Robust Single Object Tracking and Following by Fusion Strategy

Alejandro Olivas[a], Miguel Ángel Muñoz-Bañón[b], Edison Velasco[c] and Fernando Torres[d]
Group of Automation, Robotics and Computer Vision, University of Alicante, Alicante, Spain

Keywords: Target Tracking, Autonomous Vehicles, Robot Sensing Systems, Real-Time Systems.

Abstract: Single Object Tracking methods are yet not robust enough because they may lose the target due to occlusions or changes in the target's appearance, and it is difficult to detect automatically when they fail. To deal with these problems, we design a novel method to improve object tracking by fusing complementary types of trackers, taking advantage of each other's strengths, with an Extended Kalman Filter to combine them in a probabilistic way. The environment perception is performed with a 3D LiDAR sensor, so we can track the object in the point cloud and also in the front-view image constructed from the point cloud. We use our tracker-fusion method in a mobile robot to follow pedestrians, also considering the dynamic obstacles in the environment to avoid them. We show that our method allows the robot to follow the target accurately during long experimental sessions where the trackers independently fail, demonstrating the robustness of our tracker-fusion strategy.

1 INTRODUCTION

In the last years, Single Object Tracking (SOT) has been a hot research topic. The task of following an object over time is still challenging because of the changes in its appearance and in the environment. The use of Machine Learning (ML) methods has increased the accuracy and speed of the trackers, making it viable to use them in real-time applications. The current problems associated with SOT are losses of the target due to occlusions or misunderstandings with similar objects.

There is a trend in using mobile robots for collaborative tasks with people. In this context, we propose the use of pedestrian tracking for the design of following behaviour. There has been a lot of research with the aim of making a robust pedestrian follower robot (Yoshimi et al., 2006; Koide and Miura, 2016). Besides tracking the target robustly, the robot has to follow the target while avoiding dynamic obstacles for safety reasons, a fact not usually considered in the literature.

For many years, researchers have dealt with SOT in images, solving the problem with different types of approaches. One of them is *tracking-by-detection*, which uses first a detection algorithm and then an association method to match the target across the frames. The shortcomings of *tracking-by-detection* are the limitations of the detector, which may produce false positives or missed detections, and failures in the target re-identification that produces incorrect object associations.

The most trending approach in the last years is the use of *deep learning* algorithms (Zhang et al., 2021). Among these, the Siamese Neural Networks (SNN) (Tao et al., 2016) highlight because of their outperformance in terms of speed and accuracy. These networks are trained in an end-to-end manner, feeding the network with a template of the object and the current frame and retrieving the target's current bounding box. Using the obtained prediction as the template for the next frame, the target is tracked over time. The SNNs have difficulty differentiating between similar objects and recent research has focused on this problem.

There are many approaches for detecting and tracking objects in point clouds because of the use of LiDAR sensors to perceive the environment. The use of geometric information may help in being aware of the occlusions of the target, but the increase of the search space makes the problem heavier computationally. Recently, 3D SNNs have appeared to track single objects in real time, but increasing robustness and accuracy remains a challenge.

For tracking pedestrians, a recent work (He et al., 2021) uses a Convolutional Neural Network (CNN)

[a] https://orcid.org/0000-0002-3462-0713
[b] https://orcid.org/0000-0002-3220-2286
[c] https://orcid.org/0000-0003-2837-2001
[d] https://orcid.org/0000-0002-6261-9939

Olivas, A., Muñoz-Bañón, M., Velasco, E. and Torres, F.
Robust Single Object Tracking and Following by Fusion Strategy.
DOI: 10.5220/0012178900003543
In *Proceedings of the 20th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2023) - Volume 1*, pages 624-631
ISBN: 978-989-758-670-5; ISSN: 2184-2809

to detect the person and a SNN for tracking him. This approach combines different *deep learning* methods to track the pedestrian, but we prefer to combine different trackers in a probabilistic way, as they are complementary. For this reason, we present a tracker-fusion method that integrates a tracker based on *tracking-by-detection*, another based on SNNs and another on point cloud tracking. To do this, we use an Extended Kalman Filter (EKF) to consider the three trackers and coordinate them to overcome their failures. Compared with the other trackers in several experiments, our method tracks the target more robustly and precisely.

There is much research on obstacle avoidance and this problem is not in the scope of this paper, as it is focused on the SOT problem. We use a local planner based on (Muñoz-Bañón et al., 2022).

To summarize, the contributions of our work are:

1) A tracker-fusion strategy to combine heterogeneous approaches to increase the robustness and accuracy of SOT. Our method helps to automatically detect when a tracker misses and corrects it.
2) The integration of our tracker-fusion method with a local planner in an autonomous vehicle, following the target correctly while avoiding dynamic obstacles.
3) A comparison between the trackers and our approach, showing that our method outperforms them.

The rest of this paper is organized as follows. Section 2 describes the state of the art of the SOT problem. Section 3 explains the designed trackers and our tracker-fusion method. Section 4 presents the experimental results, comparing our method to the selected trackers. Finally, in Section 5, the conclusions are exposed.

2 RELATED WORK

Our method fuses a visual *tracking-by-detection* tracker, a visual tracker based on SNNs and a 3D point cloud tracker. This section shows a brief review of these topics in the literature.

Visual *tracking-by-detection*. For people tracking, there are approaches that combine a pedestrian detector with a data association and filtering algorithm, such as a particle filter (Breitenstein et al., 2011). (Andriluka et al., 2008) present an approach that combines *tracking-by-detection* with *detection-by-tracking*, a paradigm that takes advantage of the tracking information to improve the detections, in a framework to track multiple people in situations with occlusions. To re-identify the person, clothing appearance is typically used, and many descriptors have been proposed based on global or local features (Satta, 2013). This type of approach has become more robust and accurate thanks to the use of detectors based on CNNs. The You Only Look Once (YOLO) architecture (Redmon et al., 2016) deals with object detection in a fast and precise way, and it has been a hot research topic in recent years.

Siamese Neural Networks. SNNs track a single object by similarity comparison strategy. Because of the efficiency in terms of speed and accuracy, it has drawn the attention of visual tracking researchers (Held et al., 2016; Bertinetto et al., 2016). Many of these approaches have the drawback of discriminating only the foreground from non-semantic background, having difficulties in distinguishing between similar targets. SiamRPN (Li et al., 2018) employs a region proposal after the SNN to formulate the tracking problem as a local detection task. Zhu *et al.* (Zhu et al., 2018) presented DaSiamRPN, which uses a distractor-aware module to track the target between similar ones. (Ondrašovič and Tarábek, 2021) present a comprehensive review of the SNNs, highlighting the main challenges of dealing with distractors (similar interference) and severe occlusions. The review also discusses approaches that address these issues, such as DaSiamRPN, which demonstrate better performance in handling these challenges. However, these problems are still not completely solved.

3D Object Tracking. Early research about object tracking in point cloud focused on RGB-D information, improving the tracking in images by adding information of the depth channel (Song and Xiao, 2013). (Liu et al., 2018) present a method that uses 3D properties to separate objects that might look adjacent in the images to deal with occlusions. *Deep learning* methods are used to track objects in point clouds obtained by LiDAR sensors. (Luo et al., 2018) present a CNN to detect and track multiple objects and forecast their motion. It uses the bird-eye-view (BEV) of the 3D world.

Recent research has adapted SNNs for point clouds (Giancola et al., 2019) to solve the 3D SOT problem. (Qi et al., 2020) presented a method that identifies potential candidates in a 3D search area using the information of the target's template. Then, the target proposal and verification steps are executed inside the net, avoiding an exhaustive 3D search which is time-consuming. M^2-Track (Zheng et al., 2022) is based on motion-centric paradigm instead of the appearance matching paradigm of preview research. The use of a transformer (Hui et al., 2022) to learn a robust cross-correlation between the template and the

search area point clouds achieves state-of-the-art performance for many classes, although for pedestrian tracking M^2-Track shows better results. Pedestrians have many changes in their appearance while moving, so it makes sense that motion paradigm works better for this class.

These types of trackers are complementary as tracking-by-detection keeps better the target's shape, SNNs accurately track the object in short-term situations, and point cloud trackers take geometrical information into account.

3 METHODOLOGY

In this research, we designed a strategy to fuse three different trackers. Two of them are visual trackers, one based on YOLO and the other on DaSiamRPN, and we explain firstly the adaptations for the front-view images of the point cloud projection. These images have four channels: depth, intensity, reflectivity and infrared. Next, we explain the point cloud tracker, which is based on M^2-Track. Then, the combination of the trackers using an EKF is explained.

3.1 YOLO Tracker

The designed YOLO tracker follows the *tracking-by-detection* paradigm, where the detector is a YOLO neural network. Firstly, the pedestrians in the image are detected and the target is selected. We decided to choose the person who is at the front of the robot statically for a parameterized number of seconds. Then, the three trackers are initialized to follow the selected pedestrian.

For the association of the detections with the target, it is needed to define the pedestrians with descriptors. Firstly, the detected person is segmented using the depth channel to remove the background. According to previous research (Páez-Ubieta et al., 2022), when determining the depth of an identified object, its bounding box size is decreased by 40% and the depth is estimated as the mean value of that interior box. The bounding box of a pedestrian has many points of the background, so to filter them we used the median value instead of the mean. Then, considering this value, each candidate is segmented erasing the background and front obstacles.

To build the descriptor, we use histograms to define the pedestrian, in a similar way to color histograms, but in this case with the other three channels detected by the LiDAR, defining the pedestrian mainly by its clothes' appearance. The depth is not used because it is not a representative characteristic

Figure 1: The 5 parts into which the person is divided in order to calculate the descriptor.

of the person, as it only depends on the distance to the sensor. Usually, the clothes of the top part of the body are different from the ones of the bottom, so we divided the person into five parts as shown in Figure 1, although the bottom part is not considered because it has undesired ground points. For each part, three histograms are calculated, one for each channel, and the descriptor is built concatenating them. There are many possibilities to get a descriptor from a person, but we decided to use this because it has to be fast to build and compare to work in a real-time application while using other trackers.

During the tracking, YOLO detects the pedestrians of the image, and the ones inside a search area are considered candidates. The search area is explained in Section 3.4, and it is defined by the position and covariance of the target estimated by the EKF. The candidates' descriptors are calculated and compared to the target's descriptor, using the histogram correlation defined by (1), where H_1, H_2 are the histograms.

$$d(H_1,H_2) = \frac{\sum_K (H_1(K)-\overline{H}_1)(H_2(K)-\overline{H}_2)}{\sqrt{\sum_K (H_1(K)-\overline{H}_1)^2 \sum_K (H_2(K)-\overline{H}_2)^2}} \quad (1)$$

where,

$$\overline{H}_i = \frac{1}{N}\sum_J H_i(J) \quad (2)$$

and N is the total number of histogram bins.

The most similar candidate to the target is considered the target in the current frame if the correlation is above a minimum threshold. Finally, the target's descriptor is updated as the mean value between the previous and the new descriptor, because it changes when the target or the robot moves.

Among the YOLO architectures, we decided to use YOLOv5 (Jocher et al., 2022) to build our model. A dataset with 2392 LiDAR images with labelled people was created and we trained models using a different combination of the four channels of the LiDAR images. The best results were obtained with all the channels. Problems that affect this tracker are high changes in the target's descriptor or people with similar clothes to the target, but they are mitigated by our tracker-fusion method.

3.2 Siamese Tracker

For tracking, a SNN receives as input the current frame and the bounding box of the target in the previous frame and it returns directly the bounding box in the current frame. We decided to use DaSiamRPN (Zhu et al., 2018) because, even for the LiDAR images, it works better than other trackers without training a new model. The network is prepared for RGB images, i.e. three channels images, so we use the intensity, reflectivity and infrared channels. This change is done without fine-tuning the model, as it shows that it can follow targets in these images.

The pedestrian can be tracked on the boundaries of the image because the LiDAR has a view of 360º. However, DaSiamRPN is not prepared for this. To adapt the method, we have to add the pixels of the other boundary to the search area when the target is near the image limits. For doing this, the image is shifted as shown in Figure 2 and the target can be followed when it crosses the boundaries of the image.

As this tracker was not trained with LiDAR images, it may miss the target more frequently than in RGB images. Furthermore, as it is designed for general object tracking, the bounding box tends not to focus on the whole person, which increases the error on the position of the target. When fusing the trackers, these problems are overcome.

3.3 Point Cloud Tracker

We decided to use a 3D SNN for tracking the target in the point cloud because they surpass previous methods in terms of speed and accuracy. Among these, we decided to use M^2-Track (Zheng et al., 2022) because, as mentioned in the related work, it outperforms the other networks in pedestrian tracking. We used the model that was trained using the KITTI dataset, although our LiDAR sensor is a different model with more layers, specifically 128 against the 64 of the dataset.

This network receives as input the previous and current point cloud and the 3D bounding box of the target in the previous frame and returns the current 3D bounding box. The computational time of processing the point clouds and predicting the target's current position where larger than expected (more discussion in Section 4) and we take advantage of the search area in the frontal-view image, which will be explained in Section 3.4, to speed up the tracker. Instead of feeding the net with the whole point cloud captured by the LiDAR, only the point cloud inside the search area is sent to the net. Even if it requires a reconstruction from an image, it does not consume much time for a small part of the image and speeds up the tracker.

The drawbacks of this tracker are failures when the target crosses paths with other pedestrians. The computational cost may be a problem for real-time applications when using it with other trackers and we will discuss if the increase of time in the tracker-fusion is worth the gain in robustness and precision in Section 4.

3.4 Tracker Fusion

The diagram of our tracker-fusion strategy is shown in Figure 3. Using the characteristics of the sensor, it is possible to calculate the 3D coordinates of a pixel of the image using its value in the depth channel. With the centre pixel of the bounding box and the depth value estimated by the method explained in Section 3.1, the coordinates of the target are estimated, which are considered from now on as observations of a dynamic system. The centroid of the 3D bounding box predicted by the point cloud tracker is also considered an observation.

The position merging is performed with an EKF, which uses the predict and update phases to estimate the state and the covariance of the target. The state is defined by the position in the X and Y axis because our local planner only needs these two coordinates. In our case, the prediction phase works as a constant propagation model. A new observation updates the state of the EKF, and its covariance represents the expected error of it. We consider that, if the tracker returns a bounding box that is very different or far from the previous one, the error may be larger, as if the tracker failed the target would be around the previous position. Therefore, the covariance of the observation should be higher.

For the visual trackers, the Intersection over Union (IoU) is calculated and the covariance is inversely proportional to it. The IoU is defined as the percentage of the intersection area between both bounding boxes with respect to the union of both areas. Besides, there may be a larger error in the direction between the robot and the target because of errors in the target's depth estimation, so the observation covariance is estimated higher in this direction:

$$R = Rot(arctan(y/x)) \tag{3}$$

$$C = \begin{pmatrix} \alpha \cdot (c + s \cdot (1 - IoU)) & 0 \\ 0 & c + s \cdot (1 - IoU) \end{pmatrix} \tag{4}$$

$$Cov = RCR^T \tag{5}$$

Where x and y are the estimated coordinates for the bounding box with respect to the robot, $Rot(\omega)$ the 2D rotation matrix of the angle ω, c is a constant, s is a variable proportional to the state covariance and

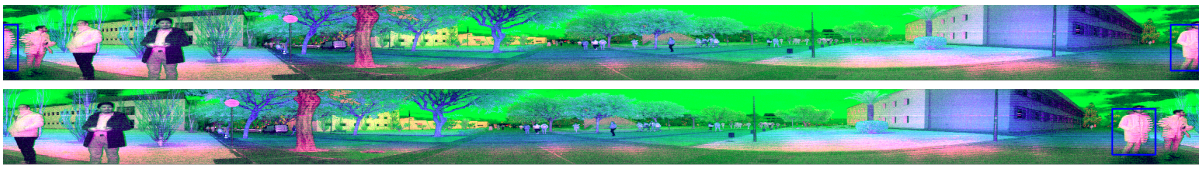


Figure 2: The blue bounding box shows the target, who is cut by the boundaries of the original image (top image). The image is shifted (bottom image) to track the target with DaSiamRPN. The front-view images are pseudo-colorized using the LiDAR channels.

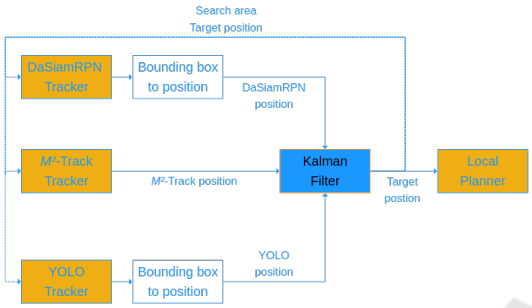


Figure 3: Scheme of our tracker-fusion method.

α is a constant greater than 1 to consider the errors due the estimation of target's depth.

For the point cloud tracker, we consider the size of the 3D bounding box constant, so in this case, the covariance is proportional to the Euclidean distance between the previous position and the current.

A pedestrian cannot move from one position to another in a short period of time. Therefore, we consider a new observation as an outlier if it is far away from the current position. We use the Mahalanobis distance for outlier rejection because it considers the covariance of the EKF state and of the observation.

To coordinate the three trackers, we estimate a search area from the current state and covariance of the EKF. Hence, the size of the search area is dynamically tuned and it is proportional to the covariance, which represents the error of the current position, and therefore the target should be searched inside the area defined by it.

When YOLO does not detect the target in the search area for several seconds, but it detects a person there, the tracker is initialised with that pedestrian. This means that the target has been tracked by the other methods and therefore it is inside the search area. If DaSaimRPN detects the target outside the search area, it is reset with the target detected by YOLO. However, this is not enough to deal with the misses of this tracker, so it is also reset if the bounding boxes of YOLO and DaSiamRPN are very different, i.e. the IoU between the bounding boxes is low, and the YOLO prediction is closer to the last position estimated by EKF. If the point cloud tracker had failed, which is determined by comparing the current position of the target and the predicted position of M^2-Track, M^2-Track is reset using the target's current position.

The target position is sent to the local planner, which computes the trajectory to follow the person while avoiding the obstacles. The planner is based on (Muñoz-Bañón et al., 2022), and the speed of the robot is adjusted with the distance to the target, so when the robot gets close to the target it slows down.

4 EXPERIMENTS

In this section, we show the results of our method. It is compared with the three presented trackers, showing the benefits of fusing them. The experiments were done in a robot equipped with the OS1-128-U LiDAR Sensor-Ouster to perceive the environment and an onboard computer with AMD Ryzen 7 5700G, NVIDIA GTX 3060 (12 GB) and 32 GB of RAM.

About the metrics for comparing the methods, we use the Euclidean distance between the predicted and the ground truth position. The ground truth of the bounding boxes was labelled manually in some frames, and for the other frames, it was estimated as the linear interpolation between the previous labelled frame and the next. Then, the position is estimated from the bounding boxes as explained in Section 3.4. For the visual trackers, the IoU is also used to compare the methods.

The experiments consist in eight sessions where the robot follows different people, using the designed tracker-fusion and the local planner. In the first experimental session, using the DaSiamRPN tracker, the target is lost when it moves near the robot. With the YOLO tracker, when he crosses paths with another person. The two pedestrians have similar clothes with respect to the LiDAR channels and the tracker confuses both after the occlusion. The point cloud tracker also missed the target at that moment. Despite the problems of the trackers when they are used separately, the target is not lost when fusing them. Figure 4 shows the position error during this session.

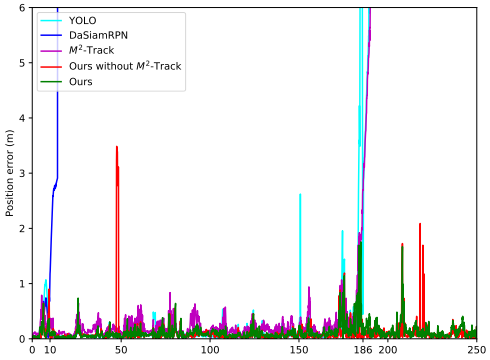


Figure 4: Position error during the first experiment.

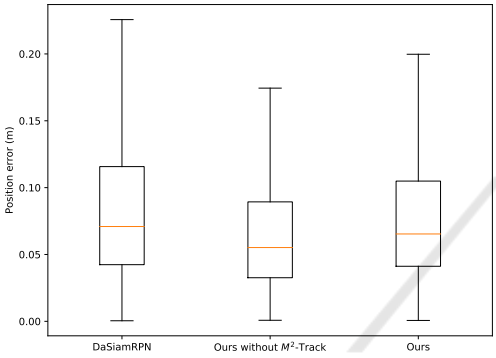
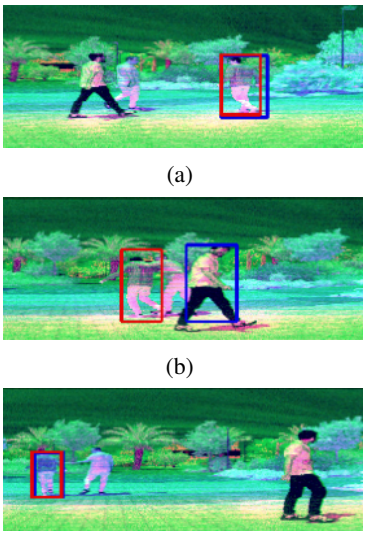


Figure 5: Box plot of the error during the second session. The outliers are not represented for clarity, but the maximum errors were 4.45 m, 1.78 m and 1.86 m respectively.

About the IoU values for the visual trackers when the fusion is done, YOLO tracker has better results with a mean value of 75.7% while DaSiamRPN has 63%. This trend is common in all the sessions because the YOLO bounding box fits better to the pedestrian than the DaSiamRPN bounding box.

In experimental session 2, M^2-Track lost the target when it crossed paths with another pedestrian, while YOLO missed it because variations in illumination caused a large change in the descriptor of the target. DaSiamRPN did not lose the target in this session, although our tracker-fusion method tracked the target more accurately, which is shown in Figure 5.

Table 1 compares the different methods, showing the time when the target is lost, if it happens, and the average position error while tracking the target in the 8 experimental sessions. Using the three trackers in isolation, the target is always lost except in session 2 with DaSiamRPN tracker and in session 6 with YOLO. Despite being more time-consuming than the other trackers, M^2-Track does not show noteworthy robustness and accuracy. Our tracker-fusion without M^2-Track outperforms the version with it, as it succeeded in more sessions.



(a)

(b)

(c)

Figure 6: (a) The trackers detect correctly the target before the occlusion. (b) After the occlusion, the YOLO tracker (red) detects correctly the target, but DaSiamRPN (blue) tracks another person. (c) DaSiamRPN tracker is corrected.

In sessions 6 to 8, there were consecutive occlusions that our tracker-fusion handled, and our method was also robust to occlusions during several seconds and could track the target when it passed from one boundary of the image to the other. Figure 6 shows how the target is restored after an occlusion when one tracker misses. However, in session 5 it failed when the target was far away, around 14 m, and the robot rotated, which produced a high change in the position of the target. Nevertheless, the robot should be close to the target for our application of following, so it is not a big issue.

We commented in Section 3.3 that the computational time of M^2-Track was large for using it in a robotic application. M^2-Track consumes on average 70.5 ms when feeding the net with the whole point cloud, while it consumes 38.3 ms when using only the point clouds of the search area. The point cloud reconstruction is included and is on average 3.2 ms. In the presented results, we used the point cloud of the search area.

About the other components, YOLO tracker consumes on average 11.5 ms and DaSiamRPN 3.7 ms, the process to get the positions from the bounding boxes plus the update of the EKF consumes 0.4 ms and the local planner 10.3 ms. It is needed to reconstruct the point cloud from the image without the target because the local planner does not have to consider it as an obstacle, and this process consumes 4.9 ms. Considering all these times, the method can work at a frequency of 13.8 Hz. Therefore, our method

Table 1: Comparison between the trackers in the 8 sessions, duration of each one in parenthesis. T_L expresses the tracking time until the target is lost (if it happens), and *Error* is the average Euclidean distance between the detected position and the ground truth.

	Session 1 (250 s)		Session 2 (600 s)		Session 3 (138 s)		Session 4 (140 s)	
	T_L (s)	Error (m)	T_L (s)	Error (m)	T_L (s)	Error (m)	T_L (s)	Error (m)
YOLO tracker	186	0.17	66	0.08	5	0.53	77	0.28
DaSiamRPN	10	0.24	-	0.13	136	0.17	81	0.11
M^2-Track	186	0.22	125	0.17	68	0.23	35	0.17
Ours without M^2-Track	-	0.11	-	0.09	-	0.18	121	0.39
Ours	-	0.12	-	0.10	-	0.19	132	0.24

	Session 5 (119 s)		Session 6 (181 s)		Session 7 (172 s)		Session 8 (240 s)	
	T_L (s)	Error (m)	T_L (s)	Error (m)	T_L (s)	Error (m)	T_L (s)	Error (m)
YOLO tracker	38.5	0.10	-	0.15	57.5	0.26	148.5	0.14
DaSiamRPN	102	0.63	43	0.10	40	0.11	170	0.15
M^2-Track	19	0.15	43	0.25	24.5	0.24	75	0.20
Ours without M^2-Track	105	0.33	-	0.15	-	0.23	-	0.10
Ours	61.5	0.47	-	0.16	41	0.12	156.5	013

works in real time, as we process the measurements faster than the frequency at which LiDAR works, 10 Hz.

Although the algorithms were executed offline to compare the different trackers, we also used our tracker fusion to follow the pedestrian while recording the sessions. During sessions 3 and 4, we used the 3 trackers, and the onboard computer could not process the point cloud correctly because of the computational cost, and the images were corrupted as shown in Figure 7. For this reason, our method failed during session 4, and the rest of the experiments were recorded without using M^2-Track.

Therefore, we conclude that M2-Track is not worthwhile for our fusion method with our current hardware. Besides, if it were possible to process the whole point cloud in less time, the point cloud tracker should miss less, as it failed many times when the robot rotated and the point cloud changed rapidly, so the search space was not enough to track the target. Even though, without using M^2-Track, our method is robust enough and can work in real time. The code is shared[1] and there is a video of the experiments[2].

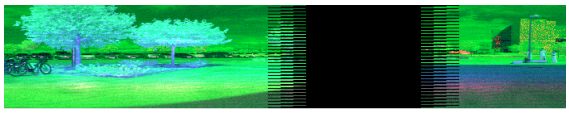


Figure 7: Part of an image from a corrupted point cloud. It has no information about a region of the image.

[1]https://github.com/AUROVA-LAB/aurova_detections

[2]https://youtu.be/n7d_BEgYg2k

5 CONCLUSIONS

In this research, after showing a comparison of pedestrian trackers in long experimental sessions, we demonstrated that fusing them is a promising way to improve accuracy and robustness. Our tracker-fusion method, which uses an EKF to integrate the trackers probabilistically, shows better performance without M^2-Track, only with visual trackers. Consequently, our method can work in real-time applications correctly, as there is no need to use time-consuming point cloud trackers.

We integrate our method in an autonomous vehicle to implement following behavior. Although we focus the research on pedestrian tracking, our method is potentially useful in all tracking applications. In addition, our method can track other dynamic objects, such as vehicles or robots, if the YOLO model is trained for those objects.

ACKNOWLEDGEMENTS

This work has been supported by the Ministry of Science and Innovation of the Spanish Government through the research project PID2021-122685OB-I00 and through the Formación del Personal Investigador [Research Staff Formation (FPI)] under Grant PRE2019-088069.

REFERENCES

Andriluka, M., Roth, S., and Schiele, B. (2008). People-tracking-by-detection and people-detection-by-tracking. In *2008 IEEE Conference on computer vision and pattern recognition*, pages 1–8. IEEE.

Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., and Torr, P. H. (2016). Fully-convolutional siamese networks for object tracking. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II 14*, pages 850–865. Springer.

Breitenstein, M. D., Reichlin, F., Leibe, B., Koller-Meier, E., and Van Gool, L. (2011). Online multiperson tracking-by-detection from a single, uncalibrated camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1820–1833.

Giancola, S., Zarzar, J., and Ghanem, B. (2019). Leveraging shape completion for 3d siamese tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1359–1368.

He, C., Zhang, X., Miao, Z., and Sun, T. (2021). Intelligent vehicle pedestrian tracking based on yolov3 and dasiamrpn. In *2021 40th Chinese Control Conference (CCC)*, pages 4181–4186. IEEE.

Held, D., Thrun, S., and Savarese, S. (2016). Learning to track at 100 fps with deep regression networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 749–765. Springer.

Hui, L., Wang, L., Tang, L., Lan, K., Xie, J., and Yang, J. (2022). 3d siamese transformer network for single object tracking on point clouds. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II*, pages 293–310. Springer.

Jocher, G., Chaurasia, A., Stoken, A., Borovec, J., Kwon, Y., Michael, K., and et. al (2022). ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation.

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

Li, B., Yan, J., Wu, W., Zhu, Z., and Hu, X. (2018). High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8971–8980.

Liu, Y., Jing, X.-Y., Nie, J., Gao, H., Liu, J., and Jiang, G.-P. (2018). Context-aware three-dimensional mean-shift with occlusion handling for robust object tracking in rgb-d videos. *IEEE Transactions on Multimedia*, 21(3):664–677.

Luo, W., Yang, B., and Urtasun, R. (2018). Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577.

Muñoz-Bañón, M. Á., Velasco-Sánchez, E., Candelas, F. A., and Torres, F. (2022). Openstreetmap-based autonomous navigation with lidar naive-valley-path obstacle avoidance. *IEEE Transactions on Intelligent Transportation Systems*, 23(12):24428–24438.

Ondrašovič, M. and Tarábek, P. (2021). Siamese visual object tracking: A survey. *IEEE Access*, 9:110149–110172.

Páez-Ubieta, I., Velasco-Sánchez, E., Puente, S. T., and Candelas, F. A. (2022). Detection and depth estimation for domestic waste in outdoor environments by sensors fusion.

Qi, H., Feng, C., Cao, Z., Zhao, F., and Xiao, Y. (2020). P2b: Point-to-box network for 3d object tracking in point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6329–6338.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Satta, R. (2013). Appearance descriptors for person re-identification: a comprehensive review. *arXiv preprint arXiv:1307.5748*.

Song, S. and Xiao, J. (2013). Tracking revisited using rgbd camera: Unified benchmark and baselines. In *Proceedings of the IEEE international conference on computer vision*, pages 233–240.

Tao, R., Gavves, E., and Smeulders, A. W. (2016). Siamese instance search for tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1420–1429.

Yoshimi, T., Nishiyama, M., Sonoura, T., Nakamoto, H., Tokura, S., Sato, H., Ozaki, F., Matsuhira, N., and Mizoguchi, H. (2006). Development of a person following robot with vision based target detection. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5286–5291. IEEE.

Zhang, Y., Wang, T., Liu, K., Zhang, B., and Chen, L. (2021). Recent advances of single-object tracking methods: A brief survey. *Neurocomputing*, 455:1–11.

Zheng, C., Yan, X., Zhang, H., Wang, B., Cheng, S., Cui, S., and Li, Z. (2022). Beyond 3d siamese tracking: A motion-centric paradigm for 3d single object tracking in point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8111–8120.

Zhu, Z., Wang, Q., Li, B., Wu, W., Yan, J., and Hu, W. (2018). Distractor-aware siamese networks for visual object tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 101–117.