances. For example, the tracking target might be occluded by other instances, the lighting condition in the scene might change rapidly, and the target might change its pose dramatically (e.g., squat down and pick up something from the floor or removing a bag from the person (see Figure 1.1)). As a result, a reliable human tracking algorithm is a significant component of the person-following robots.

1.2 Human tracking task description

In this subsection, We will describe the task in terms of input, output, and major evaluation metrics.

Input. Assume that a sequence of images from a video is given. In most cases, the video has 30 frames per second. A human target with a rectangular bounding box is given in the first frame. The bounding box is defined as the top left corner and bottom right corner in pixel coordinates in the image (See

Figure 1.1: Different cases that a person following tracking algorithm should handle. (a) picking a bag. (b) wearing a bag. (c) sitting. (d) squatting. (e) illumination. (f) side facing. (g) partial occlusion. (h) complete occlusion. (i) standing side-by-side with the same clothes. (j) front crossing with the same clothes. (k)-(l) appearance changed.

Figure 1.2).

Output. On each of the subsequent frames, a bounding box should be produced upon execution of the tracking algorithm (See Figure 1.3). There are two types of tracking algorithms. One is online tracking. The algorithm outputs the bounding box frame by frame when it is running. The other type is offline tracking, where the algorithm can see the whole video sequence before predicting the bounding box. In this offline model, many have usually used a dynamic programming algorithm to find the global optimal target motion trajectory. In this thesis, We are focusing on an online model, since most of the robotic applications require real-time (roughly at least 15fps) tracking.

1.2.1 Evaluation metrics

In the object tracking stream, different datasets serve a different purpose. In general, there are two major quantitative metrics we need to focus on, which are bounding box position accuracy and bounding box size accuracy. To evaluate these, we use the method called the overlapping area between the ground truth bounding box (r_o) and the predicted bounding box (r_t) over the union of them, called *intersection over union* [3] (IoU), denote as $IoU = \frac{|r_t \cap r_o|}{|r_t \cup r_o|}$. Here, we use the most common metrics called ***success plots*** (See Figure 1.4). In *success plots*, ***x-axis*** is the overlap threshold from 0 to 1.0, where 0 means there is no overlapping area between the ground truth bounding box and the

(a) frame 0 **(b)** frame 14 **(c)** frame 29 **(d)** frame 44 **(e)** frame 59 **(f)** frame 74

Figure 1.2: Input video sequence with a given bounding box (in red colour) on the first frame. The images showing here are selected on every other 15 frames.

(a) frame 0 **(b)** frame 14 **(c)** frame 29 **(d)** frame 44 **(e)** frame 59 **(f)** frame 74

Figure 1.3: Output video sequence with predicted bounding boxes on the testing images. The images showing here are selected on every other 15 frames.

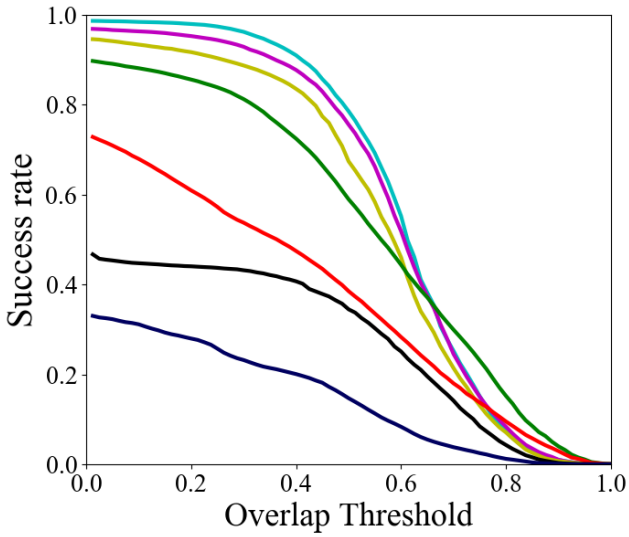


Figure 1.4: An example of Success Plot.

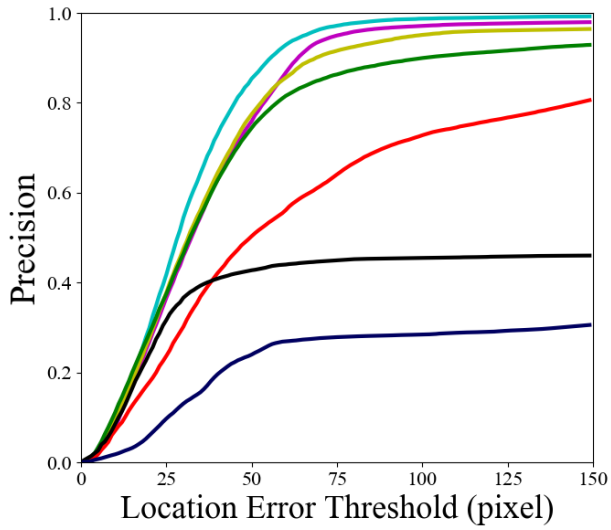


Figure 1.5: An example of Precision Plot. (Primary evaluation metric)

predicted bounding box, and 1.0 means two bounding boxes are identical in one frame. ***y-axis*** is the ratio between the number of success frames (a success frame is defined as its IoU is greater than the overlap threshold) and the total number of frames. Based on the success plot, choosing a success rate at threshold $t = 0.5$ and the area under the curve are the two popular quantitative evaluations for a general object tracking algorithm.

There is another metric called ***precision plots*** [3] (See Figure 1.5), which evaluates on the center location of the bounding boxes. In *precision plots*, ***x-axis*** is the center location error threshold, where center location error is measuring the center distance between ground truth bounding box and the predicted bounding box. ***y-axis*** is the ratio between the number of success frames (a success frame is defined as its center location error is less than the threshold) and the total number of frames. In precision plots, the quantity evaluation is also based on the area under the curve, and precision at location error threshold 50 pixels.

For the purpose of person following, if a tracking algorithm always provides the target center location, then the robot would not lose the target being tracked. So, we consider the precision plot as our primary evaluation metric.

1.3 Background

In this subsection, we will briefly survey the literature on human tracking for person-following robots in the past twenty years. This subsection is separated into two parts: Human Tracking Algorithm and the Hardware Used to Determine the Depth.

Human Tracking Algorithm: In 1999, Ku et al. [26] attached a rectangular shape to the back of the person as the interest region with a particular color. Their method could solve the simple detection problem, but it did not provide any robustness. In 1998, Piaggio et al. [27] started using optical flows for a person following robot. Similar work was done in [28] and [29] as well. However, optical flow has the restriction that the person and background must have different motions which is not always the case. In 2003, Beymer et al. [30] used wheel odometry to subtract background motion and estimate the person location. However this only works well on a uniform surface. In 2003, Tarokh et al. [31] used colour and shape of the person's clothes as features for detection. Although their method improved the robustness over Ku et al. [26], they did not consider situations when the target changes his/her appearance heavily. In 2006, Yoshimi et al. [32] used feature points (edges or corner points) detection and combined the pre-registered color and texture of the clothes. This method provided good robustness when the person is

making a turn or walking in upright poses. In 2007, Calisi et al. [33] used a pre-trained appearance model to detect and track the person. Their method could provide a good tracking result if they trained the model well enough with a lot of data. However, dynamic environments are unpredictable, and the target might change its appearance from time to time. Similarly, in 2007, Chen et al. [34] used sparse Lucas-Kanade features to track the target. But the features could be lost if the person is turning, or changing appearance. Again in 2007, Takemura et al. [35] used H-S Histogram in hue-saturation-value (HSV) color space, where HSV is robust to illumination since V (lightness) can be considered separately. In 2009, Satake et al. [36] used depth templates and SVM to train a human upper body classifier to track the person. However, this method did not handle cases, such as crossing, partial occlusion, etc. In 2010, Tarokh et al. [37] used HSV and controlled the light exposure to handle light variations. An update was made in 2014 to improve the following speed [38]. Some other fundamental feature tracking algorithms were also used in later literature, e.g., SIFT feature based [39] in 2012, HOG feature based [40] in 2013 and [41] in 2014, etc. In the recent work (2016), Koide et al. [42] applied height and gait with appearance features for person tracking and identification, but height and gait are only limited to the target walking in an upright position. The method is not robust when the target changes its

clothes or puts on a backpack ([43] also has this problem).

Hardware Used to Determine the Depth: In this thesis, We use depth to assist the tracking model for improving the reliability. Yoon et al. [44] gained aid from depth information to improve the computational speed and accuracy. Depth could also help with background and foreground issues by eliminating the sudden depth changing pixels, e.g., occlusion. Doisy et al. [45] used the Kinect camera and a laser sensor to propose an algorithm which solves the person depth information for person following. Bajracharya et al. [46] used depth from a stereo camera for detecting and tracking pedestrians in outdoor environments.

Nowadays, there are many different types of depth sensors in the market. In the recent publications, researchers prefer RGBD cameras, e.g., Kinect [47] and ASUS xTion [43] [44]. These cameras provide very good depth information only if the robot is running indoor without strong sunlight. Our approach uses a visible light stereo camera (e.g., Point Grey Bumblebee [1], ZED Camera [2]) which can be used both indoors and outdoors. Laser sensors provide another approach to detect depth [40] and [41]. But a laser sensor is expensive and often not permitted in places like hospitals, universities, malls and other similar places. To obtain the depth information of each pixel in an

[1] `https://autonomoustuff.com/product/flir-bumblebee-stereo-vision-camera-systems/`

[2] `https://www.stereolabs.com`

image, we use a stereo image based algorithm to compute the depth [48].

Briefly, the past literature had shown some working solutions, but they are neither sufficiently robust to track a complicated human target (with different poses or different clothes) nor to track a target in different environments (indoor and outdoor). In this thesis, we demonstrate three approaches to solve the problems with a stereo camera only.

1.4 Contributions

The purpose of this thesis is to build an efficient and reliable human tracking algorithm for person-following robots under challenging situations. This technology can be found in a wide range of robotic applications; such as, autonomous shopping carts, personal guides in public facilities, autonomous suitcases for travelers, etc.

In this thesis, We propose three different human tracking algorithms for different hardware setups. For the computing system without a dedicated GPU, We propose an online ada-boosting approach for human tracking. And, We also propose two convolutional neural network based tracking algorithms run on a system with a dedicated mobile GPU. We tested these algorithms on a dataset that was captured in real world environment and showed that these algorithms are able to track a human target under challenging situations.

Parts of the thesis have been published in the following conference papers:

- B.X. Chen*, R. Sahdev*, and J.K. Tsotsos, ”Person Following Robot using Selected online ada-boosting with stereo camera”, in 14th Conference on Computer and Robot Vision (CRV), pp 48-55, IEEE, 2017. [15] (Best Robotics Paper Award.) (partially presented in Chapter 2)

- B.X. Chen*, R. Sahdev*, and J.K. Tsotsos, ”Integrating Stereo vision with a CNN tracker for a person-following robot”, in International Conference on Computer Vision Systems (ICVS), pp. 300-313, Springer, 2017. [14] (Finalist for the Best Conference paper award.) (partially presented in Chapter 3)

- B.X. Chen, and J.K. Tsotsos, ”Fast Visual Object Tracking using Ellipse Fitting for Rotated Bounding Boxes”, in Proceedings of the IEEE International Conference on Computer Vision Workshops, 2019. [49] (partially presented in Chapter 4)

* Denotes equal contribution

1.5 Thesis outline

The thesis contains five chapters.

- **Chapter 1** introduces the motivation of this thesis, and describes the human tracking task definition and the background knowledge of human tracking algorithms.

- **Chapter 2** describes a modification of the existing Online AdaBoosting Tracker (OAB), so that the new tracker can track a target in a 3D environment.

- **Chapter 3** illustrates a deep learning tracker that performs real-time online tracking with model updating.

- **Chapter 4** presents the state-of-the-art pre-trained deep learning tracker that achieves 37fps real-time online known object tracking on our person-following dataset.

- **Chapter 5** concludes the thesis with showing the significance of this work and providing the potential future work on the thesis topic.

Chapter 2

A Selected Online Ada-Boosting for Human Tracking using Stereo Camera

2.1 Introduction

In this chapter, We present a Selected Online Ada-Boosting (SOAB) approach, a modified Online Ada-Boosting (OAB) tracking algorithm with integrated scene depth information obtained from a stereo camera which runs in real-time on a mobile robot. We build and share our results on the performance of our technique on a new stereo dataset for the task of person-following. The dataset covers different challenging situations like squatting,

partial and complete occlusion of the target being tracked, people wearing similar clothes, appearance changes, walking facing the front and back side of the person to the robot, and normal walking.

2.2 Related works

Boosting algorithms have been used in many areas in machine learning and computer vision ([50], [51], [52], [53]). Boosting usually trains with offline datasets. The Online Ada-Boosting algorithm for tracking an object in real-time has been described by Grabner et al. in [54] and [55]. To achieve real-time tracking, Grabner et al. used Haar wavelet features to improve robustness when appearance changes gradually, which was described by Wang et al. in [56].

In OAB, the tracking target is assumed to be given in the first frame (selected by human or detected by an off-line detection algorithm). The selected patch is used as a positive example to train the classifier. Then random patches are extracted from four regions (upper right, upper left, bottom right, bottom left, see Figure 2.1(f)) in the search area as negative examples. These random patches contain negative features, e.g., windows, wall, furniture, etc. An initial classifier is trained from these positive and negative patches. In the second frame, the target is detected using the classifier. The patch in the search re-

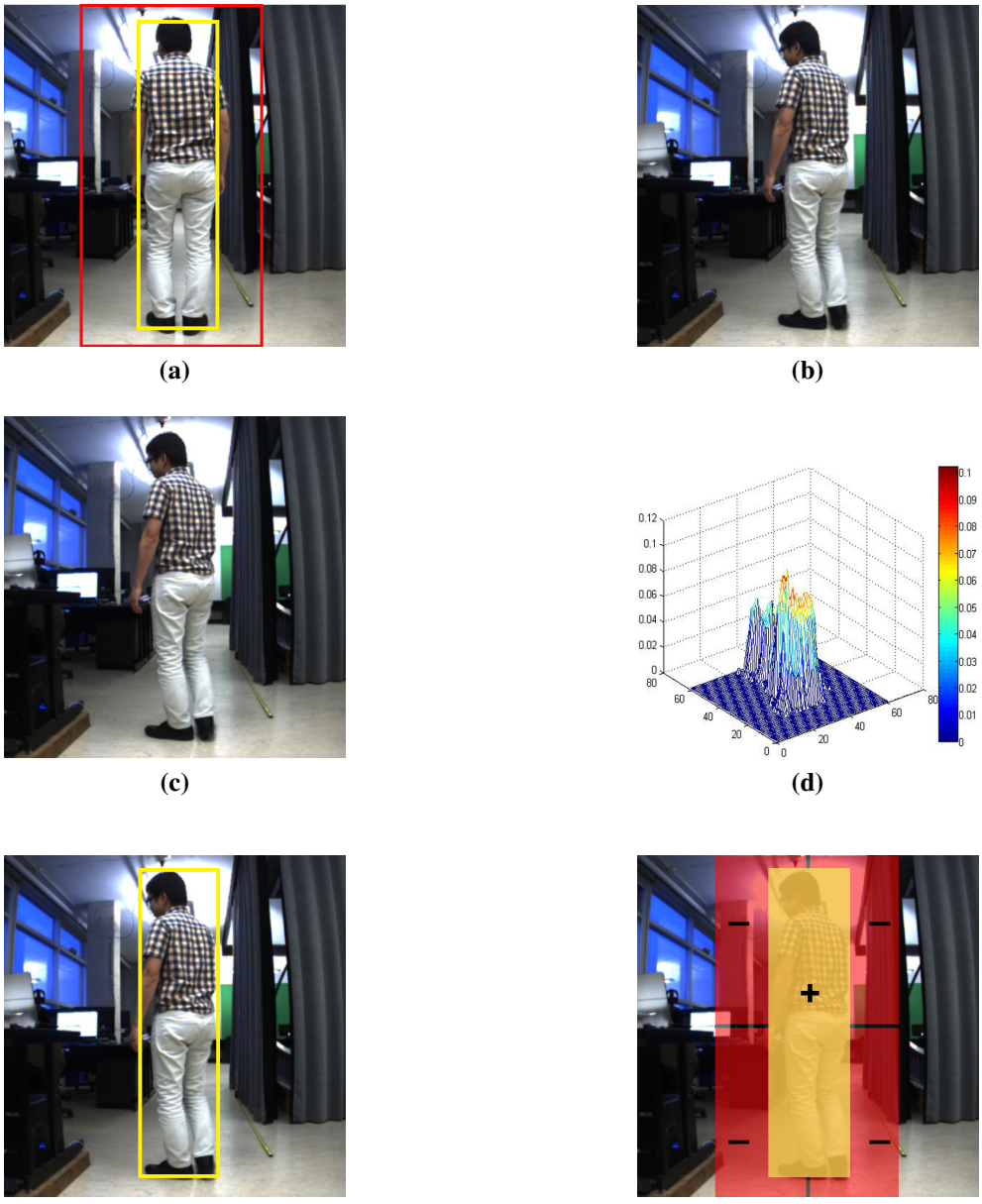


Figure 2.1: OAB updating process: (a) yellow box is the target region, the red box is the search region. (b) is the next frame. (c) is searching and evaluating the patches in the search region. (d) is the confidence map of the evaluation. (e) is the best matching with minimum error. (f) update the classifier with positive and negative patches. After (f) then go back to (a) to search in the next frame. Similar to [54], [55]

gion with minimum error is the best responding example. This patch is used as a positive example and the surrounding random patches from the four regions as negative examples to update the classifier. The steps performed on the second frame are continued on the subsequent frames (see Figure 2.1).

In order to achieve real-time boosting, OAB does not use all weak classifiers to calculate a strong classifier [55]. Instead, it selects N weak classifiers from all M global weak classifiers. In the following equations, H^{weak} is the set of all weak classifiers, $H^{selected}$ is the set of selected weak classifiers from H^{weak}, y is the prediction of boosting, and α_n is the weight of each selected classifiers.

$$H^{weak} = \{h_1^{weak}, ..., h_M^{weak}\} \tag{2.1}$$

$$H^{selected} = \{h_1^{selected}, ..., h_N^{selected}\} \tag{2.2}$$

$$h_n^{selected} = h_m^{weak} \tag{2.3}$$

$$y = \sum_{n=1}^{N} \alpha_n * h_n^{selected} \tag{2.4}$$

α_n in Equation 2.4 is calculated according to the error of selected weak classifier $h_n^{selected}$.

2.3 Approach

Selected Online Ada-Boosting (SOAB)

In this section, We will describe how to optimize OAB with given depth information on each pixel.

One of the weaknesses of the OAB algorithm is that the target might not always maintain the same size in the scene. The size of the target could be changed when it is occluded, changing poses, or tracking improperly in the current frame (see Figure 1.1). These weak detections pollute our classifier badly. Once the classifier adapts to those unwanted features, the tracker looses the target easily. Here unwanted features include background and foreground features. So, the depth of each pixel plays a significant role to calculate the proportion of unwanted features in the current positive patch. We call this proportion the depth ratio, R. Before computing this depth ratio for the positive patch, we need to determine where our target is in the previous frame (here we focus on the distance between the robot and the person).

Once the initial disparity (called *preDisp*) is computed, we estimate the disparity in the second frame. To do this, we run the original OAB algorithm to detect the positive patch in the second frame. Assuming that the displacement of the target can not be more than a threshold β (on the Bumblebee Camera, we use $\beta = 15$), the possible disparities that belong to the

person are $preDisp \pm \beta$. Then we compute the mean of the pixels within $preDisp \pm \beta$ range as the current disparity (called *curDisp*, see Eq. 2.5). We assign *curDisp* to *preDisp* and repeat this for the subsequent frames to perform tracking.

$$curDisp = Mean(I_p[I_p \in preDisp \pm \beta]) \tag{2.5}$$

The next step is to update the classifier. We do this differently than OAB. We introduce the depth ratio R to evaluate the current positive patch containing a minimum amount of unwanted features. R equals the ratio of the number of pixels that are used to calculate *curDisp* to that of the total number of pixels in the current patch. The width of patch I_p is w, and the height is h.

$$R = \frac{\sum[I_p \in preDisp \pm \beta]}{w * h} \tag{2.6}$$

Now our algorithm (SOAB) makes the decision. If the depth ratio R is greater than a threshold γ, then we update the classifier using the current positive patch. Otherwise, we do not update the classifier.

2.4 Experiments and Evaluation Design

Since the proposed method is different from what people did in the past, we could not find an existing dataset which satisfies our need (a stereo dataset

Data: CameraStream
fetch left and right image from CameraStream;
select target to track;
calculate $curDisp$;
$preDisp \leftarrow curDisp$;
pre-train OAB;
while *true* **do**
  fetch left and right image from CameraStream;
  run OAB to extract a positive patch I_p;
  $curDisp \leftarrow Mean(I_p[I_p \in preDisp \pm \beta])$;
  $R \leftarrow \frac{\sum[I_p \in preDisp \pm \beta]}{w*h}$;
  if $R \geq \gamma$ **then**
    update the classifier;
  end
  $preDisp \leftarrow curDisp$;
end

Algorithm 1: SOAB

for a human following robot under challenging situations). As a result, we build a dataset of 4 image sequences to test the robustness of the person following robot system. The person being followed in our dataset exhibits varying motions and challenging poses in different indoor environments (see Figure 1.1). The dataset is built from image sequences captured by the robot in these places. The robot is following a person in a university hallway, a living room, and a lecture hall. We make the dataset of these three places publicly available at our project page[1]. We provide ground truth for our dataset which has the bounding box drawn on the target being tracked. Demo videos of the robot following behavior of our proposed approach can also be found at

[1] `http://jtl.lassonde.yorku.ca/2017/02/person-following`

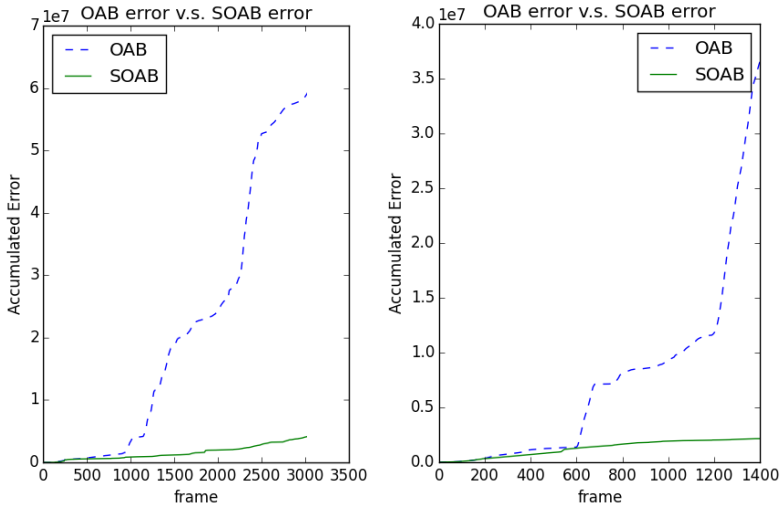
Figure 2.2: Pioneer 3AT robot mounted with a Point Grey Bumblebee stereo camera.

the project page[1]. The dataset consists of the person being followed under varying illumination conditions, different poses of the person being followed, partial and complete occlusion of the person being followed and multiple people present in the scene. The resolution of the images is $640 * 480$ pixels in our dataset. We are able to track people while the robot is moving at up to speeds of 0.70 m/s. It should be noted that our proposed system could be deployed on any mobile robot platform. We tested our proposed approach on a Pioneer 3AT robot (see Figure 2.2). Our algorithm can run in real-time at a

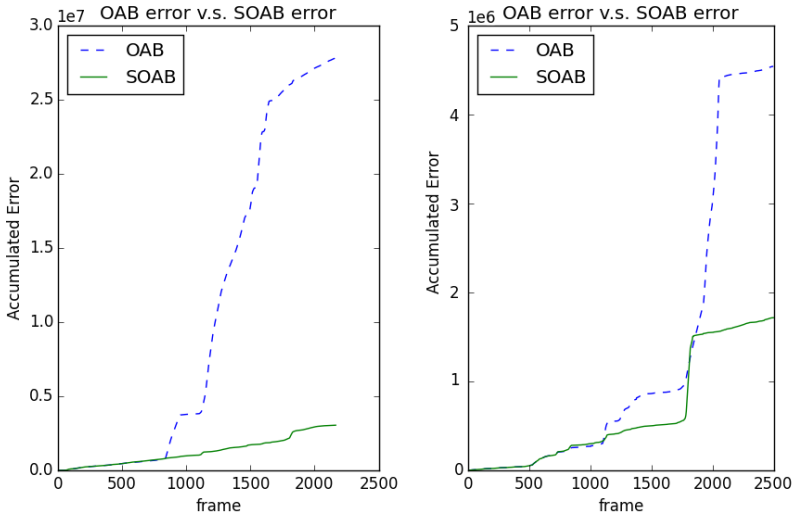
Figure 2.3: (a-g) is tracking using original OAB algorithm. (h-n) is tracking using SOAB with depth ratio threshold $\gamma = 0.30$. (o-u) is tracking SOAB with with depth ratio threshold $\gamma = 0.60$.

frame rate of 15fps on a single CPU core.

First, we tested our algorithm on an experimental sequence of images. The target in the sequence is turning around, and squatting down (see Figure 2.3). From the result, we could distinguish that SOAB with depth ratio threshold $\gamma = 0.60$ outperforms the original OAB. By selecting the patches to



(a) Sequence Hall Way **(b)** Sequence Multiple Crossing



(c) Sequence Same Clothes **(d)** Sequence Lecture Hall

Figure 2.4: The graphs are comparing the accumulated square error on three different image sequences captured in different places and the target acted very differently.

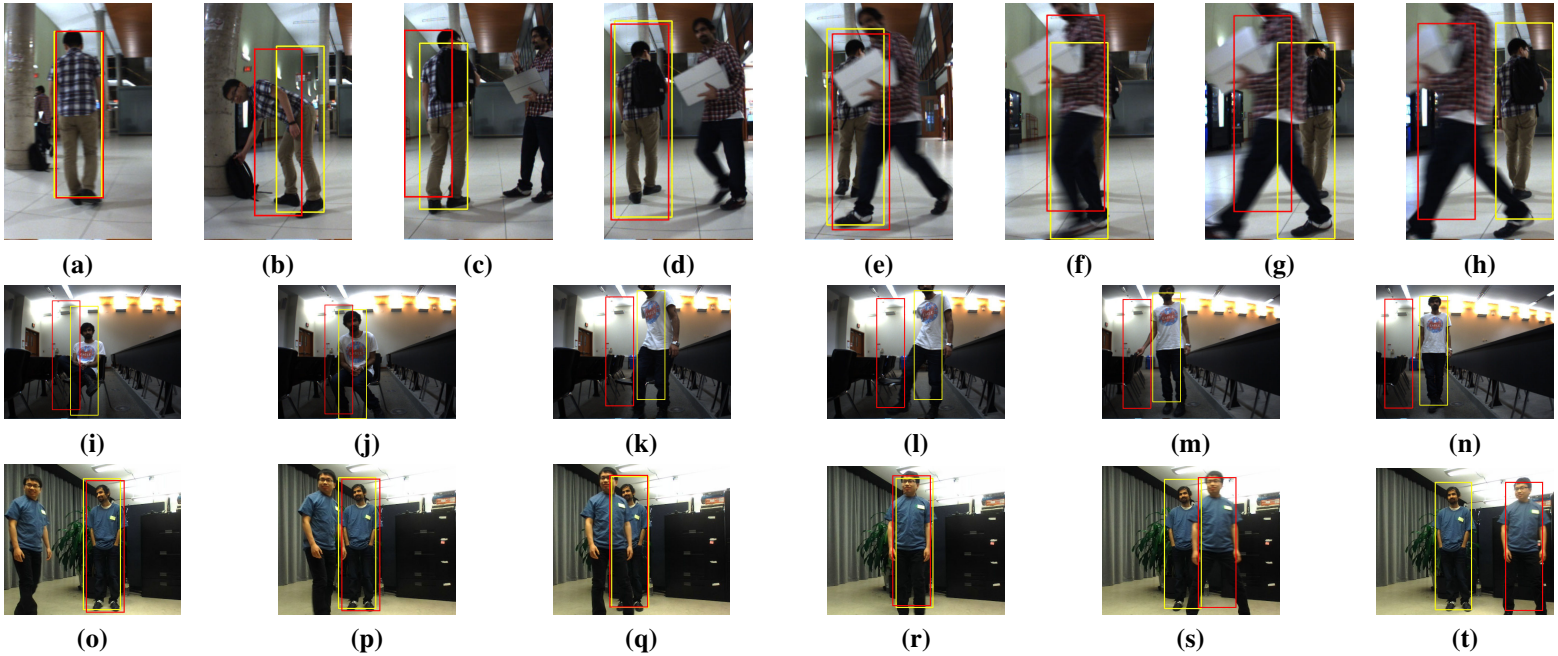
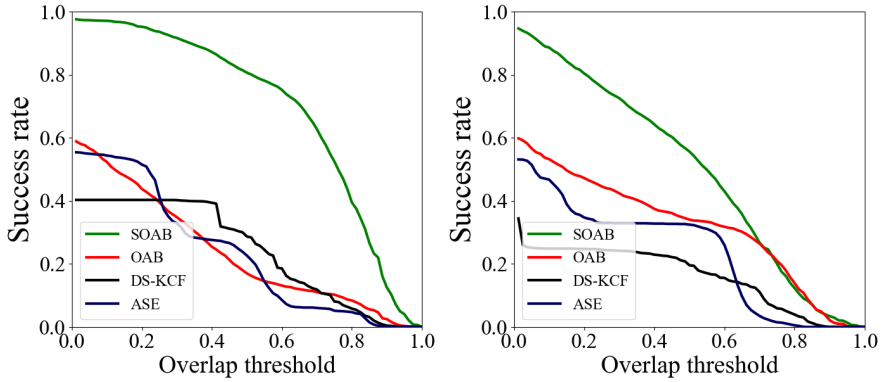


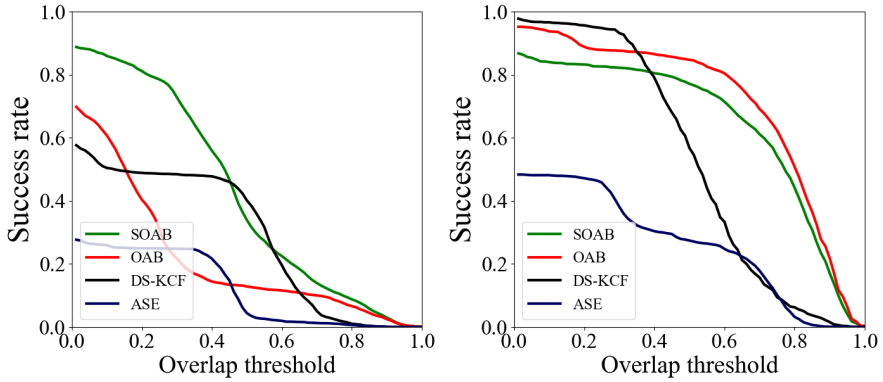
Figure 2.5: Red box is tracking using original OAB algorithm. Yellow box is tracking using SOAB with depth ratio threshold $\gamma = 0.60$. (a-h) are sequences from a hallway. (i-n) are sequences from a lecture hall. (o-t) are sequences showing crossings with same clothes.

update the classifier does make a huge improvement. In Figure 2.3(f), OAB did a mistake and updated the classifier. The classifier then learned the background as the important feature and as a result continuously made mistakes in later frames. On the other hand, SOAB avoids this problem by using the depth information to make decision on whether or not to update the classifier. We also select a depth ratio threshold $\gamma = 0.80$ for testing. Since the threshold is too high, SOAB skipped most of the frames. This is not how we want SOAB to behave. In the later experiment, we fixed the depth ratio threshold as 0.60.

We made another image sequence to test more challenging scenarios. The target is picking up a backpack from the ground, and someone is passing between the robot and the target in the sequence (see Figure 2.5). Again in this test case, SOAB achieved the best result overall. Comparing Figure 2.5(b), OAB learnt the background features leading to a mistake in Figure 2.5(c). From Figure 2.5(e-g), OAB learned the features of the crossing person. The second person became the target of the OAB tracker. Since the depth information is used as a gate, SOAB did not update the classifier with unwanted features when depth ratio is less than the threshold. Figure 2.4(a) shows the accumulated square error of OAB and SOAB. The green line in the graph increases very smoothly meaning that SOAB performed very well without losing track. But, OAB looses track at about frame 900 and becomes very

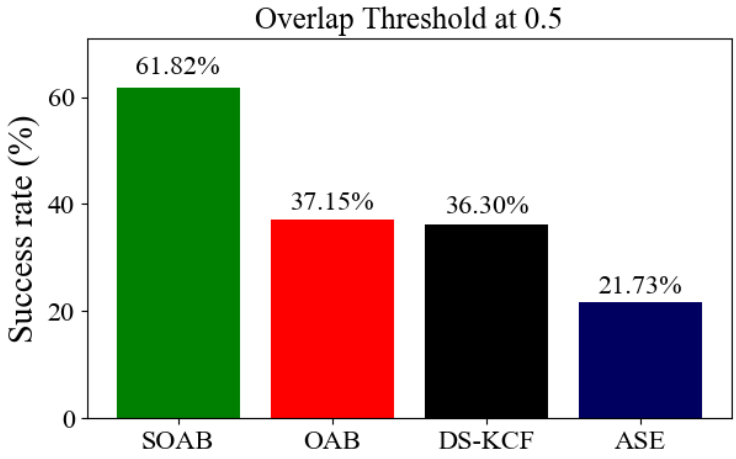


(a) Hallway **(b)** Multi-crossings

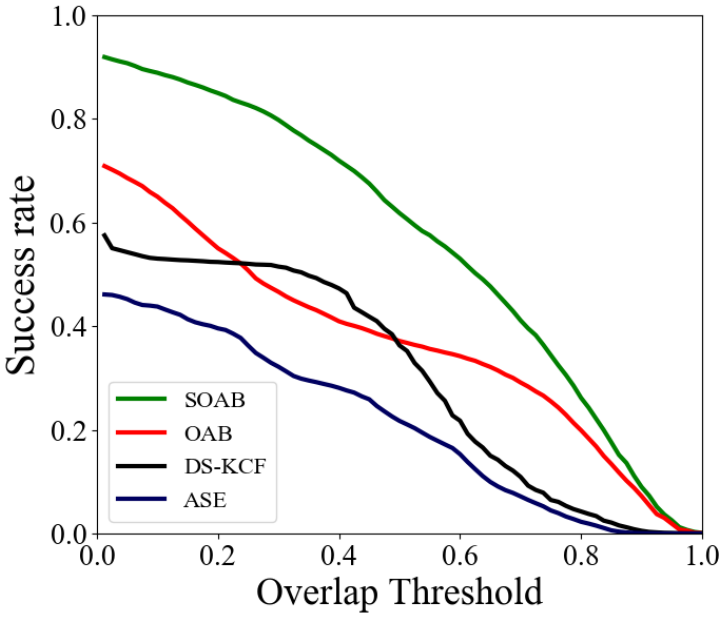


(c) Same Clothes **(d)** Lecture Hall

Figure 2.6: *Success-plots*: comparison between our trackers and different tracking algorithms on 4 sequences. The evaluation is based on the area under the curve. SOAB (ours), OAB [57], ASE [58], DS-KCF [59].

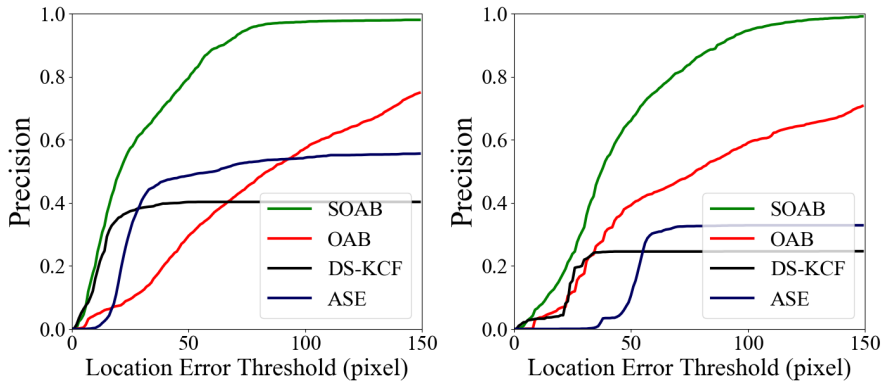


(a) Success rate at overlap threshold 0.5

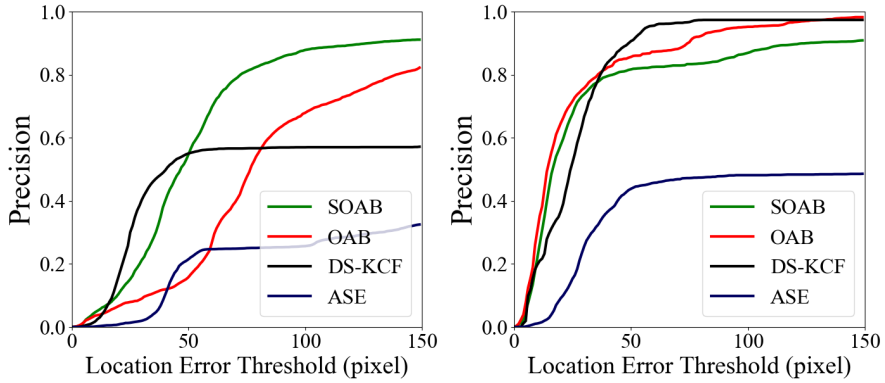


(b) Success Plot

Figure 2.7: *Success-plot*: comparison between our trackers and different tracking algorithms on 4 sequences. The evaluation is based on the area under the curve. SOAB (ours), OAB [57], ASE [58], DS-KCF [59]

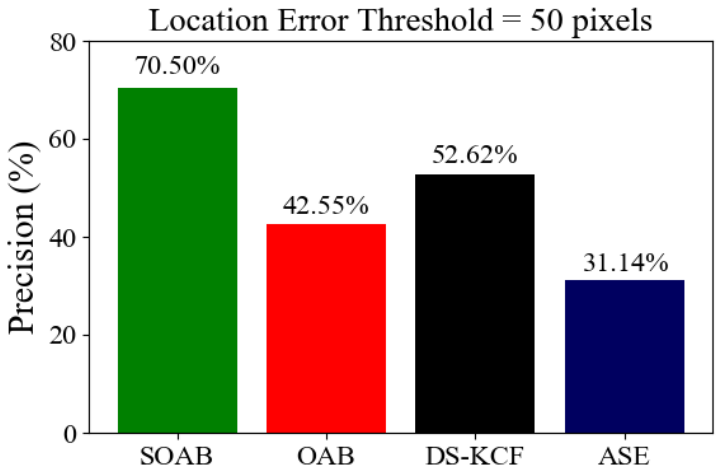


(a) Hallway **(b)** Multi-crossings

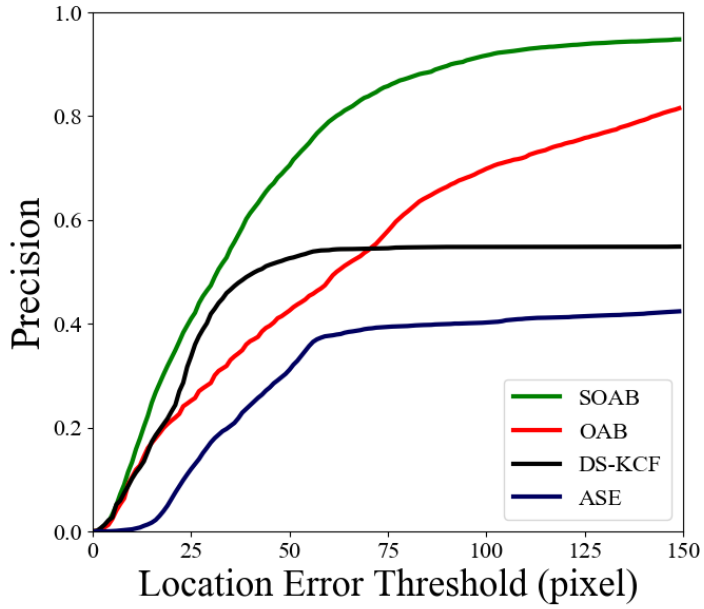


(c) Same Clothes **(d)** Lecture Hall

Figure 2.8: *Precision-plots*: comparison between our trackers and different tracking algorithms on 4 sequences. The evaluation is based on the area under the curve. SOAB (ours), OAB [57], ASE [58], DS-KCF [59].



(a) Precision at location error threshold 50 pixels



(b) Precision Plot

Figure 2.9: *Precision-plot*: comparison between our trackers and different tracking algorithms on 4 sequences. The evaluation is based on the area under the curve. SOAB (ours), OAB [57], ASE [58], DS-KCF [59]

unstable later, roughly at the occlusion in Figure 2.5(f).

Another image sequence was made to test multiple crossing with different speed. The comparison between OAB and SOAB can be seen in Figure 2.4(b). There are 12 crossing actions in this sequence. SOAB completed this test case without failure. But, OAB failed after the fourth crossing.

The third sequence is for testing when two people are wearing the same clothes. This sequence is the most significant one in our dataset. The result can be seen in Figure 2.4(c). In this sequence, two people are crossing each other, walking in a circle. As expected, the robot is following the same person all the time using SOAB.

The overall performance can be seen in Figure 2.6, 2.7, 2.8, and 2.9, where Figure 2.6 and 2.7 are *Success-plots*, and Figure 2.8 and 2.7 are the primary evaluation metric *Precision-plots*. We evaluated the performance of our approach on 4 challenging sequences which exhibit varying situations. It shows that our SOAB outperforms the original OAB tracker and two other popular trackers on three sequences, and SOAB achieves similar performance as the other trackers on Lecture Hall sequence in both *Success* and *Precision* plots.

2.5 Conclusion and Future Work

2.5.1 Conclusion

In this chapter, we described a robust person following robot system using a modified version of Online Ada-Boosting algorithm with only a stereo camera. The system was optimized to perform well in a dynamic environment. Our modified version of OAB performs much better than the original algorithm (see Figure 2.4). We handled difficult situations dealing with similar clothes of people crossing, appearance changes in terms of removing the target's jacket, partial and complete occlusions and were able to run our approach in real-time on a mobile robot. It should also be noted that even though we present our approach for the human following robot, this can be applied to any object following robot as well, but the object needs to be known *a priori*. For instance the robot can follow objects like a handbag, shopping cart, an animal (cat/dog), etc. In this sense our approach targets not only the human following task but also generalizes to other objects as well.

2.5.2 Future Work

We proposed changes to the OAB algorithm. We believe that there could be further improvements, e.g., using a more robust online boosting tracking algorithm called Online Multiple Instance Learning [60], or increasing the

classification error if the bounding box jumps unstably from frame to frame. Another possible future work would be to include the recognition aspect by making use of a human detector to aid in the process of having a better model for the classifier. Another approach of making the following system more reliable could be adding a path planning and obstacle avoidance strategy to the robot control module in our system.

Chapter 3

A Deep Learning approach with Online Update for Human Tracking using Stereo Camera

3.1 Introduction

In this chapter, We introduce a stereo vision based convolutional neural network(CNN) tracker. The tracker is able to track a person in real-time using an online convolutional neural network. my approach can track a target under challenging situations like occlusions, appearance changes, pose changes, crouching, illumination changes or people wearing the same clothes in different environments. The algorithm can also re-identify the target around

corners even when it is momentarily unseen. Raghavender Sahdev and I built an extensive dataset for person-following robots under challenging situations.

3.2 Related work

3.2.1 CNN Trackers

Real-time object tracking is an important task for a person-following robot. Many state of the art algorithms exist that can achieve high accuracy (robustness), e.g., [61] (MGbSA), [62] (CNN as features), [63] (Proposal Selection), [64] (deep learning), [65] (Locally orderless tracking), etc. However these approaches do not target real-time performance. Some other works that focus on computation speed include [66] (Struck SVM with GPU), [67] (Structure preserving), [68] (Online Discrimination Feature Selection), [57] (Online Ada-Boosting), etc. Recent work from Camplani et al. [59] (DS-KCF) used RGBD image sequences from a Kinect sensor to track objects under severe occlusions and ranks highly on the Princeton Tracking Benchmark [4] with real-time performance (40fps). One of the earliest works using convolutional neural networks (CNNs) for tracking appeared in 2010 by Fan et al. [69]. They considered tracking as a learning task by using spatial and temporal features to estimate location and scale of the target. Hong et al. [70] used a pre-trained CNN to generate features to train an SVM classifier. Zhai

et al. [64] also used a pre-trained CNN, but added a Naive Bayes classifier after the last layer of the CNN. Zhang et al. [71] used one single convolutional layer with 50 4-by-4 filters in the CNN structure. The network was trained from scratch, and updated every 5 frames. Gao et al. [62] used pre-trained CNN as feature generator to enhance the ELDA Tracker [72].

3.2.2 CNN Using RGBD images

Training a CNN model with RGB and stereo depth images is another focus of this chapter. Previous work used RGBD CNNs on object detection [73] and object recognition [74]. Couprie et al. [75] used RGBD images to train a single stream CNN classifier to handle semantic segmentation. Eitel et al. [74] trained RGB layers and D layer separately in two CNN streams. These two streams were combined in the fully connected layer. Due to the complexity of the model architecture, we need to modify these models to achieve real-time performance. In the next section, we will show and compare different CNN models for online training and real-time tracking.

3.3 Approach

Here We describe our proposed CNN models and the learning process. The input to the CNN is the RGB channel and the computed depth from the stereo images, We call this as RGBSD (RGB-Stereo Depth). Stereo Depth (SD) is

computed using the ZED SDK[1].

We develop three different CNN models and use each of them separately to validate our approach. The first model (CNN_v1) uses RGBSD layers each as a single image to feed the Convnet. Similar to conventional CNN architectures, the network contains convolutional layers, fully-connected layers, and an output layer (see Figure 3.1). The second model (CNN_v2) uses 2 convolutional streams and the input is RGB channels for one stream and just the stereo depth image for the other (see Figure 3.2). In the fully connected layer, the input is a combination of the flattened output from those two convolutional streams. The third convnet (CNN_v3) is a regular RGB image based CNN (see Figure 3.3). It has a similar structure as that of the first model. Now We describe our approach to initialize and update the CNN tracker.

Initial training set selection: In order to use the CNN model to track a person, We must initialize the CNN classifier. The initialization is done from scratch using random weights. A pre-defined rectangular bounding box is placed in the center of the first frame. To activate the robot following behaviour, a person must stand inside the bounding box at a certain distance from the robot or the target to be tracked can be manually selected. Once the CNN is activated, the patch in the bounding box is labeled as class-1.

[1] `https://github.com/stereolabs/zed-opencv`

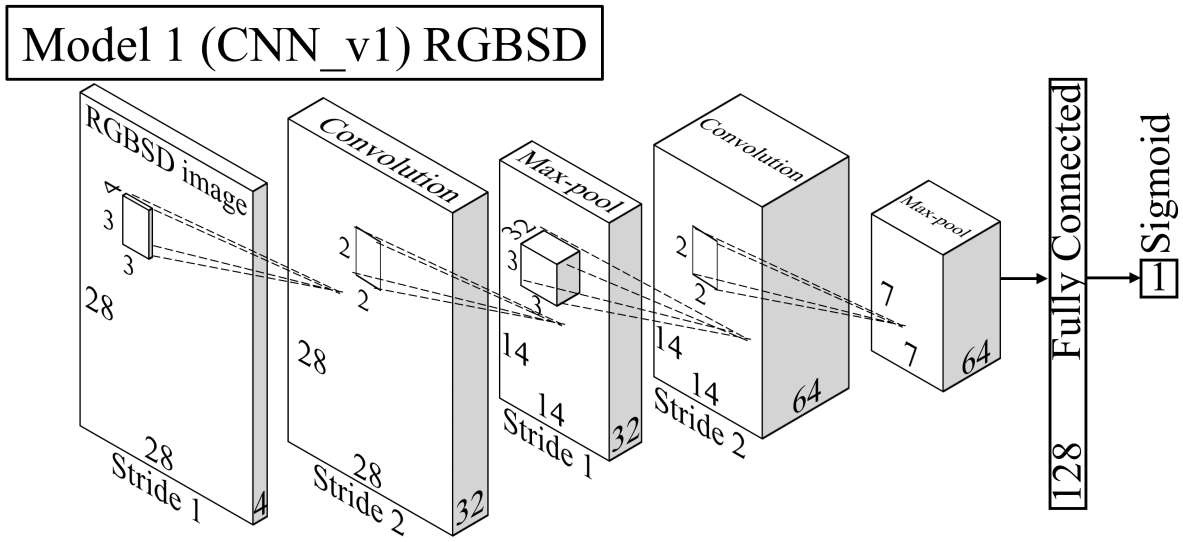


Figure 3.1: CNN model: Model CNN_v1 takes a 4-channel RGBSD image as input. The parameters of the CNN in each of the layers are chosen empirically for real-time performance.

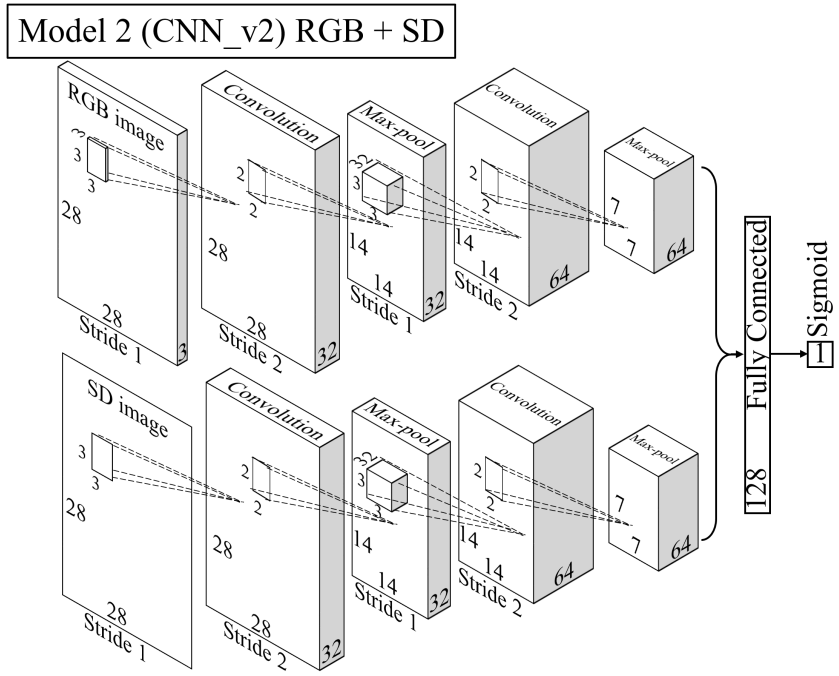


Figure 3.2: CNN model: Model CNN_v2 takes an RGB image and a SD image as input. The parameters of the CNN in each of the layers are chosen empirically for real-time performance.

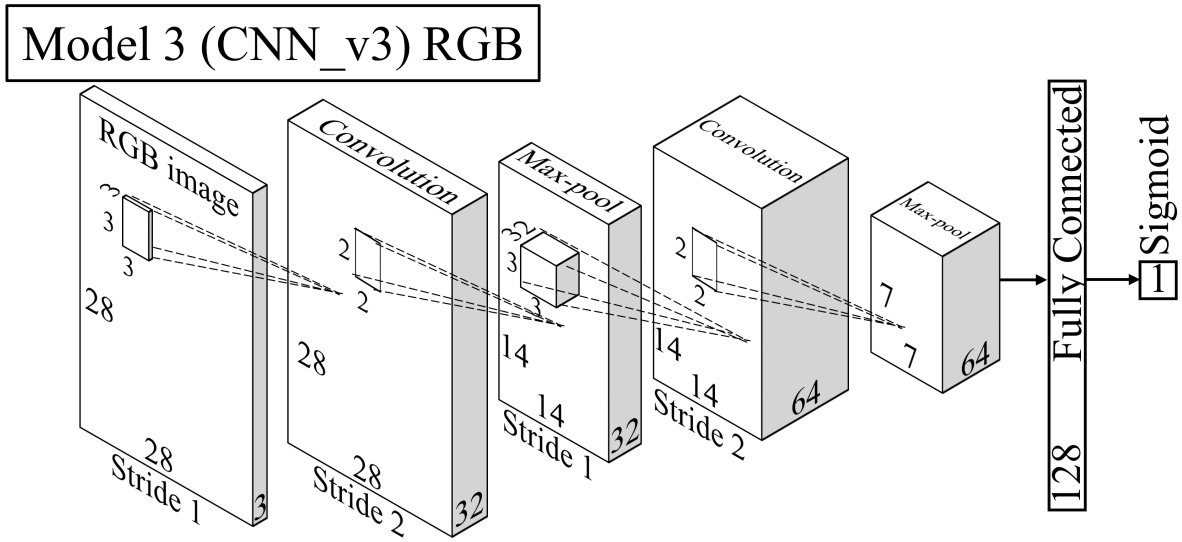
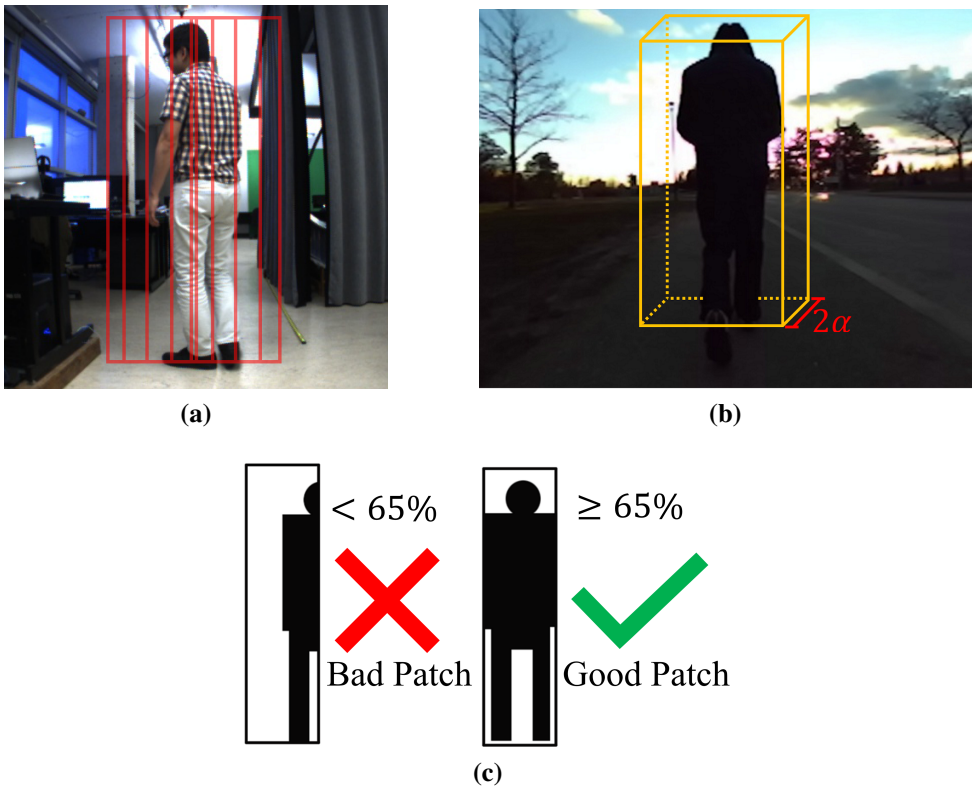


Figure 3.3: CNN model: Model CNN_v3 takes an RGB image only as the input. The parameters of the CNN in each of the layers are chosen empirically for real-time performance.

The patches around the bounding box are labeled as class-0. Since these two classes are highly unbalanced, we uniformly select n patches from class-0, and copy the class-1 patch n times to form the training set ($n = 40$ in our experiment). This initial training set is used to train a CNN classifier until it has a very high accuracy on the training set. This might make the classifier over-fit the training set. To handle this strong over-fitting, we assume that the target pose and appearance should not change a lot in the first 50 frames (about 2-3 seconds).

Test set selection: Once the CNN classifier is initialized or updated, we use it to detect the target in the next frame. When a new frame is available along with the stereo depth layer, we search the test patches in a local image region as shown in Figure 3.4a. We also restrict the search space with respect to the depth as shown in Figure 3.4b. If the patches in the image do not have the depth within $previous\_depth \pm \alpha$, we do not consider them (Figure 3.4c), where α is the search region in depth direction (We use $\alpha = 0.25$ meters). By doing this, most of the patches belonging to the background will be filtered out before passing to the CNN classier. Only the highest responses on class-1 will be considered as the target in the current frame. If no target is detected (e.g., highest responses on class-1 < 0.5) after 0.5 sec, it will enter the target missing mode. Then, the whole image is scanned to create a test set.



(a) (b) (c)

Figure 3.4: 3D search region for test set (a) candidate test patches in 2D region (based on a sliding window approach), (b) search region with respect to depth, (c) pixels in black are within $\pm\alpha$ *meters* from the previous depth. If black pixels are less than 65% of the patch, the patch is bad, else, it is a good patch. The number 65% is chosen experimentally as this covers the human body completely in most of the cases. According to (c), the red and blue patches in (a) are bad patches, the green, pink, and yellow patches are good patches.

Update CNN tracker: To update the classifier, a new training set needs to be selected. The update step is performed only if the detection step finds the target (class-1) in the test set. In order to maintain robustness, the most recent 50 class-1 patches are retained from the previous frames to form the class-1 patch pool which is implemented as a First-In-First-Out queue. The

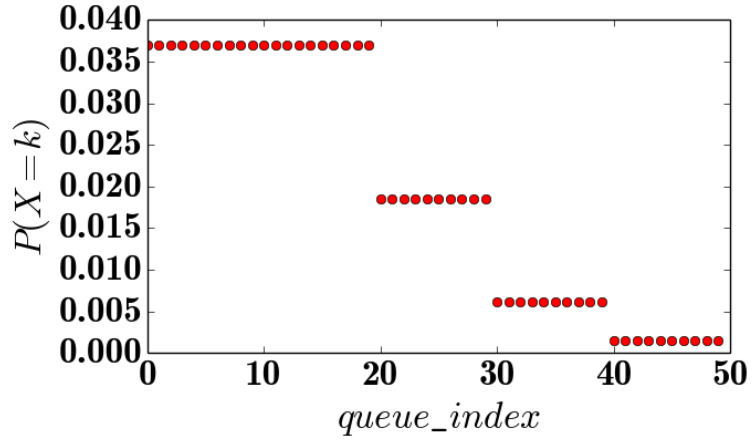
patches around the target form the class-0 patch pool. In this new training set, we again uniformly select n patches from class-0 patch pool. For selecting n patches from class-1 patch pool, we sample the patches based on a Poisson distribution with $\lambda = 1.0$ and $k = \lfloor \frac{queue\_index}{10} \rfloor$ (see Equation 3.1 and Figure 3.5). This gives a higher probability of selecting patches from the recent history rather than selecting older patches. This training set is used to update the classifier. The Poisson distribution based sampling of class-1 patches avoids over fitting and provides a chance to recover from bad detection in the previous frame(s). It should be noted here that over a period of time the classifier only becomes better as the update step adds more data to the model thereby improving the system.

$$P(k) = e^{-\lambda} \frac{\lambda^k}{k!} \tag{3.1}$$

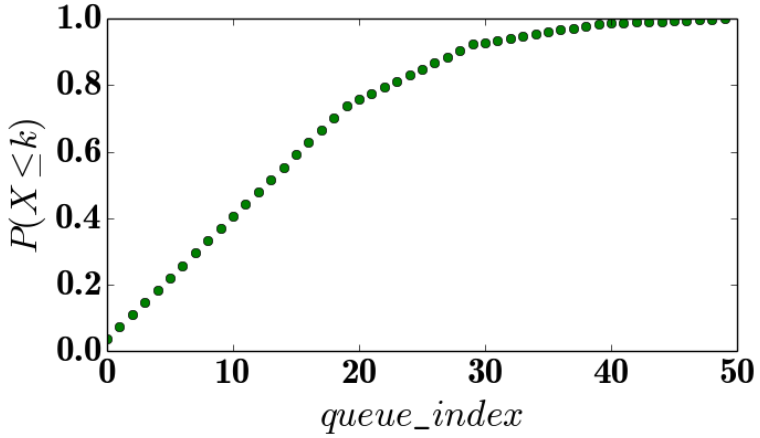
3.4 Dataset

Dataset: Several Datasets exist for pedestrian detection and tracking[2]. In particular, the Princeton Tracking Benchmark [4] provides a unified RGBD dataset for object tracking which includes various occlusions and some appearance changes. But, each sequence is very short (maximum 900 frames,

[2] `http://homepages.inf.ed.ac.uk/rbf/CVonline/Imagedbase.htm#people`



(a) normalized PMF



(b) CDF

Figure 3.5: Poisson distribution with $\lambda = 1.0$ and $k = \lfloor \frac{queue\_index}{10} \rfloor$, where *queue_index* is the patch index in First-In-First-Out queue. To select an index, just randomly generate a real number from 0 to 1.0. Then, base on (b) the CDF graph, an index is selected.

most of them are under 300 frames). Many other works exist that aim at solving the person following problem, however there is a lack of a standardized dataset which could be used to validate the tracking algorithm used for person following robots. In this work, my partner Raghavender Sahdev and I

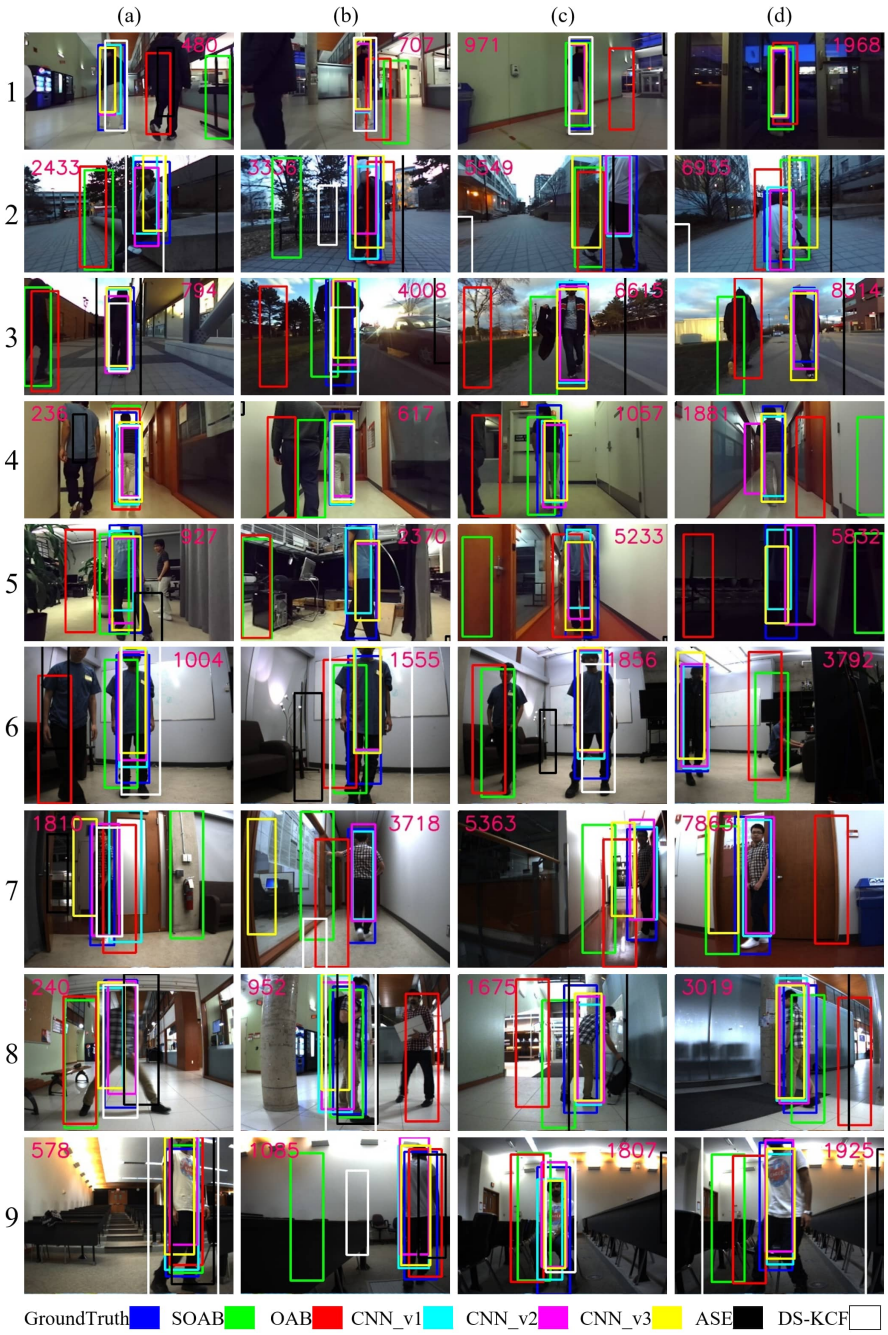


Figure 3.6: Compare some tracking algorithms on our dataset. (1): Hallway 2; (2): Walking Outdoor; (3): Sidewalk; (4): Corridor Corners; (5): Lab & Seminar; (6): Same Clothes 1; (7): Long Corridor; (8): Hallway 1; (9): Lecture Hall. (SOAB [15], OAB [57], ASE [58], DS-KCF [59])

built an extensive stereo dataset (left, right and depth images) of 9 indoor and 2 outdoor sequences. Each sequence has more than 2000 frames and up to approximately 12000 frames. The dataset has challenging sequences which have pose changes, intense illumination changes, appearance changes (target removing/wearing a jacket, exchanging jacket with another person, removing/wearing a backpack or picking-up/putting-down an object), crouching and walking, sitting on a chair and getting up, partial and complete occlusions, occlusions by another person wearing same clothes and some other different situations. The dataset also has image sequences when the target is not visible transiently in the image and reappears after some time. The dataset is built in different indoor and outdoor environments in a university context. Some of the samples from the dataset can be seen in Figure 3.6. The images are captured at a frame rate of 20 Hz and the resolution is standard VGA (640 x 480) for bumblebee2 and (672 x 376) for ZED. We also provide with ground truth of the image sequences at this link[3]. The ground truth contains the bounding box labelled for the target (human) which is manually labelled by human annotators for each frame.

Evaluation metric: The interest of person following task is to follow a person, so the size of the bounding box is not important for the robot. How-

[3] `http://jtl.lassonde.yorku.ca/2017/05/person-following-cnn/`

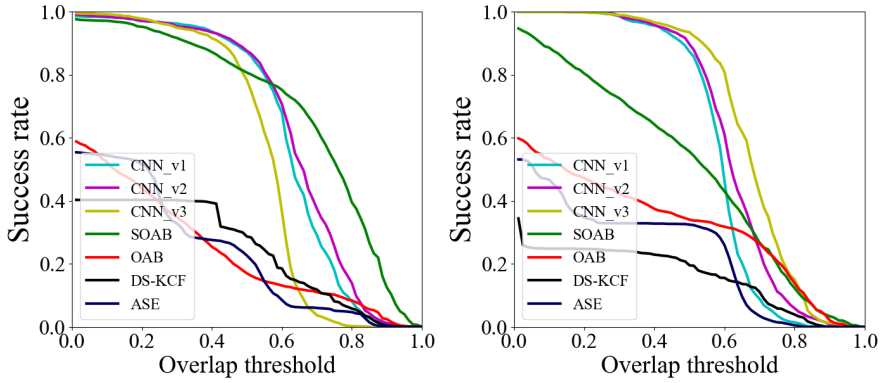
	Success	Precision	Speed (fps)
SOAB	0.5898	0.7443	15
CNN_RGBSD (v1)	**0.7857**	**0.8538**	
CNN_RGB_SD (v2)	0.7560	0.7590	20
CNN_RGB (v3)	0.6741	0.7746	

Table 3.1: Comparing the processing speed, the precision at location error threshold 50 pixels, and the success rate at overlap threshold 0.5 on our proposed trackers

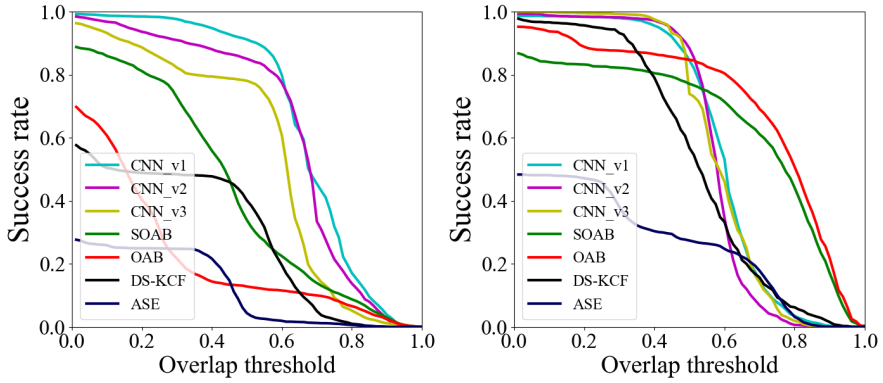
ever, the centre of the target plays an important role. The evaluation of tracking algorithms have been done in numerous ways. Wu et al. [3] provide details about various existing evaluation metrics that have been used for tracking. For our dataset we use the *precision-plot* as defined in [3] as the metric to evaluate the performance of our proposed approach. We report the percentage of frames in which the center of the detected bounding box is within a specific range of pixels from the ground truth. Since the initial bounding box size is about (100 x 350) for all the video sequences, we compute the average precision of all sequences using location error threshold 50 (pixels) to evaluate tracker performance.

3.5 Experiments

Experiments: We validated our proposed approach in different indoor and outdoor environments. We achieved a frame rate of approx. 20 fps depending on the search window size that we use for the depth range and the local im-

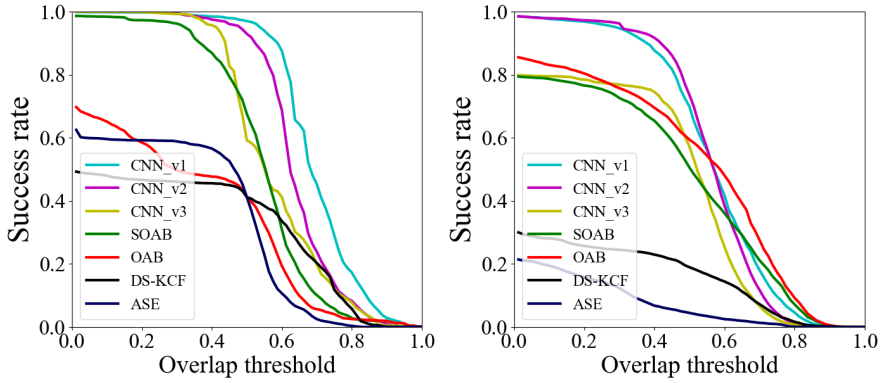


(a) Hallway 1 **(b)** Multi-crossings

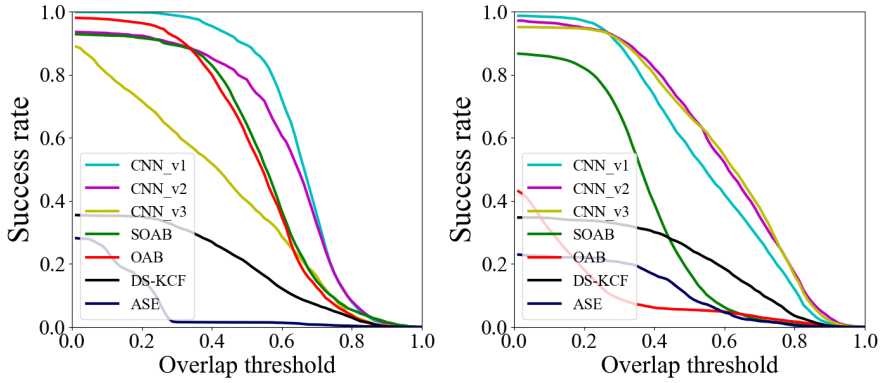


(c) Same Clothes 1 **(d)** Lecture Hall

Figure 3.7: *Success-plots*: comparison between our trackers and different tracking algorithms on our dataset. The evaluation is based on the area under the curve. SOAB [15], OAB [57], ASE [58], DS-KCF [59]

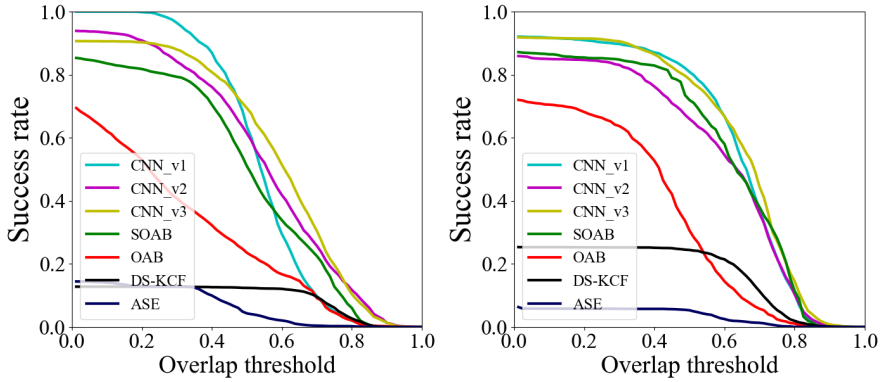


(a) Same Clothes 2 **(b)** Long Corridor

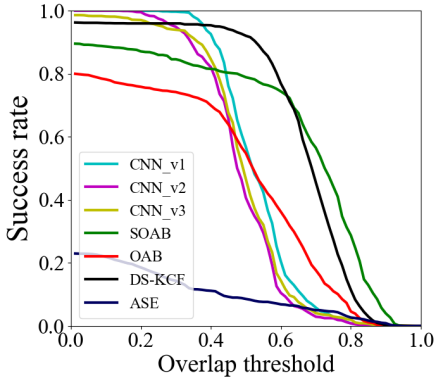


(c) Walking Outdoor **(d)** Sidewalk

Figure 3.8: *Success-plots*: comparison between our trackers and different tracking algorithms on our dataset. The evaluation is based on the area under the curve. SOAB [15], OAB [57], ASE [58], DS-KCF [59]

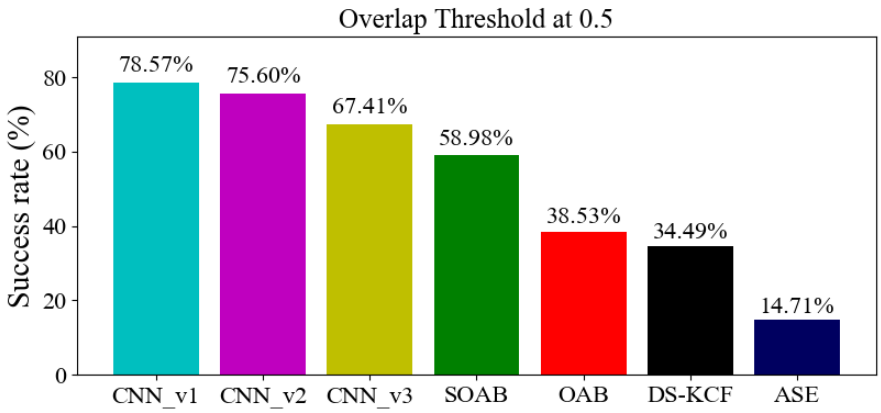


(a) Lab & Seminar **(b)** Corridor Corners

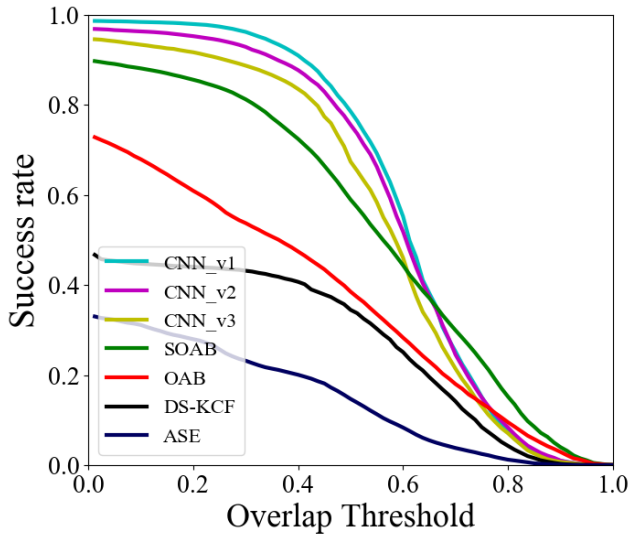


(c) Hallway 2

Figure 3.9: *Success-plots*: comparison between our trackers and different tracking algorithms on our dataset. The evaluation is based on the area under the curve. SOAB [15], OAB [57], ASE [58], DS-KCF [59]

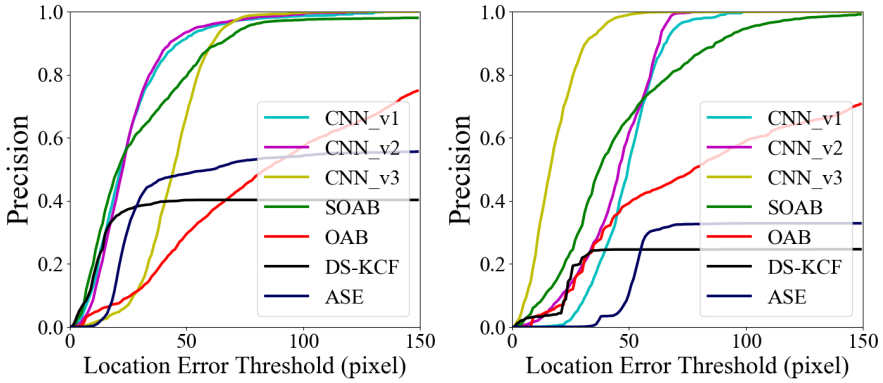


(a) Success rate at overlap threshold 0.5

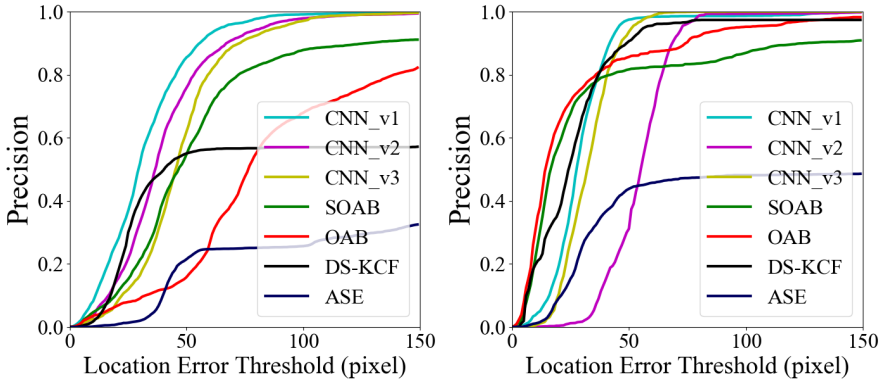


(b) Success Plot

Figure 3.10: *Success-plot*: comparison between our trackers and different tracking algorithms on our dataset with 11 sequences. The evaluation is based on the area under the curve. SOAB [15], OAB [57], ASE [58], DS-KCF [59]

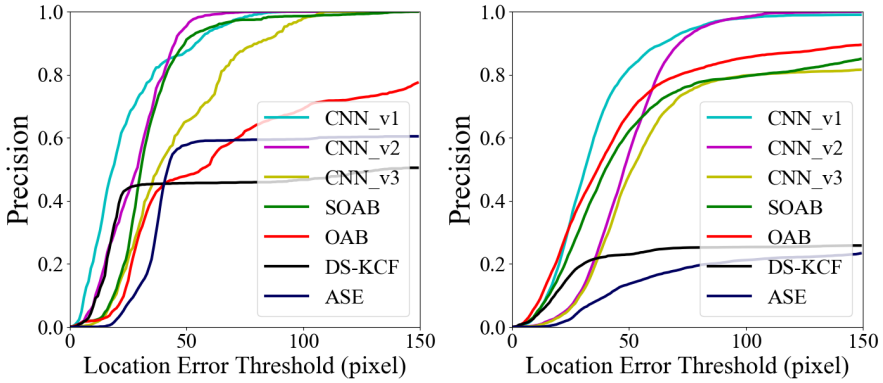


(a) Hallway 1 **(b)** Multi-crossings

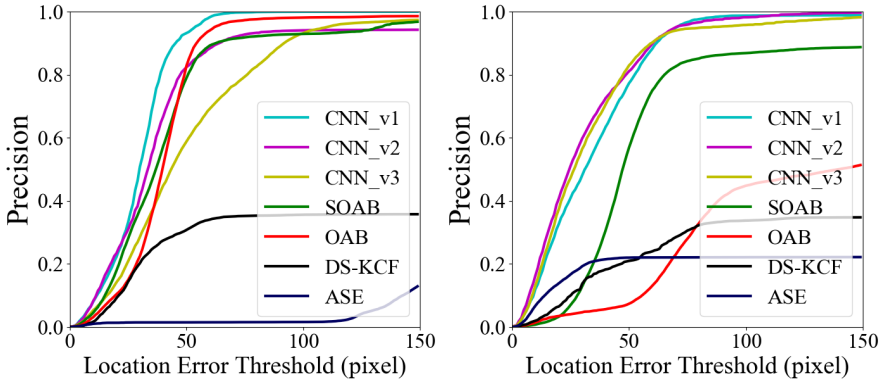


(c) Same Clothes 1 **(d)** Lecture Hall

Figure 3.11: *Precision-plots*: comparison between our trackers and different tracking algorithms on our dataset. The evaluation is based on the area under the curve. SOAB [15], OAB [57], ASE [58], DS-KCF [59]

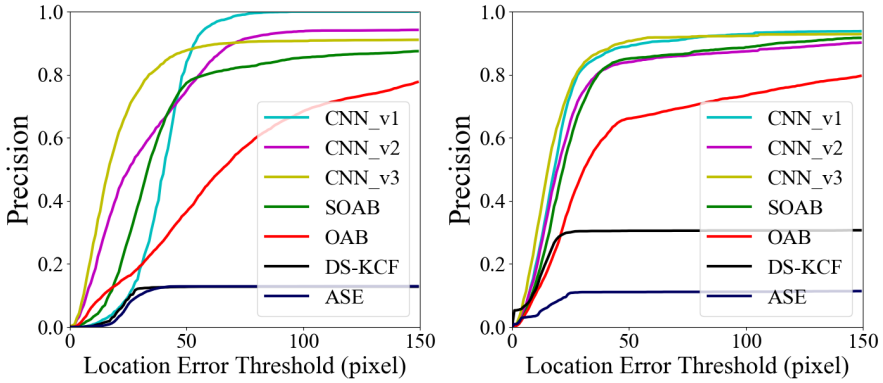


(a) Same Clothes 2 **(b)** Long Corridor



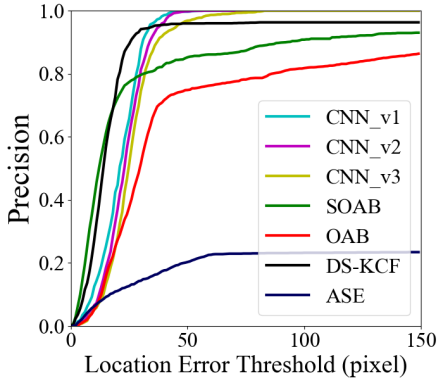
(c) Walking Outdoor **(d)** Sidewalk

Figure 3.12: *Precision-plots*: comparison between our trackers and different tracking algorithms on our dataset. The evaluation is based on the area under the curve. SOAB [15], OAB [57], ASE [58], DS-KCF [59]



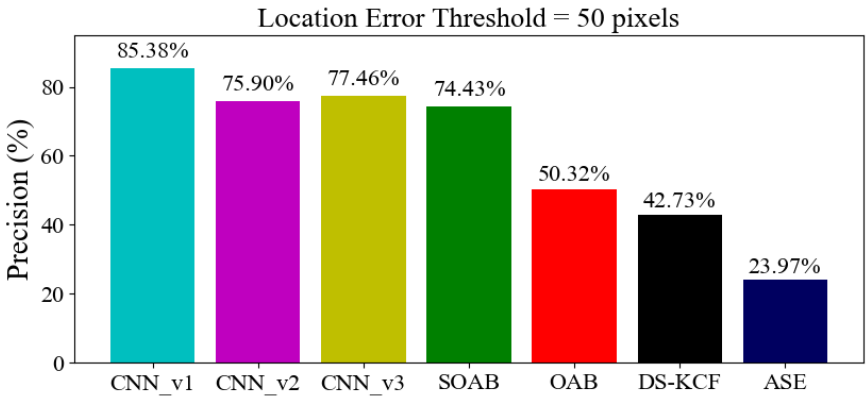
(a) Lab & Seminar

(b) Corridor Corners

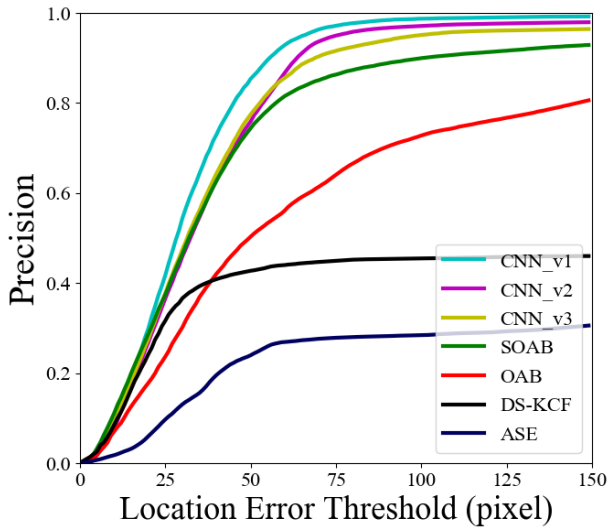


(c) Hallway 2

Figure 3.13: *Precision-plots*: comparison between our trackers and different tracking algorithms on our dataset. The evaluation is based on the area under the curve. SOAB [15], OAB [57], ASE [58], DS-KCF [59]



(a) Precision at location error threshold 50 pixels



(b) Precision Plot

Figure 3.14: *Precision-plot*: comparison between our trackers and different tracking algorithms on our dataset with 11 sequences. The evaluation is based on the area under the curve. SOAB [15], OAB [57], ASE [58], DS-KCF [59]

age search region (See Table 3.1). For evaluation we compare 3 versions of our tracking algorithm with 4 other existing stereo vision based trackers (for which the code is publicly available). We used the *precision-plot* evaluation metric as defined in [3] to report the performance of our system. The performance can be seen in Figure 3.7, 3.8, 3.9, 3.10, 3.11, 3.12, 3.13, and 3.14, where Figure 3.7, 3.8, 3.9, and 3.10 are *Success-Plots*, and Figure 3.11, 3.12, 3.13, and 3.14 are the primary evaluation metric *Precision-Plots*. We evaluated the performance of our approach on 11 challenging sequences which exhibit varying situations as described in the previous section. It was found that the RGBSD based CNN (CNN_v1) outperformed all other existing approaches. The RGB based CNN (CNN_v3) could not perform better than SOAB [15] in some sequences. We also compare our approach with Martin et al. [58] (ASE with monocular images) and Camplani et al. [59] (DS-KCF with RGBD images). A demo video of our approach on the robot under different situations can be found at the link [3].

3.6 Conclusion and Future Work

3.6.1 Summary

In this chapter, we described a robust real-time person following robot tracking algorithm using an online update Convolutional Neural Network model.

The proposed tracker (CNN_RGBSD / CNN_v1) could perform very well in dynamic environments under challenging situations, e.g., heavy appearance changes, pose changes, full occlusions, illumination changes, distractors with similar appearance, etc. The tracker runs at about 20 fps on a laptop with a mobile GPU. We also captured a novel human tracking dataset with stereo images, where the left, right, and depth are given. The dataset is open to the public and posted on our webpage [3].

3.6.2 Future work

The updating approach to train a convolutional neural network online is very time-consuming. It is necessary to have a pre-trained model with fix parameters for human tracking. But, these pre-trained models require a large dataset and training time for parameters/hyper-parameters fine-tuning. To achieve the goal, we need to build a large human tracking dataset which covers all the challenge situations repeatedly under different environments.

Chapter 4

A Deep Learning Approach without Online Update for Human Tracking using Stereo Camera

4.1 Introduction

In this chapter, We evaluate a pre-trained Siamese architecture based CNN, called SiamMask [76], for online Human Tracking on our human tracking dataset. Since the network is heavily trained on an object tracking dataset with 97 object categories (containing the human category also), this approach predicts the bounding boxes on the human body very precisely, so the algorithm can track a target with a stronger IoU and precision. By integrating the

stereo depth and a temporary appearance model, the performance is brought to 98.92% accuracy on 50 pixels location error threshold comparing to the top performance 85% in Chapter 3. The integrated tracker could handle more challenging cases like deformations(e.g., dancing, running), sudden illumination changes, identical clothed people crossing each other, etc.

4.2 Related work

In this section, We will review several significant Siamese based CNN tracking algorithms.

The first Siamese network based object tracking algorithm (**SiamFC**) was introduced by Bertinetto *et al.* [77] in 2016. The Siamese network is trained offline on a dataset for object detection in videos. The input to the network are two images, one is an exemplar image z, the other one is the search image x. Then, a dense response map is generated from the output of the network. **SiamFC** learns and predicts the similarity between the regions in x and the exemplar image z. In order to handle the object scale variantion, SiamFC searches for objects at five scales $1.025^{\{2,1,0,1,2\}}$ near the target's previous location. As a result, there will be 5 forward passes on each frame. SiamFC runs at about 58 fps, which is the fastest fully convolutional network (CNN) based tracker comparing to online training and updating networks in

2016. However, SiamFC is an axis-aligned bounding box tracker. It couldn't outperform the online training and updating deep CNN tracker MDNet [78] (1 fps) in terms of average overlap accuracy.

He *et al.* [79] combines two branches (Semantic net and Appearance net) of Siamese network (**SA-Siam**) to improve the generalization capability of **SiamFC**. Two branches are individually trained, and then the two branches are combined to output the similarity score at testing. S-Net is an AlexNet [80] pretrained on an image classification dataset. A-Net is a SiamFC pretrained on an object detection from video dataset. S-Net improves the discrimination power of the SA-Siam tracker because different objects activate different sets of feature channels in the Semantic branch. Due to the complexity of the two branches, SA-Siam runs at 50 fps when tracking with pretrained model.

By modifying the original Siamese net with a Region Proposal Network(RPN) [81], Li *et al.* [82] proposed a Siamese Region Proposal Network (**SiamRPN**) to estimate the target location with the variable bounding boxes. The output of **SiamRPN** contains a set of anchor boxes with corresponding scores. So, the bounding box with the best score is considered as the target location. The benefit of RPN is to reduce the multi-scale testing complexity in the traditional Siamese networks (SiamFC, SA-Siam). An updated ver-

sion SiamRPN++ [83] has released in 2019. In terms of processing speed, SiamRPN is 160 fps and SiamRPN++ is about 35 fps.

Unlike **SiamFC**, **SA-Siam**, and **SiamRPN** yielding axis-aligned bounding boxes, **SiamMask** [84] uses the advantage from a video object segmentation dataset and trained a Siamese net to predict a set of masks and bounding boxes on the target. The bounding boxes are estimated based on the masks using rotated minimum bounding rectangle (MBR) at the speed of 87 fps. However, the MBR does not always predict the bounding boxes that perfectly align with the ground truth bounding boxes. Although the same bounding boxes prediction algorithm used in VOT2016 for generating the ground truth can improve the average overlap accuracy dramatically, the running speed decreases to 5 fps. To address this problem, we present a new method in Section 4.3 that can process frames in real-time and achieves a better result.

4.3 Approach

Because of the characteristic of **SiamMask** network, it can track a human target with almost perfect precision under most challenging situations. However, according to our experiments, **SiamMask** weakly handles the long-term target tracking cases involved like complete occlusion, target disappearance in the image frame, target wearing the similar clothes to other objects, etc

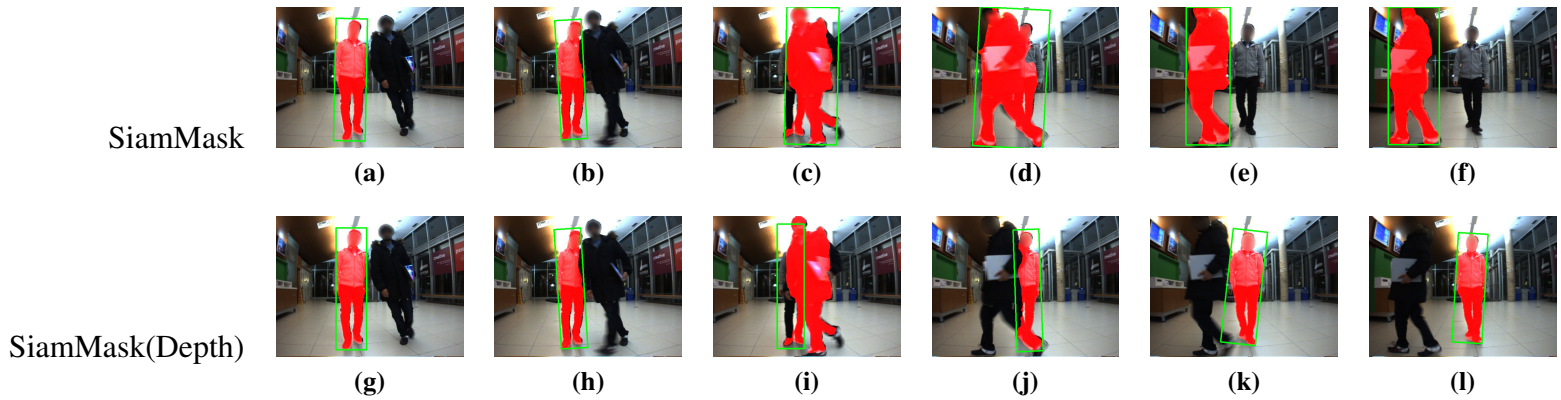


Figure 4.1: Sample tracking results when the target is completely occluded.

(See Figure 4.1). In this section, We will introduce two modifications on SiamMask to improve the long-term tracking capability.

4.3.1 Using depth (SiamMD)

Here we describe our modification to **SiamMask**. The input to the network is the RGB channel. There is an extra layer of computed depth from the stereo images, we call this as RGBSD (RGB-Stereo Depth). Stereo Depth (SD) is computed using the ZED SDK[1]. This stereo depth is applied to restrict the search area in 3-dimensional space.

Since **SiamMask** could compute a precise mask on the target region, we could calculate the target depth ($Target\_Depth$) using Equation 4.1 on subsequent frames.

$$Target\_Depth = Mean(Img_{Depth}[Img_{Mask} > 0]) \tag{4.1}$$

Once occlusion happens, the $Target\_Depth$ will become unreliable. Assuming that the displacement of the target cannot be more than a threshold β (on Bumblebee camera and ZED Camera, We use $\beta = 0.2meter$), so that the valid current $Target\_Depth$ is defined as the equation below:

$$Target\_Depth_{curr} \in Target\_Depth_{pre} \pm \beta \tag{4.2}$$

[1] `https://github.com/stereolabs/zed-opencv`

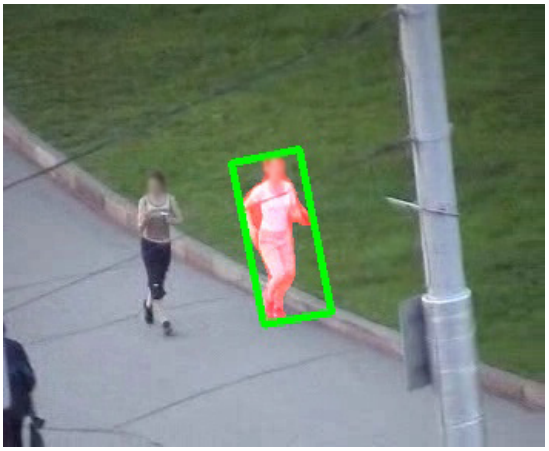


Figure 4.2: An example of fine mask generated by SiamMask [76]

If the $Target\_Depth_{curr}$ is valid, then the current bounding box will be updated; otherwise, we keep the previous bounding box and the previous $Target\_Depth$. By applying this simple strategy, our tracker performs with more than 97% accuracy on a person-following dataset.

Figure 4.15, and 4.14 show the quantitative comparison on our person-following dataset in Chapter 3. The **SiamMD** tracker is an update version of SiamMask after integrating the Depth channel without using an appearance model. In the next subsection, We will describe an approach using an appearance model.

4.3.2 Appearance model (SiamMDH)

Unlike the online learning tracking models (e.g.,EnCKF [85], KCF [86]), Siamese trackers are not tracking the object based on the high-level features (e.g., texture, color, etc.) but the low-level features from the convolutional

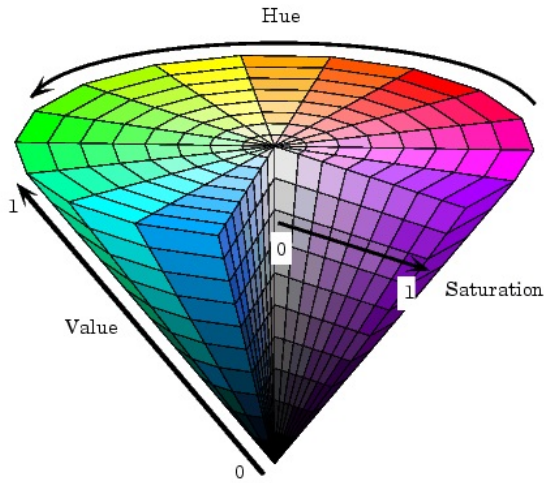


Figure 4.3: HSV color space. Image source `https://edoras.sdsu.edu/doc/matlab/toolbox/images/color11.html`

neural network (CNN). The CNN features are more likely to represent a class of the object, rather than the instance itself. Because of the weakness of the CNN features, those trackers have difficulty handling the full occlusion and out-of-view challenges. Therefore, it is required to have a stronger appearance model to distinguish the distractors and the real target.

Since **SiamMask** generates fine masks on the target (See Figure 4.2), by comparing the color spectrum of the predicted masks, we get a feasible appearance model to differentiate the distractors and the true target. The colour histogram is robust to a variety of viewpoints and deformable objects [87]. By using the HSV (Figure 4.3) colour space, we can reduce the effect of illumination or brightness information by leaving out the V (intensity) channel [88].

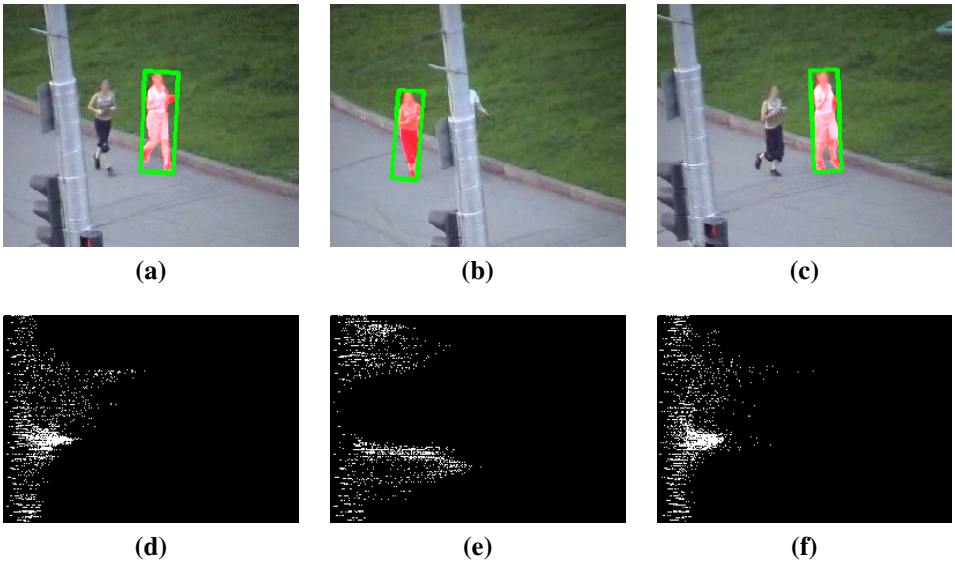


Figure 4.4: H-S histograms comparison between different humans and poses. The images at the bottom are the histogram of the masks in the first row.

To compare two H-S histograms, it is important that Normalized Cross Correlation (NCC) is robust to the object size variant mathematically. As a result, NCC is a feasible function to compare two histograms.

Figure 4.4 shows that the H-S histograms among people wearing different clothes are highly distinguishable. On the other hand, the same person in different poses has very similar H-S histograms. Notice that in Figure 4.4 (a) and (c) are the same person whereas (b) is a different person. The performance of applying the appearance model (SiamMDH) can be found in Figure 4.15, and 4.14.

	Success	Precision	Speed (fps)
SOAB	0.5898	0.7443	15
CNN_RGBSD (v1)	0.7857	0.8538	
CNN_RGB_SD (v2)	0.7560	0.7590	20
CNN_RGB (v3)	0.6741	0.7746	
SiamMask	0.8723	0.9296	**50**
SiamMD	0.9093	0.9740	39
SiamMDH	**0.9294**	**0.9892**	37

Table 4.1: Comparing the processing speed, the precision at location error threshold 50 pixels, and the success rate at overlap threshold 0.5 on our proposed trackers

4.4 Experiments

We tested our proposed approach on the same dataset that was described in Chapter 3. Our SiamMDH tracker achieved a frame rate of approx. 37 fps depending on the local search window size, and SiamMD achieved approx. 39 fps (see Table 4.1). For evaluation, we compare 2 versions of our Siamese tracking algorithm with 8 other existing stereo vision based trackers (some of them are from the previous chapters). We used the primary evaluation metric *precision-plot* to report the performance. The performance can be seen in the following figures: Figure 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, and 4.15, where Figure 4.6, 4.7, 4.8, 4.9, and 4.10 are *Success-plots*, and Figure 4.11, 4.12, 4.13, 4.14, and 4.15 are the primary evaluation metric *Precision-Plots*. It was found that our SiamMDH tracker outperformed all other trackers. Despite the state-of-the-art overall performance, the Siamese

network based trackers provide the most robust tracking performance. On the other hand, the online CNN trackers that described in Chapter 3 are not stable in some image sequences (e.g., Figure 4.11(b)(e), etc.). In these sequences (*Multi-crossings* and *Same Clothes 2*), we aim to test intensive distractor crossing between the target and the camera. Particularly, the sequence *Same Clothes 2* has two identical clothed people, and the distractor and the target are spatially very close to each other in some video segments (See Figure 4.5).

4.5 Conclusion and Future Work

4.5.1 Summary

In this chapter, we described a state-of-the-art robust human tracking algorithm using a Siamese based convolutional neural network. The proposed tracking algorithm reached 98.92% accuracy on our dataset which was captured in dynamic environments under challenging situations. The result showed that our SiamMDH tracker is the most stable and robust on our novel human tracking dataset with 11 challenging sequences.

4.5.2 Future work

Our approach focused on the depth of the target and its appearance. On a different aspect, if a proper motion model is employed, we believe the result

could move to the next level. To attain this, a real-time algorithm is needed to differentiate the camera the target motion in order to estimate the real target motion. As well as, we need to beware the other dynamic distractors in the scene. If the target locations are known, the future target positions can be simulated using the Kalman filter [89].

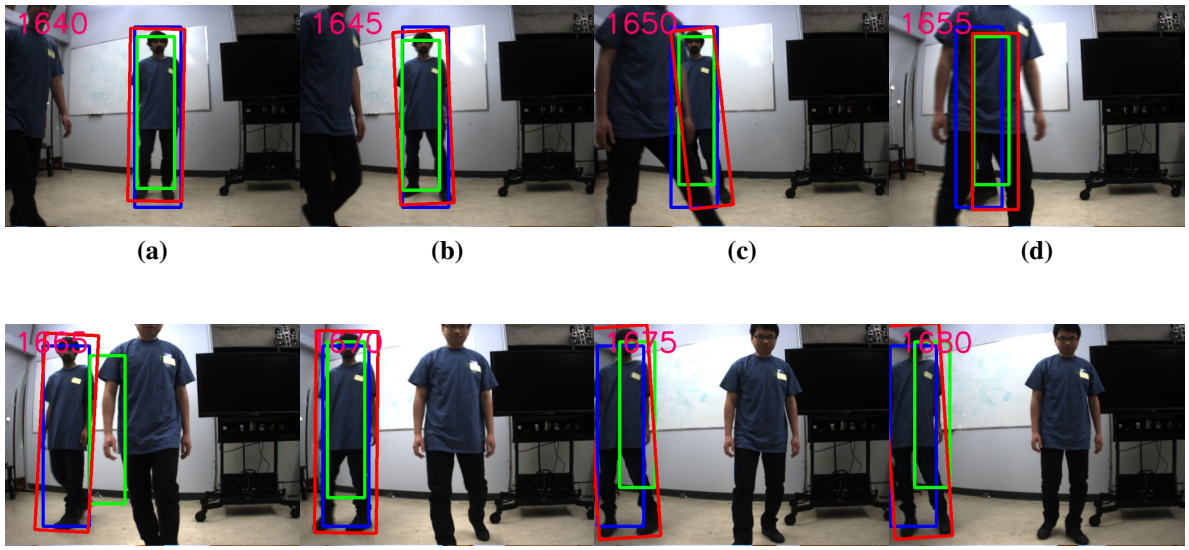
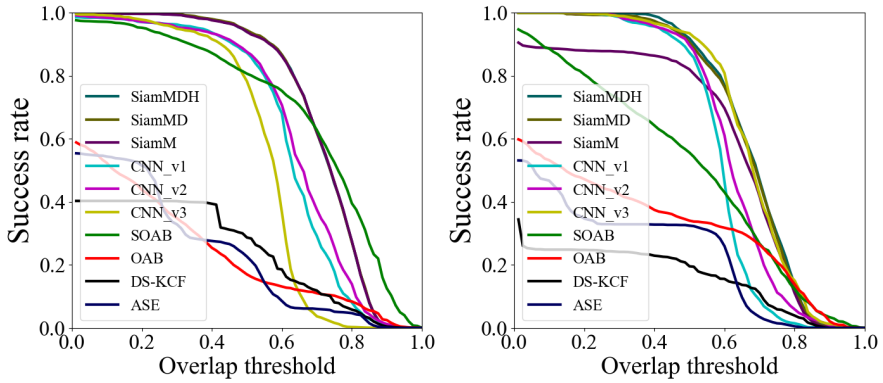


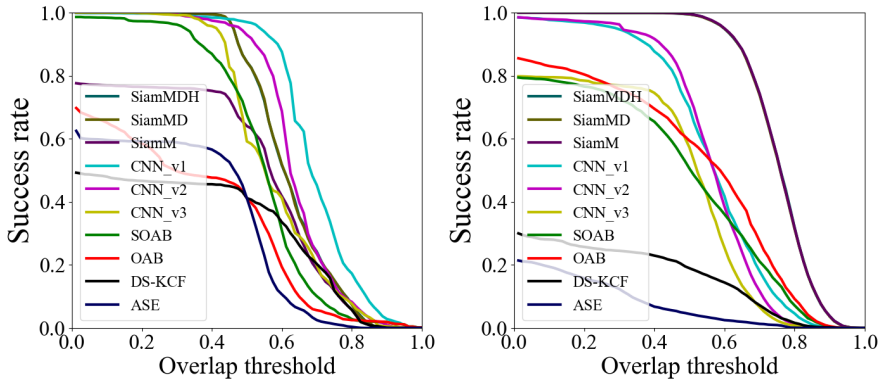
Figure 4.5: Sample tracking results on sequence *Same Clothes 2* when the target is completely occluded by a similar clothed person. The blue box is the ground truth, the red box is SiamMDH, and the green box is CNN_RGBSD.



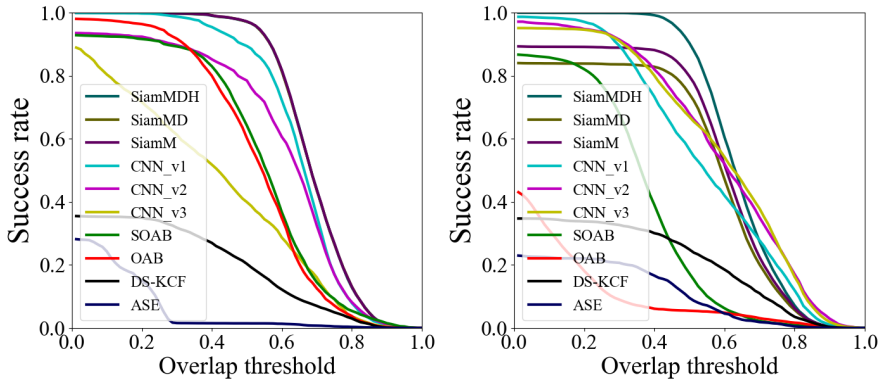
(a) Hallway 1 **(b)** Multi-crossings

(c) Same Clothes 1 **(d)** Lecture Hall

Figure 4.6: *Success-plots*: comparison between our trackers and different tracking algorithms on our dataset. The evaluation is based on the area under the curve. SiamM is the short form for SiamMask [76], SOAB [15], OAB [57], ASE [58], DS-KCF [59]

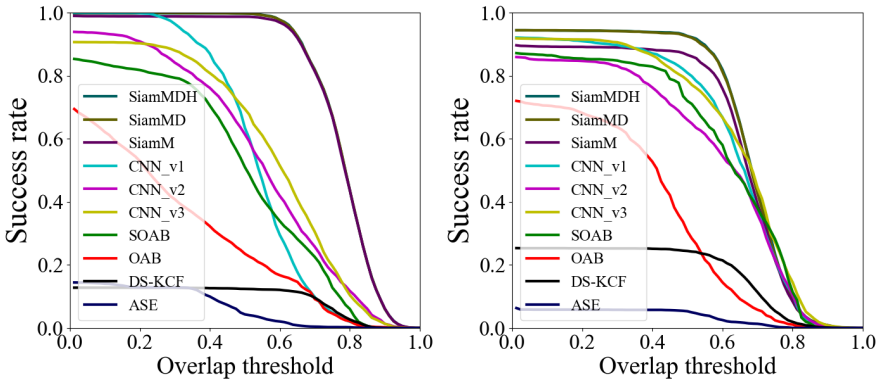


(a) Same Clothes 2 **(b)** Long Corridor

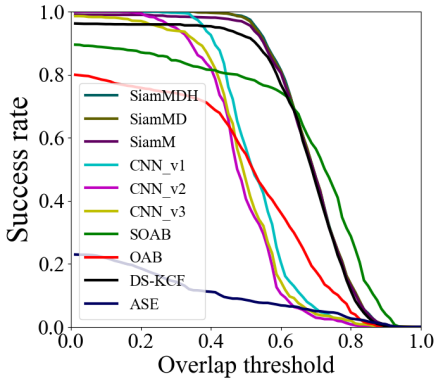


(c) Walking Outdoor **(d)** Sidewalk

Figure 4.7: *Success-plots*: comparison between our trackers and different tracking algorithms on our dataset. The evaluation is based on the area under the curve. SiamM is the short form for SiamMask [76], SOAB [15], OAB [57], ASE [58], DS-KCF [59]



(a) Lab & Seminar **(b)** Corridor Corners



(c) Hallway 2

Figure 4.8: *Success-plots*: comparison between our trackers and different tracking algorithms on our dataset. The evaluation is based on the area under the curve. SiamM is the short form for SiamMask [76], SOAB [15], OAB [57], ASE [58], DS-KCF [59]

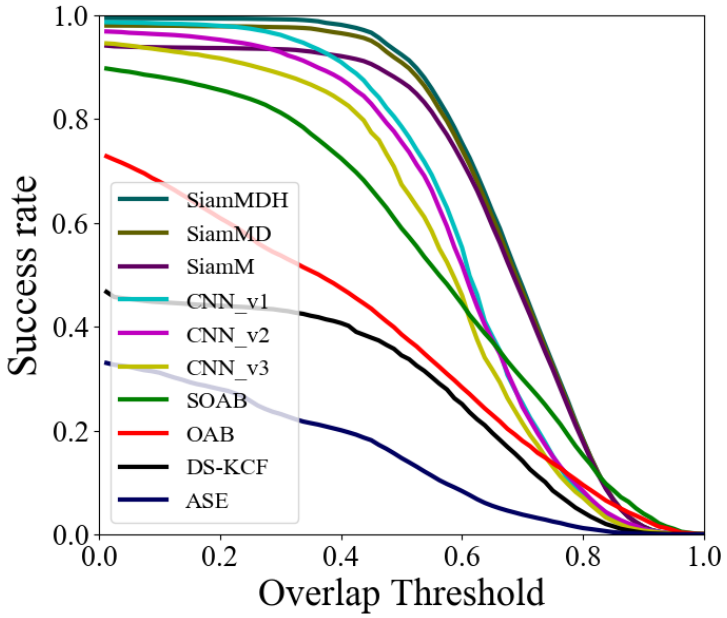


Figure 4.9: *Success-plot*: comparison between our trackers and different tracking algorithms on our dataset with 11 sequences for 10 different trackers. The evaluation is based on the area under the curve. SiamM is the short form for SiamMask [76], SOAB [15], OAB [57], ASE [58], DS-KCF [59]

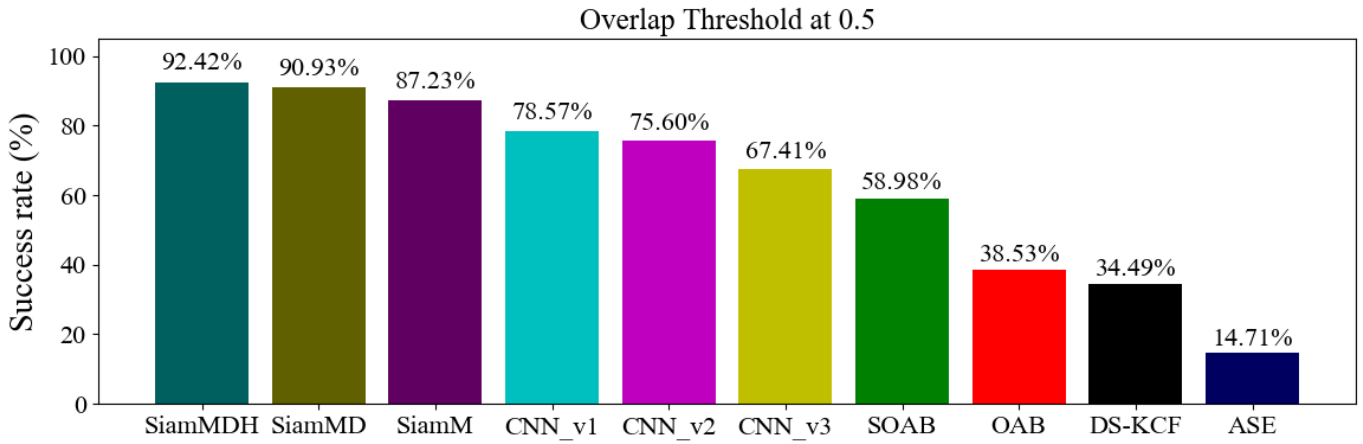
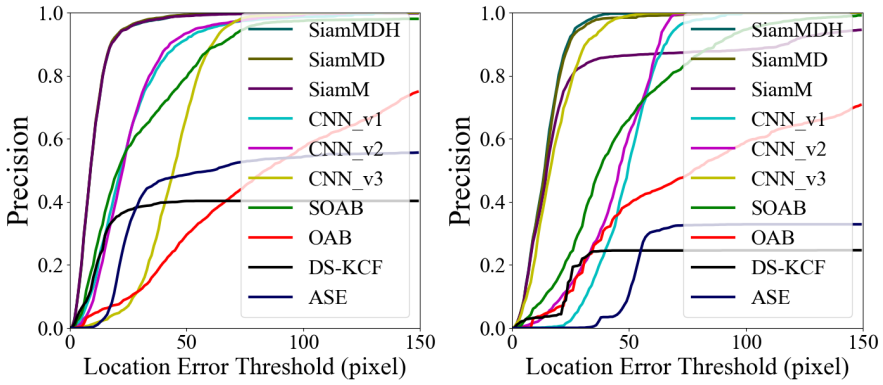
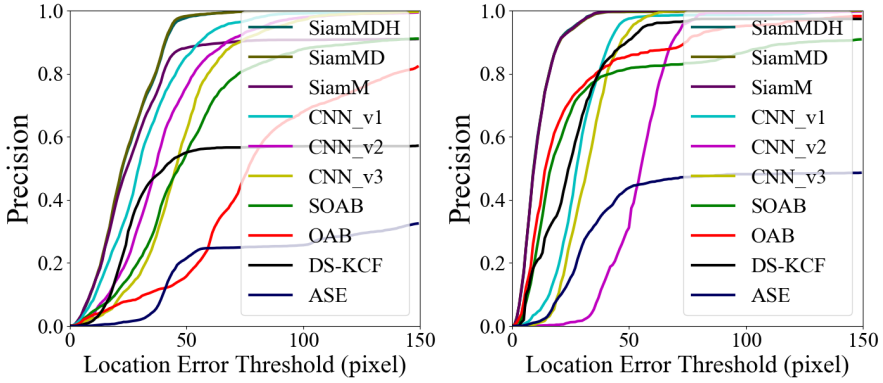


Figure 4.10: Success rate at overlap threshold 0.5: comparison between our trackers and different tracking algorithms on our dataset with 11 sequences for 10 different trackers. SiamM is the short form for SiamMask [76], SOAB [15], OAB [57], ASE [58], DS-KCF [59]

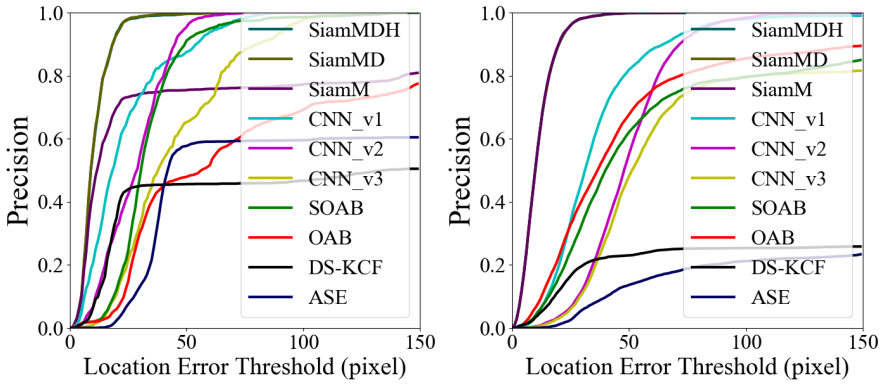


(a) Hallway 1 **(b)** Multi-crossings

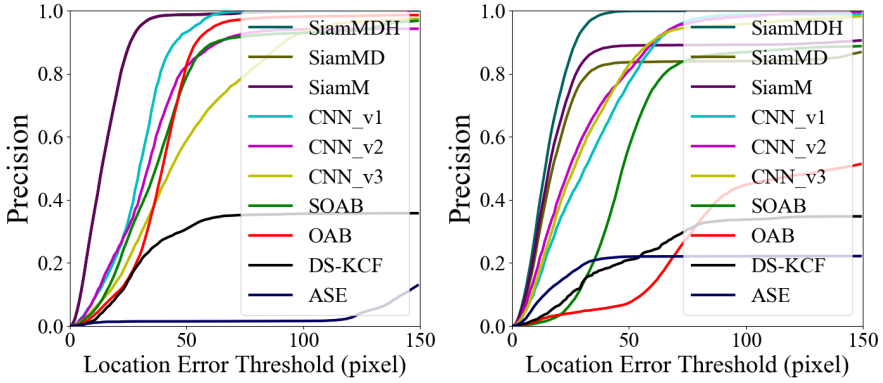


(c) Same Clothes 1 **(d)** Lecture Hall

Figure 4.11: *Precision-plots*: comparison between our trackers and different tracking algorithms on our dataset. The evaluation is based on the area under the curve. SiamM is the short form for SiamMask [76], SOAB [15], OAB [57], ASE [58], DS-KCF [59]

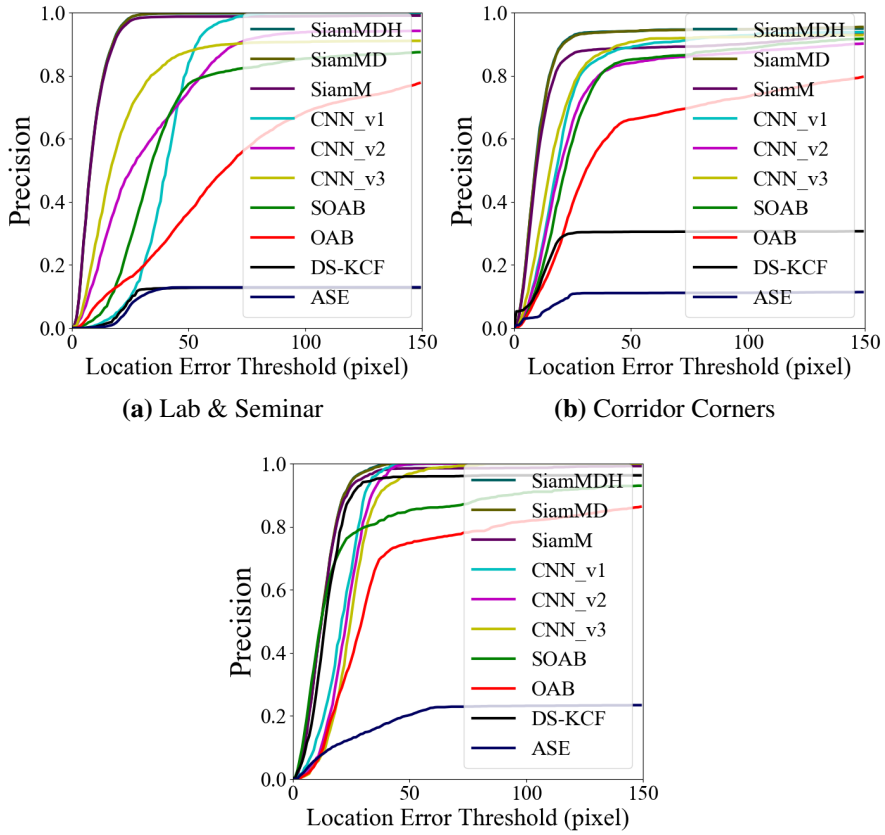


(a) Same Clothes 2 **(b)** Long Corridor



(c) Walking Outdoor **(d)** Sidewalk

Figure 4.12: *Precision-plots*: comparison between our trackers and different tracking algorithms on our dataset. The evaluation is based on the area under the curve. SiamM is the short form for SiamMask [76], SOAB [15], OAB [57], ASE [58], DS-KCF [59]



(a) Lab & Seminar

(b) Corridor Corners

(c) Hallway 2

Figure 4.13: *Precision-plots*: comparison between our trackers and different tracking algorithms on our dataset. The evaluation is based on the area under the curve. SiamM is the short form for SiamMask [76], SOAB [15], OAB [57], ASE [58], DS-KCF [59]

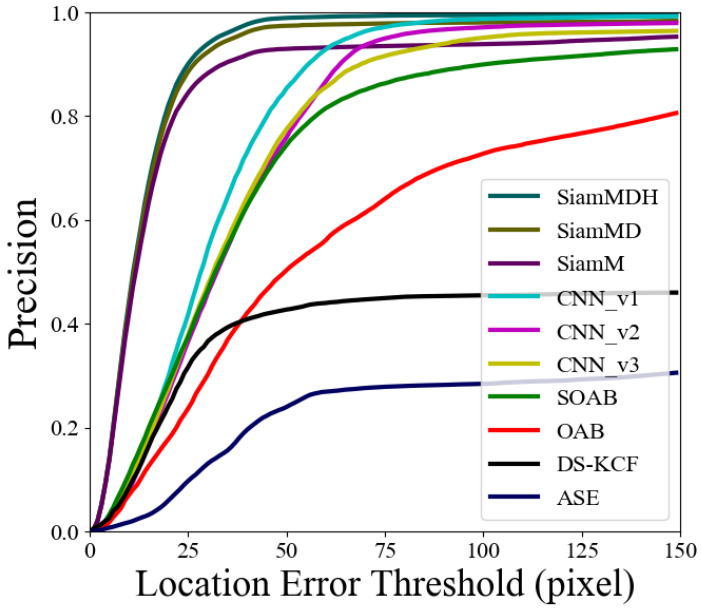


Figure 4.14: *Precision-plot*: comparison between our trackers and different tracking algorithms on our dataset with 11 sequences for 10 different trackers. The evaluation is based on the area under the curve. SiamM is the short form for SiamMask [76], SOAB [15], OAB [57], ASE [58], DS-KCF [59]

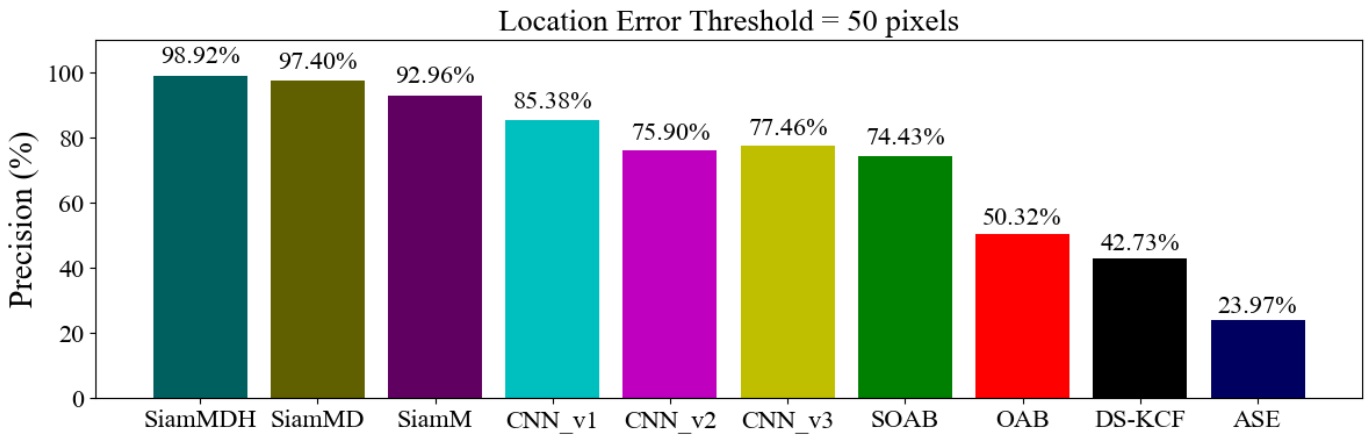


Figure 4.15: Precision at location error threshold 50 pixels: comparison between our trackers and different tracking algorithms on our dataset with 11 sequences for 10 different trackers. The evaluation is based on the area under the curve. SiamM is the short form for SiamMask [76], SOAB [15], OAB [57], ASE [58], DS-KCF [59]

Chapter 5

Conclusion

5.1 Summary

This thesis contains three different online human tracking algorithms: (i) Modified version of OAB (SOAB), (ii) CNN tracker using Online Learning, and (iii) Modified version of SiamMask (SiamMDH). SOAB has a decent performance in terms of tracking accuracy, and the algorithm can be deployed on the computer systems without a high-end GPU. In contrast, SiamMDH is a pretrained Siamese based convolutional neural network that requires a moderate GPU to be able to tracking a human target in real-time, but it out performed the other two trackers (SOAB and CNN_RGBSD). In addition, we also developed a novel human tracking dataset with stereo images. To the best of our knowledge, this dataset is the first dataset for human tracking on a mobile robotic platform. The dataset consists of a variety of different challenging

situations, e.g., pose changes, illumination changes, appearance changes, partial and complete occlusions, similar clothed persons, etc. The dataset is open to the public and posted on our webpage [3].

5.2 Future Work

There are many kinds of improvements that can be made for human tracking on a mobile robot.

Motion model: Wang *et al.* [90] proposed an algorithm using optical flow to estimate the target trajectory to guides finding an accurate tracking region. But, this method is not efficient enough to run on a mobile platform. A real-time motion estimation model is needed to solve the run-time problem under the situation such that the camera is not steady.

Path planning: To increase the robustness of the following behaviors, a path planning algorithm can be used to avoid obstacles. An efficient motion planning and obstacle avoidance algorithm has been implemented in [91] and [92]. Further integration of the system is still needed.

Hardware: In Chapter 4, we described a tracking algorithm that outputs masks on the human target. Instead of the stereo camera, we could replace it with a monocular camera to estimate the distance between the target and the robot by the size of the bounding box instead of a stereo camera. But this

approach provides a perceptual depth rather than the accurate depth. Although we have not tested this approach, we would recommend that it be tried since using monocular cameras will simplify the hardware complexity and provide an economical solution. Another approach to calculating the depth is to use a laser sensor. A laser sensor could prevent accidents since it is isolated from the camera.

Bounding box optimization: In [49], an efficient bounding box optimization has been described to improve the *IoU* (*Success-Plots*) accuracy. Since the *Success-Plot* is our secondary evaluation metric, we did not implement the optimization algorithm to increase the system complexity. So, we do not recommend to integrate the bounding box optimization algorithm for the purpose of the human tracking with a stereo camera. But, for the circumstance with a monocular camera only, bounding box optimization is necessary.

5.3 Conclusion

In conclusion, this thesis proposed and evaluated three novel tracking algorithms (SOAB, CNN_RGBSD, and SiamMDH) that can track a human target under a variety of challenging situations (Full occlusion, Appearance changes, Pose changes, Illumination changes, Similarly clothed instances, Moving out of the view and re-entering, etc.). As well as, an extensive person-following

dataset is built for evaluating the tracking algorithms.

Bibliography

[1] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

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

[4] Shuran Song and Jianxiong Xiao. Tracking revisited using rgbd camera:

Unified benchmark and baselines. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 233–240. IEEE, 2013.

[5] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[6] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[7] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[8] Xiaowei Zhou, Menglong Zhu, Georgios Pavlakos, Spyridon Leonardos, Konstantinos G Derpanis, and Kostas Daniilidis. Monocap: Monocular human motion capture using a cnn coupled with a geometric prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[9] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and

Kostas Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1263–1272. IEEE, 2017.

[10] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[11] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stereoscopic neural style transfer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[12] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 963–968. Ieee, 2011.

[13] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[14] Bao Xin Chen, Raghavender Sahdev, and John K Tsotsos. Integrating stereo vision with a cnn tracker for a person-following robot. In *International Conference on Computer Vision Systems*, pages 300–313. Springer, 2017.

[15] Bao Xin Chen, Raghavender Sahdev, and John K. Tsotsos. Person following robot using selected online ada-boosting with stereo camera. In *Computer and Robot Vision (CRV), 2017 14th Conference on*, pages 48–55. IEEE, 2017.

[16] Markus D Solbach and John K Tsotsos. Vision-based fallen person detection for the elderly. *CoRR, abs/1707.07608*, 2017.

[17] Simin Wang, Salim Zabir, and Bastian Leibe. Lying pose recognition for elderly fall detection. *Robotics: Science and Systems VII*, 345, 2012.

[18] Raghavender Sahdev and John K Tsotsos. Indoor place recognition system for localization of mobile robots. In *Computer and Robot Vision (CRV), 2016 13th Conference on*, pages 53–60. IEEE, 2016.

[19] Yi Hou, Hong Zhang, and Shilin Zhou. Evaluation of object proposals and convnet features for landmark-based visual place recognition. *Journal of Intelligent & Robotic Systems*, pages 1–16, 2017.

[20] Bao Xin Chen, Raghavender Sahdev, Dekun Wu, Xing Zhao, Manos Papagelis, and John K Tsotsos. Scene classification in indoor environments for robots using context based word embeddings. *arXiv preprint arXiv:1908.06422*, 2019.

[21] Amir Rasouli and John K Tsotsos. Sensor planning for 3d visual search with task constraints. In *Computer and Robot Vision (CRV), 2016 13th Conference on*, pages 37–44. IEEE, 2016.

[22] Alper Aydemir, Kristoffer Sjöö, John Folkesson, Andrzej Pronobis, and Patric Jensfelt. Search in the real world: Active visual object search based on spatial relations. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2818–2824. IEEE, 2011.

[23] Raghavender Sahdev, Bao Xin Chen, and John K Tsotsos. Indoor localization in dynamic human environments using visual odometry and global pose refinement. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 360–367. IEEE, 2018.

[24] Ji Zhang and Sanjiv Singh. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2174–2181. IEEE, 2015.

[25] Raghavender Sahdev. Free space estimation using occupancy grids and dynamic object detection. *arXiv preprint arXiv:1708.04989*, 2017.

[26] Ching-Heng Ku, Wen-Hsiang Tsai, et al. Smooth vision-based autonomous land vehicle navigation in indoor environments by person fol-

lowing using sequential pattern recognition. *Journal of Robotic Systems*, 16(5):249–262, 1999.

[27] Maurizio Piaggio, Roberto Fornaro, Alberto Piombo, Luca Sanna, and Renato Zaccaria. An optical-flow person following behaviour. In *Intelligent Control (ISIC), 1998. Held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), Intelligent Systems and Semiotics (ISAS), Proceedings*, pages 301–306. IEEE, 1998.

[28] Tsuyoshi Yamane, Yoshiaki Shirai, and Jun Miura. Person tracking by integrating optical flow and uniform brightness regions. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 4, pages 3267–3272. IEEE, 1998.

[29] Giorgio Chivilò, Flavio Mezzaro, Antonio Sgorbissa, and Renato Zaccaria. Follow-the-leader behaviour through optical flow minimization. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3182–3187. IEEE, 2004.

[30] David Beymer and Kurt Konolige. Tracking people from a mobile platform. In *Experimental robotics VIII*, pages 234–244. Springer, 2003.

[31] M Tarokh and P Ferrari. Case study: Robotic person following using fuzzy control and image segmentation. *Journal of Field Robotics*, 20(9):557–568, 2003.

[32] Takashi Yoshimi, Manabu Nishiyama, Takafumi Sonoura, Hideichi Nakamoto, Seiji Tokura, Hirokazu Sato, Fumio Ozaki, Nobuto Matsuhira, and Hiroshi Mizoguchi. Development of a person following robot with vision based target detection. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5286–5291. IEEE, 2006.

[33] Daniele Calisi, Luca Iocchi, and Riccardo Leone. Person following through appearance models and stereo vision using a mobile robot. In *VISAPP (Workshop on on Robot Vision)*, pages 46–56, 2007.

[34] Zhichao Chen and Stanley T Birchfield. Person following with a mobile robot using binocular feature-based tracking. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 815–820. IEEE, 2007.

[35] Hiroshi Takemura, Keita Ito, and Hiroshi Mizoguchi. Person following mobile robot under varying illumination based on distance and color

information. In *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on*, pages 1500–1505. IEEE, 2007.

[36] Junji Satake and Jun Miura. Robust stereo-based person detection and tracking for a person following robot. In *ICRA Workshop on People Detection and Tracking*, pages 1–10, 2009.

[37] Mahmoud Tarokh and Paulo Merloti. Vision-based robotic person following under light variations and difficult walking maneuvers. *Journal of Field Robotics*, 27(4):387–398, 2010.

[38] Mahmoud Tarokh and Ranjitha Shenoy. Vision-based robotic person following in fast walking. In *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, pages 3172–3177. IEEE, 2014.

[39] Junji Satake, Masaya Chiba, and Jun Miura. A sift-based person identification using a distance-dependent appearance model for a person following robot. In *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, pages 962–967. IEEE, 2012.

[40] Masashi Awai, Takahito Shimizu, Toru Kaneko, Atsushi Yamashita, and H. Asama. Hog-based person following and autonomous returning using generated map by mobile robot equipped with camera and laser range

finder. In *Intelligent Autonomous Systems 12*, pages 51–60. Springer, 2013.

[41] Masashi Awai, Atsushi Yamashita, Takahito Shimizu, T. Kaneko, Y. Kobayashi, and H. Asama. Development of mobile robot system equipped with camera and laser range finder realizing hog-based person following and autonomous returning. *Journal ref: Journal of Robotics and Mechatronics*, 26(1):68–77, 2014.

[42] K Koide and J Miura. Identification of a specific person using color, height, and gait features for a person following robot. *Robotics and Autonomous Systems*, 84:76–87, 2016.

[43] Youngwoo Yoon, Woo-han Yun, Hosub Yoon, and Jaehong Kim. Real-time visual target tracking in rgb-d data for person-following robots. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 2227–2232. IEEE, 2014.

[44] Youngwoo Yoon, Hosub Yoon, and Jaehong Kim. Depth assisted person following robots. In *RO-MAN, 2013 IEEE*, pages 330–331. IEEE, 2013.

[45] Guillaume Doisy, Aleksandar Jevtic, Eric Lucet, and Yael Edan. Adaptive person-following algorithm based on depth images and mapping. In *Proc. of the IROS Workshop on Robot Motion Planning*, 2012.

[46] Max Bajracharya, Baback Moghaddam, Andrew Howard, Shane Brennan, and Larry H Matthies. A fast stereo-based system for detecting and tracking pedestrians from a moving vehicle. *The International Journal of Robotics Research*, 28(11-12):1466–1485, 2009.

[47] Akansel Cosgun, Dinei A Florencio, and Henrik I Christensen. Autonomous person following for telepresence robots. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4335–4342. IEEE, 2013.

[48] Masayuki Kanbara, Takashi Okuma, Haruo Takemura, and Naokazu Yokoya. Real-time composition of stereo images for video see-through augmented reality. In *Multimedia Computing and Systems, 1999. IEEE International Conference on*, volume 1, pages 213–219. IEEE, 1999.

[49] Bao Xin Chen and John K Tsotsos. Fast visual object tracking with rotated bounding boxes. *arXiv preprint arXiv:1907.03892*, 2019.

[50] Cha Zhang and Zhengyou Zhang. Boosting-based face detection and adaptation. *Synthesis Lectures on Computer Vision*, 2(1):1–140, 2010.

[51] Alina Beygelzimer, Satyen Kale, and Haipeng Luo. Optimal and adaptive algorithms for online boosting. In *25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*, pages 4120–4124, 2016.

[52] Yun Sha and Guo-ying Zhang. An adaptive weighted boosting algorithm for road detection. In *Networking, Sensing and Control (ICNSC), 2010 International Conference on*, pages 582–586. IEEE, 2010.

[53] Kham Nguyen, Tim Ng, and Long Nguyen. Adaptive boosting features for automatic speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4733–4736. IEEE, 2012.

[54] 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.

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

[56] Jianyu Wang, Xilin Chen, and Wen Gao. Online selecting discriminative tracking features using particle filter. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 1037–1042. IEEE, 2005.

[57] Helmut Grabner, Michael Grabner, and Horst Bischof. Real-time tracking via on-line boosting. In *Proceedings of the British Machine Vision Conference 2006, Edinburgh*, pages 47–56, 2006.

[58] 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.

[59] Massimo Camplani, Sion L Hannuna, Majid Mirmehdi, Dima Damen, Adeline Paiement, Lili Tao, and Tilo Burghardt. Real-time rgb-d tracking with depth scaling kernelised correlation filters and occlusion handling. In *British Machine Vision Conference, Swansea, UK, September 7-10, 2015*. BMVA Press, 2015.

[60] 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.

[61] Faegheh Sardari and Mohsen Ebrahimi Moghaddam. A hybrid occlusion free object tracking method using particle filter and modified galaxy based search meta-heuristic algorithm. *Applied Soft Computing*, 50:280–299, 2017.

[62] Changxin Gao, Huizhang Shi, Jin-Gang Yu, and Nong Sang. Enhancement of elda tracker based on cnn features and adaptive model update. *Sensors*, 16(4):545, 2016.

[63] Yang Hua, Karteek Alahari, and Cordelia Schmid. Online object tracking with proposal selection. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[64] Mengyao Zhai, Mehrsan Javan Roshtkhari, and Greg Mori. Deep learning of appearance models for online object tracking. *arXiv preprint arXiv:1607.02568*, 2016.

[65] Shaul Oron, Aharon Bar-Hillel, Dan Levi, and Shai Avidan. Locally orderless tracking. *International Journal of Computer Vision*, 111(2):213–228, 2015.

[66] 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.

[67] Lu Zhang and Laurens van der Maaten. Structure preserving object tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1838–1845, 2013.

[68] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang. Real-time object tracking via online discriminative feature selection. *IEEE Transactions on Image Processing*, 22(12):4664–4677, 2013.

[69] 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.

[70] 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.

[71] 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.

[72] Changxin Gao, Feifei Chen, Jin-Gang Yu, Rui Huang, and Nong Sang. Robust visual tracking using exemplar-based detectors. *IEEE Transactions on Circuits and Systems for Video Technology*, 2015.

[73] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.

[74] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, and Wolfram Burgard. Multimodal deep learning for robust rgb-

d object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 681–687. IEEE, 2015.

[75] Camille Couprie, Clément Farabet, Laurent Najman, and Yann Lecun. Indoor semantic segmentation using depth information. In *International Conference on Learning Representations (ICLR2013), April 2013*, 2013.

[76] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. *arXiv preprint arXiv:1812.05050*, 2018.

[77] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[78] 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.

[79] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition, pages 4834–4843, 2018.

[80] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[81] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[82] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8971–8980, 2018.

[83] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. *arXiv preprint arXiv:1812.11703*, 2018.

[84] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. *arXiv preprint arXiv:1812.05050*, 2018.

[85] Burak Uzkent and YoungWoo Seo. Enkcf: Ensemble of kernelized correlation filters for high-speed object tracking. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1133–1141. IEEE, 2018.

[86] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *European conference on computer vision*, pages 702–715. Springer, 2012.

[87] Michael J Swain and Dana H Ballard. Indexing via color histograms. In *Active Perception and Robot Vision*, pages 261–273. Springer, 1992.

[88] Patrick Sebastian, Yap Vooi Voon, and Richard Comley. The effect of colour space on tracking robustness. In *2008 3rd IEEE Conference on Industrial Electronics and Applications*, pages 2512–2516. IEEE, 2008.

[89] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[90] Jianren Wang, Yihui He, Xiaobo Wang, Xinjia Yu, and Xia Chen. Prediction-tracking-segmentation. *CoRR*, abs/1904.03280, 2019.

[91] Javier Minguez, Florant Lamiraux, and Jean-Paul Laumond. Motion

planning and obstacle avoidance. In *Springer handbook of robotics*, pages 1177–1202. Springer, 2016.

[92] Xueling Luo, Dan Zhang, and Xiaodong Jin. A real-time moving target following mobile robot system with depth camera. In *IOP Conference Series: Materials Science and Engineering*, volume 491, page 012004. IOP Publishing, 2019.