ART & TECHNOLOGY PUBLISHING PTE. LTD.

Progress in Human Computer Interaction

2021 Volume 1, Issue 1

ISSN: 2630-4635

# Real-Time Correction Based on Wheel Odometry to Improve Pedestrian Tracking Performance in Small Mobile Robot

**Jaehun Park[1], Min Sung Ahn[2], Jeakweon Han[3]***

[1]Department of Physics, Hanyang University, Seoul, Republic of Korea

[2]Department of Mechanical and Aerospace Engineering, UCLA, Los Angeles, California, United States of America

[3]Department of Robotics, Hanyang University, Seoul, Republic of Korea

***Corresponding author:** Jeakweon Han, jkhan@hanyang.ac.kr



## Abstract

With the growth in the intelligence of mobile robots, interaction with humans is emerging as a very important issue for mobile robots, and the pedestrian tracking technique following the designated person is adopted in many cases in a way that interacts with humans. Among the existing multi-object tracking techniques for pedestrian tracking, Simple Online and Real-time Tracking (SORT) is suitable for small mobile robots that require real-time processing while having limited computational performance. However, SORT fails to reflect changes in object detection values caused by the movement of the mobile robot, resulting in poor tracking performance. To solve this performance degradation, this paper proposes a more stable pedestrian tracking algorithm by correcting object tracking errors caused by robot movement in real-time using wheel odometry information of a mobile robot and dynamically managing the survival period of the tracker that tracks the object. In addition, the experimental results show that the proposed methodology using data collected from actual mobile robots maintains real-time and has improved tracking accuracy with resistance to the movement of the mobile robot.

## Keywords

Multiple object tracking

Mobile robot

Real-time

## 1. Introduction

Mobile robots that have demonstrated industrial success as logistics robots and delivery robots [1] are becoming increasingly integrated into human life in various forms and ways beyond industrial use. Techniques for human tracking are being adopted in many cases, such as the commercialization of companion robots that follow a single person [2] and the development of robot performance art content where robots mimic human movements. In line with these trends, research on

human tracking with mobile robots is currently actively progressing.

For a robot to follow a person, it must be capable of recognizing each individual as a unique object even when multiple people are present. To achieve this, multiple object tracking (MOT) technology is essential. However, there are several challenges in applying MOT to small mobile robots. The first challenge is real-time processing. Real-time performance is a top priority for mobile robots that make control decisions based on object-tracking results. However, most existing techniques do not guarantee real-time performance. Looking at the results of MOT20 [4], the most recent benchmark of the MOT Challenge [3], the top 25 techniques in the accuracy performance metric [5,6] all have processing speeds below 20 Hz and only four of the top 32 techniques guarantee speeds above 20 Hz.

Considering these results, the SORT algorithm [7] is a suitable technique among existing multi-object tracking methods for mobile robots. SORT, denoted as Simple, Online, and Real-time Tracking, was first introduced in 2016 and achieved a top 32 ranking in terms of accuracy in the most recent MOT Challenge, the MOT20 benchmark, recording the fastest processing speed of 57.3 Hz among MOT benchmarks. SORT is widely used in embedded systems like mobile robots due to its fast processing speed and simplicity [8]. Nevertheless, when applying SORT to mobile robots, there is a degradation in object tracking accuracy performance, and the reasons for this are as follows, which leads to the second problem.

In particular, since SORT does not cope with temporary object detection failures, it is greatly affected by temporary changes in input frames because it does not cope with temporary object detection failures, and the performance of IOU tracker [9] based multi-object tracking techniques such as SORT is significantly degraded. Some methods consider object features to compensate for these problems, but the resulting increase in computation time is unavoidable [10].

Due to these problems, the existing multi-object tracking techniques are not suitable for person-tracking mobile robots that need to maintain real-time and stable tracking. In this paper, we propose a methodology that overcomes the limitations of these existing techniques and makes them applicable to small mobile robots with limited computational capabilities. We introduce a method that estimates the rotation degree of the robot using wheel odometry and utilizes it to calibrate the human tracking algorithm in real time. Furthermore, we present a dynamic setting method for the object tracker's survival period to maintain stable tracking.

## 2. Related research

### 2.1. Pedestrian tracking for mobile robot

Classic methods for applying person-tracking techniques to mobile robots have predominantly relied on distance measurement sensors. Research based on laser sensors has explored methodologies such as recognizing a person's legs to infer their movement trajectories [11] and employing multiple laser sensors in multiple hypothesis tracking (MHT) [12] to perform human tracking [13].

Methodologies using camera sensors include stereo camera tracking [14] and RGB-D (color and depth) camera tracking [15,16], which have shown high performance in terms of processing speed. However, these distance measurement techniques have limitations in that they cannot track objects beyond the perception limit of the distance measurement sensor, and their dependence on the sensor makes them difficult to apply to small mobile robots.

Currently, with the development of deep learning, especially convolutional neural network-based vision recognition technology, visual object tracking techniques are showing higher accuracy performance than classical methodologies [17]. Since these methods use only camera images as input, there is no dependence on specific ranging sensors, so they are generally applicable to small robots and a lot of research has been done on them [18-20].

## 2.2. Visual object tracking

Visual object tracking is the task of detecting objects within a given frame and maintaining an identifiable state for each object. For tracking, it is necessary to first receive information about the object to be tracked, and based on whether a detection model is used or not, it is categorized into detection-based trackers, which use a detection mode, and detection-free trackers.

Detection-free trackers have the advantage of being able to track untrained objects since they determine the target to track based on an initial target at the beginning of the tracking sequence. Moreover, since they do not typically apply computationally demanding detection models, they offer computational advantages. However, detection-free trackers are only suitable for short sequences and are not well-suited for single-object tracking (SOT), which tracks only one object.

With the advancement of convolutional neural network (CNN) techniques and the development of object detection models based on them, many object detection models have achieved a high level of accuracy. Based on this, detection-based trackers have achieved high performance in the MOT domain. Compared to detection-free trackers, detection-based trackers that perform MOT show high performance even in long sequences, and since they track multiple objects, the algorithm is highly scalable, making it suitable for human tracking. On the other hand, in the field of MOT, methodologies based on the Siam tracker [21] or CFtracker [22] perform the best in the SOT field, while there is no obvious algorithm that performs the best in the MOT field. In particular, the complexity of the algorithm does not seem to lead to better performance, with a simple IoU tracker [9] using a good object detector often performing better than more complex algorithms.

## 2.3. SORT (Simple, Online, and Real-time Tracking)

SORT is an online real-time tracking technique based on the Kalman Filter and Hungarian algorithm, which approximates the motion of each tracked object with a linear constant velocity model. The object tracking process of SORT is as follows.

(1) Detect the positions of all objects in the frame using an object detection model.
(2) Predict the next location of tracked objects using the Kalman Filter.
(3) Match the detection values and prediction values based on IoU scores using the Hungarian algorithm.
(4) Assign unique IDs to each object based on the matching results and update the Kalman Filter.

SORT uses only object detection results for matching between the current frame and the immediately preceding frame, ignoring other ancillary information such as bounding box locations. Because it does not use object features for tracking, temporary detection failures due to occlusion or overlapping targets will result in tracking failures.

Additionally, the camera viewpoint changes due to robot movement, making it difficult to match object detections with tracking predictions. **Figure 1** is an example of object tracking failure due to robot movement. This can be a critical problem for small mobile robots, where the time interval between consecutive frames is relatively long due to low computational processing speed.

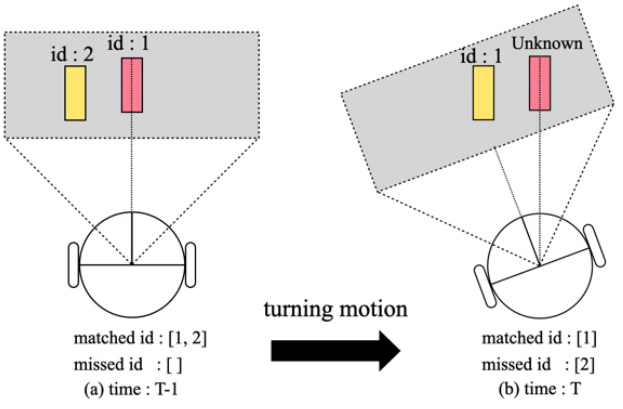


**Figure 1.** Example of object tracking failure due to rotation of mobile robot. When two objects remain stationary, the objects tracked in T-1 **(a)** are assigned IDs 1 and 2. The robot rotates between T-1 and T **(b)**, and the object assigned 1 fails to track in T, and object 2 is incorrectly tracked as object 1 in T.

A methodology that compensates for the above problems caused by not including object feature information is DeepSORT [23], which reflects the feature points of the target recorded in the feature map of the object detection model in the matching, but DeepSORT increases the processing time by about 3 times compared to SORT but does not achieve a noticeable performance improvement in terms of accuracy, from 59.8 to 61.4 of SORT [23].

### 2.4. Research on camera motion measurement

To address the problem of degraded object estimation performance due to moving cameras, we need to estimate the degree of camera motion. Visual odometry (VO) [24], a method for estimating camera motion from vision information, can be divided into stereo visual odometry (SVO) and monocular visual odometry (MVO) depending on whether a stereo or monocular camera is used.

VO methods may suffer from performance degradation due to factors such as dim lighting conditions or environmental interference, and they require significant computational resources as they operate by comparing multiple feature points across consecutive frames. Even the ARM-VO [25] technique proposed for small systems can only achieve a processing speed of 8 Hz when applied to Raspberry Pi 3.

Rather than this computationally intensive VO method, an odometry calculation method based on wheel encoders is more suitable for small mobile robots. The method of calculating the wheel encoder and using it to estimate the camera movement cannot be universally applied to all mobile camera cases like VO, and measurement errors may occur due to the slip of the mobile robot's wheels, but it can obtain camera movement estimation results with less computation than VO.

## 3. Methodology

### 3.1. Hardware configuration and wheel odometry measurement of the mobile robot

The hardware of the mobile robot EDIE used in this study is shown in **Figure 2**.

EDIE is a differential drive robot based on two wheels, with a wheel radius of 0.035 m, a distance between wheels of 0.108 m, and an encoder resolution of 374 pulses per rotation. The MCU used is a Raspberry Pi 4, and the VPU for parallel computation processing is a NeuralCompute Stick 2 from Intel.

The odometry of a robot is usually represented by $x$ , $y$, and $\theta$ information to describe the relative position between the starting point and the robot, but

**Figure 2.** EDIE hardware CAD model and physical hardware

the amount of change in $x$ and $y$ between 1 Hz relative to the camera frame is small compared to the amount of change in $\theta$, so we did not consider correction for x and y in this study. In the case of the robot used in the actual study, if the camera frame is updated at a frequency of 20 Hz when the speed of each motor is 1 m/s, $x$ can have a maximum change of 0.05 m, but in the case of rotation, if the left and right wheels are rotated in reverse, it can have a maximum change of 53.05°.

When the camera is fixed to the robot, the camera movement can be obtained by applying the transformation matrix to the robot's odometry, and the rotation angle of the robot's odometry was calculated using the wheel encoder values as shown in Equation (1).

$$\Delta\theta = \frac{\Delta Distance_{Right} - \Delta Distance_{Left}}{wheelTrack} \quad (1)$$

To correct the camera frame using the odometry calculated through the wheel encoder, synchronization between the odometry and the camera is required. For this purpose, the camera buffer size was changed to 1 to get only real-time images from the camera, and it was operated asynchronously with the main program using multithreads. In addition, the odometry information and the time stamp of the measurement were stored in the buffer to synchronize with the camera frame.

## 3.2. Real-time correction of object tracking considering camera rotation

The main goal of this calibration is to ensure that the Kalman Filter's linear prediction model for predicting the next position of an object only recognizes the motion of the object, not the motion of the camera. This provides the basis for maintaining tracking in the face of camera movements and being resistant to temporary object detection failures. The flowchart of the human tracking technique, including the calibration process, is shown in **Figure 3**.

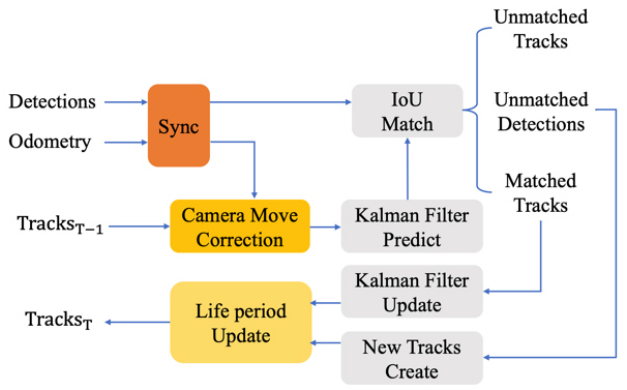


**Figure 3.** Flowchart of person tracking algorithm including real-time rotation correction

As shown in **Figure 3**, compensation for camera movement is performed just before the Predict stage of the Kalman Filter based on synchronized odometry. In the Predict stage, the Kalman Filter predicts the next state of the object according to a linear constant velocity system, and in the Update stage, the prediction error covariance and state variables are corrected based on the error between the matched actual object detection and prediction values. Since the Kalman Filter is updated after compensating for camera motion, the prediction error covariance and state variables can be updated with only the intact motion of the object. The Kalman Filter state variables used in this process are shown in Equation (2) [7].

$$State = [x, y, s, r, \dot{x}, \dot{y}, \dot{s}]^T \quad (2)$$

The values of interest are $x$ and $\dot{x}$, where represents the horizontal pixel value of the object center coordinate and $\dot{x}$ represents the velocity value of the object. The rotational motion correction corresponds to the correction of the horizontal axis on the frame and corrects $x$ by the difference in camera rotation angle between the current and previous viewpoints using Equation (3).

$$x' = x + (\theta_T - \theta_{T-1}) \times \frac{frame\_width}{fov} \quad (3)$$

Here, frame_width represents the horizontal pixel

width of the frame, and fov represents the camera's horizontal field of view. Since $\dot{x}$ is calculated based on the corrected $x$ value, only the intact motion of the object is included in the velocity calculation. The effect of applying these real-time corrections can be seen in **Figure 4**.

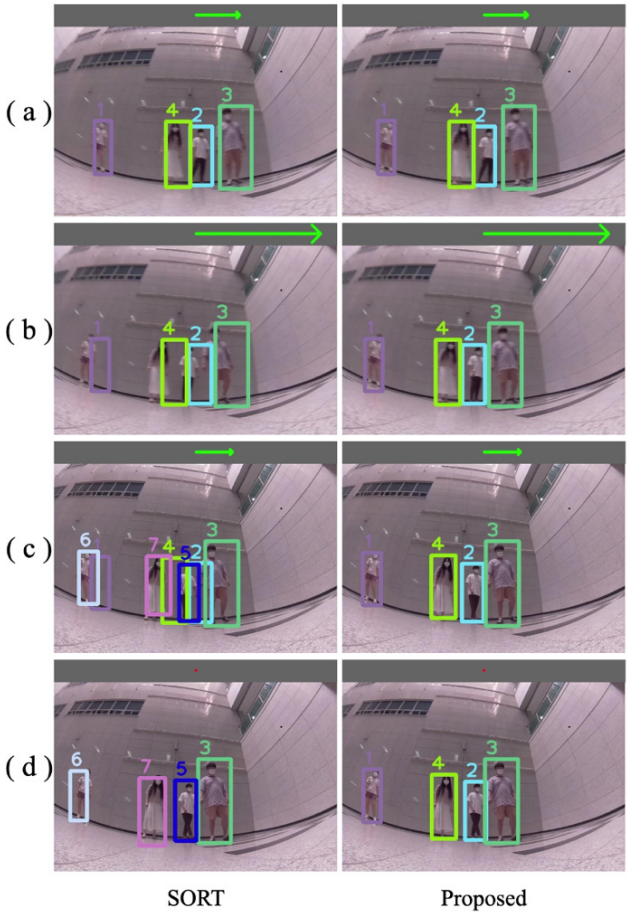


**Figure 4.** Object tracking prediction results during clockwise rotation

**Figure 4** depicts a scenario in which the mobile robot is rotating clockwise, and in four consecutive frames, four people are continuously detected. In the case of SORT, despite successful object detection in all frames, matching fails due to errors caused by rotation, leading to a change in object IDs from **(b)** onwards, ultimately resulting in three out of four object IDs changing in **(d)**. Nonetheless, in the proposed technique, object tracking is successfully maintained, even in the presence of robot motion, by predicting the next position stably.

## 3.3. Dynamical setting of the survival period of object trackers

In multi-object tracking, when the matching between object detection results and predictions fails, it is necessary to determine how long to maintain tracking of objects. The setting of the survival period of these object trackers should consider computational efficiency and tracking stability.

SORT determines that tracking does not need to be maintained if an object has not been detected for $T_{lost}$ frames, and in all experiments, $T_{lost}$ was set to 1. The reason for this setting is that SORT does not trust the predictions of the linear model and does not address the re-identification problem. Therefore, deleting object trackers that have failed to match quickly can increase computational efficiency, but it can lead to frequent tracking failures in the case of a detection model with relatively low detection performance.

However, in the proposed technique, camera motion correction improves the reliability of predictions, allowing for an extension of the object tracker's survival period. Furthermore, to reflect the improved prediction performance of the linear model when tracking is maintained for a longer period, the survival period of the existing static object tracker was set to increase proportionally to the tracking maintenance period. If tracking is maintained for $T_{Hits}$ frames, the survival period $T_{Age}$ of the object tracker is calculated as shown in Equation (4).

$$T_{Age} = min\left(T_{Lost} + \frac{T_{Hits}}{R}, \max\_age\right) \quad (4)$$

Here, max_age is the maximum for limiting the age of survival, and R is a variable that determines how much of $T_{Hits}$ should be taken into account.

The performance gains from dynamically setting the object tracker's survival time based on its prediction confidence are more pronounced in the presence of motion blur. Motion blur is a common occurrence in mobile robots, and the degradation of object tracking performance due to motion blur can be seen in **Figure 5**.

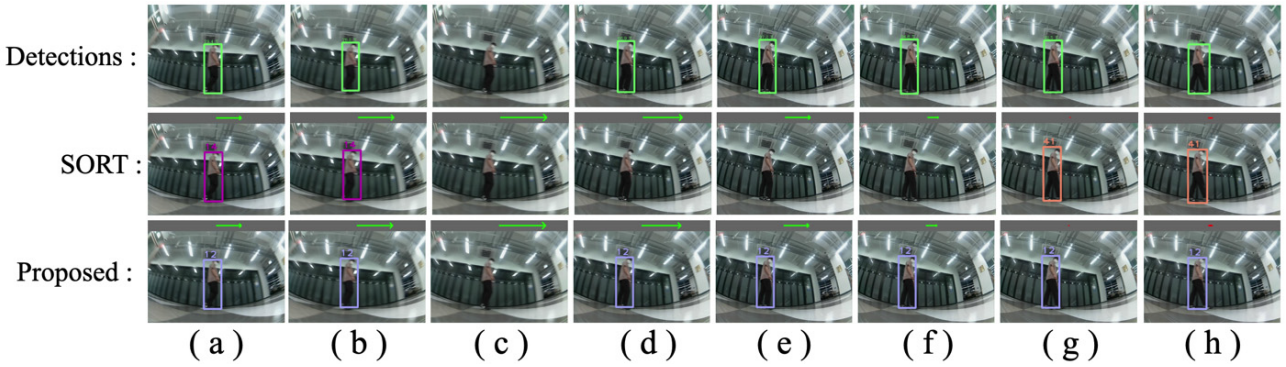


**Figure 5.** Object tracking results when object detection fails due to motion blur

**Table 1.** Overview of the PxRx sequences

| Sequence | FPS | Resolution | Length | Boxes | |
|---|---|---|---|---|---|
| PmRr | 15 | 640 × 480 | 266 (00:18) | 899 | 3.076° |
| PsRm | 15 | 640 × 480 | 366 (00:24) | 1,169 | 2.420° |
| PmRm1 | 15 | 640 × 480 | 460 (00:30) | 1,442 | 2.369° |
| PmRm2 | 20 | 640 × 480 | 449 (00:22) | 1,461 | 2.091° |
| PoRt1 | 20 | 640 × 480 | 739 (00:37) | 2,698 | 0.858° |
| PoRt2 | 20 | 640 × 480 | 564 (00:28) | 1,712 | 1.034 |

At the **(c)** time point in **Figure 5**, object detection fails, and the object tracker cannot update through matching. Instead, it only predicts the next location of the object using the Kalman Filter. In this situation, with only prediction in progress, the camera's viewpoint changes due to motion again, leading to tracking failure in SORT at **(g)** and a change in object IDs. In contrast, the proposed technique's object tracker, through stable prediction of the next position, successfully maintains tracking when objects are detected again at **(d)**.

# 4. Experiments

## 4.1. Test data

MOT Challenge data is commonly used as the performance evaluation criterion for multi-object tracking techniques. However, this data does not contain information about camera movement. Therefore, to evaluate the performance of the proposed methodology, test data that includes odometry information had to be constructed.

The data was collected from a camera attached to a mobile robot and contains camera odometry information and object position information for each frame. The ground truth position data of the objects was collected in the MOT20 Challenge format, where the odometry data contains $x$, $y$, and $\theta$ and the corresponding frame ID and is synchronized with the ground truth position data based on the frame ID.

The test data was collected in a total of 6 sequences and information about each sequence can be found in **Table 1**. $th_{avg}$ represents the average rotation angle of the mobile robot per frame, which can be used to observe how much the robot rotated at a time during the sequence.

## 4.2. Evaluation metrics

The following metrics were used for evaluating the performance of multi-object tracking techniques.

- *MOTA*(↑): Multi-object tracking accuracy [3]
- *MOTP*(↑): Summary of overall tracking

precision [3]

- $IDF_1$(↑): The ratio of correctly identified detections over the average number of ground-truth and computed detections [26]
- *IDP*(↑): Identification precision [26]
- *IDR*(↑): Identification recall [26]
- *FP*(↓): number of false detections [3]
- *FN*(↓): number of missed detections [3]
- *IDsw*(↓): Number of times the reported identity of ground-truth track changes [3]
- *Hz*(↑): Processing speed (in frames per second excluding the detector)

Metrics marked with (↑) indicate higher performance, whereas metrics marked with (↓) indicate lower performance.

## 4.3. Object detection model

Since object tracking is based on object detection results, it is highly dependent on object detection performance, hence, selecting an appropriate object detection model is a very important issue for object tracking. There are various CNN-based object detection models, but YOLO [27] and SSD [28], which are one-stage object detection models that can perform real-time inference even with limited computing power, are suitable for mobile robots.

The object detection accuracy of YOLOv4 [29], YOLOv4-tiny [30], and SSDlite [31], which are object detection models in the YOLO and SSD series, was measured using PxRx data. Among these models, YOLOv4 showed the highest accuracy, followed by YOLOv4-tiny and SSDlite. For this experiment, the YOLOv4-tiny model was selected and used as the object detection model considering both accuracy and processing speed.

## 4.4. Dynamic setting of object tracker's survival period experiment

The survival period of an object tracker is determined by how much the tracker's predictions can be trusted. In the case of SORT, the highest performance was achieved when $T_{lost}$ was set to 1, since the design only considers frame-to-frame, and the prediction error is bound to be larger in the presence of camera movement.

**Table 2** shows the experimental results using test data collected from a mobile robot, which shows a MOTA of 50.77 when $T_{lost}$ is set to 1, followed by 49.63, 49.63, and 49.83 when $T_{lost}$ is set to 2, 3, and 4, respectively.

In contrast, when the survival period of the object tracker was dynamically set through camera motion correction to increase prediction reliability, the tracking accuracy improved. Experiment results using MOT17-

**Table 2.** Performance evaluation table according to the survival period setting of the object tracker

| Method | | *MOTA*(↑) | *FN*(↓) | *IDsw*(↓) |
|---|---|---|---|---|
| SORT | 1 (Default) | 50.77 | 4,332 | 274 |
| SORT | 2 | 49.63 | 4,449 | 266 |
| SORT | 3 | 49.63 | 4,447 | 265 |
| SORT | 4 | 49.83 | 4,437 | 258 |
| SORT | Dynamic | 51.93 | 4,226 | 268 |

**Table 3.** Tracking performance test results on MOT17

| Method | | *MOTA*(↑) | *MOTP*(↑) | *FN*(↓) | *IDsw*(↓) |
|---|---|---|---|---|---|
| SORT | 1 | 49.01 | 85.83 | 51,343 | 1,043 |
| SORT | Dynamic | 49.97 | 85.65 | 49,997 | 1,051 |

**Table 4.** Tracking performance test results on PxRx sequence

| Method | *MOTA*(↑) | *MOTP*(↑) | *FP*(↓) | *FN*(↓) | *Hz*(↑) | $IDF_1$(↑) | *IDP*(↑) | *IDR*(↑) | *IDsw*(↓) |
|---|---|---|---|---|---|---|---|---|---|
| SORT | 49.44 | 81.50 | 13 | 4,453 | 466.3 | 29.86 | 43.27 | 22.79 | 277 |
| SORT + Dynamic | 51.93 | 81.34 | 15 | 4,226 | 429.6 | 37.62 | 52.94 | 29.18 | 268 |
| SORT + Odom | 53.42 | 82.23 | 11 | 4,179 | 440.6 | 39.19 | 54.86 | 30.49 | 180 |
| Proposed | 57.58 | 81.90 | 14 | 3,813 | 420.7 | 46.02 | 61.68 | 36.70 | 152 |

02, 05, 09, 10, and 11 data collected with the camera facing forward [32] in **Table 3** demonstrated that a simple adjustment of the object tracker's survival period led to a 0.96% increase in tracking accuracy.

## 4.5. Experimental results

Object tracking performance measurements were conducted through sequences that included ground truth information for multiple objects and odometry information. In **Table 4**, SORT, SORT+Dynamic (applying dynamic survival period setting only), SORT+Odom (applying motion correction only), and Proposed (applying both motion correction and dynamic survival period setting) were evaluated using collected test data.

In the case of SORT+Odom, the performance exceeded SORT's all metrics, with a significant improvement in the *IDsw* metric, decreasing from 277 to 180. This signifies enhanced tracking stability in dynamic situations due to the proposed correction technique.

SORT+Dynamic showed only a slight improvement of 2.49 in terms of MOTA compared to SORT. However, when applied together with camera motion correction, Proposed exhibited a significant 4.16 improvement in performance compared to SORT+Odom. This indicates that the performance improvement is higher when the survival period is dynamically set, reflecting the increased prediction performance of the tracker.

Most metrics, apart from MOTA, which represents overall tracking accuracy, showed the highest performance when both techniques were used together, demonstrating a meaningful improvement in both accuracy and stability. Moreover, the processing speed was only slightly reduced, from SORT's 2.24 ms to 2.38 ms, showing that the tested multi-object tracking techniques ran on a system consisting of a 1.5 GHz Cortex-A72 single-core processor and 4 GB memory.

# 5. Conclusion

In this paper, we have proposed a multi-object tracking method that enhances the tracking performance of small mobile robots in human tracking scenarios by real-time correction of tracking errors caused by the robot's motion using wheeler encoder odometry information and dynamically managing the survival period of the object tracker. To evaluate the performance of the methodology, we collected six sequences of data, including odometry information, from the mobile robot. When using the YOLOv4-tiny object detector, the proposed method maintained real-time processing at a speed of 2.38 ms, demonstrating a 16.46% improvement in MOTA compared to SORT experimentally. In addition, by dynamically managing the survival period based on the confidence of the prediction model for the tracked objects, we confirmed that object identities are maintained without switching, leading to enhanced tracking stability over extended periods. At a time when interactions between mobile robots and humans are becoming increasingly active, we anticipate that the technique proposed in this paper will enable more precise and stable tracking of humans. This, in turn, will facilitate high-confidence interactions between robots and humans through human tracking.

## Disclosure statement

The authors declare no conflict of interest.

## Acknowledgment

This research was supported by the Ministry of Trade, Industry, and Energy in Korea, under the Global Talent Growth Support Project for Robot Industry Innovation (P0017311) supervised by the Korea Institute for Advancement of Technology (KIAT).

# References

[1] Bogue R, 2016, Growth in E-Commerce Boosts Innovation in the Warehouse Robot Market. Industrial Robot, 43(6): 583–587. https://doi.org/10.1108/IR-07-2016-0194

[2] Hitti N, 2020. Ballie the Rolling Robot is Samsung's Near-Future Vision of Personal Care. Dezeen. https://www.dezeen.com/2020/01/08/samsung-ballie-robot-ces-2020/

[3] Leal-Taixé L, Milan A, Reid I, et al., 2015, MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking. arXiv preprint, arXiv: 1504.01942. https://doi.org/10.48550/ arXiv.1504.01942

[4] Dendorfer P, Rezatofighi H, Milan A, et al., 2020, MOTO20: A Benchmark for Multi Object Tracking in Crowded Scenes. arXiv preprint, arXiv: 2003.09003. https://doi.org/10.48550/ arXiv.2003.09003

[5] Xu Y, Ban Y, Delorme G, et al., 2021, TransCenter: Transformers with Dense Representations for Multiple-Object Tracking. arXiv preprint, arXiv: 2103.15145. https://doi.org/10.48550/arXiv.2103.15145

[6] Stadler D, Beyerer J. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 20–25, 2021: Improving Multiple Pedestrian Tracking by Track Management and Occlusion Handling. 2021, Nashville, 10953–10962. https://doi.org/10.1109/ cvpr46437.2021.01081

[7] Bewley A, Ge Z, Ott L, et al. 2016 IEEE International Conference on Image Processing (ICIP), September 25–28, 2016: Simple Online and Realtime Tracking. 2016, Phoenix, 3464–3468. https://doi.org/10.1109/icip.2016.7533003

[8] Kapania S, Saini D, Goyal S, et al., 2020, Multi Object Tracking with UAVs using Deep SORT and YOLOv3 RetinaNet Detection Framework. Proceedings of the 1st ACM Workshop on Autonomous and Intelligent Mobile Systems, 2020: 1–6. https://doi.org/10.1145/ 3377283.3377284

[9] Bochinski E, Eiselein V, Sikora T. 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), August 29–September 1, 2017: High-Speed Tracking-by-Detection Without Using Image Information. 2017, Lecce, 1–6. https://doi.org/10.1109/avss.2017.8078516

[10] Pereira R, Carvalho G, Garrote L, et al., 2022, Sort and Deep-SORT Based Multi-Object Tracking for Mobile Robotics: Evaluation with New Data Association Metrics. Applied Science, 12(3): 1319. https://doi.org/10.3390/app12031319

[11] Horiuchi T, Thompson S, Kagami S, et al. 2007 IEEE International Conference on Systems, Man, and Cybernetics, October 7–10, 2007: Pedestrian Tracking from a Mobile Robot Using a Laser Range Finder. 2007, Montreal, 931–936. https://doi.org/10.1109/icsmc.2007.4413964

[12] Reid D, 1979, An Algorithm for Tracking Multiple Targets. IEEE Transactions on Automatic Control, 24(6): 843–

854. https://doi.org/10.1109/tac.1979.1102177

[13] Tsokas NA, Kyriakopoulos KJ, 2011, Multi-Robot Multiple Hypothesis Tracking for Pedestrian Tracking. Autonomous Robots, 32: 63–79. https://doi.org/10.1007/s10514-011-9259-7

[14] Nam B, Kang S-I, Hong H. 2011 17th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV), February 9–11, 2011: Pedestrian Detection System Based on Stereo Vision for Mobile Robot. 2011, Ulsan, 1–7. https://doi.org/10.1109/fcv.2011.5739758

[15] Basso F, Munaro M, Michieletto S, et al. Fast and Robust Multi-People Tracking from RGB-D Data for a Mobile Robot. In Intelligent Autonomous System 12. 2013, Springer, Berlin, Heidelberg, 265–276. https://doi.org/10.1007/978-3-642-33926-4_25

[16] Zhang H, Reardon C, Parker LE, 2013, Real-Time Multiple Human Perception with Color-Depth Cameras on a Mobile Robot. IEEE Transactions on Cybernetics, 43(5): 1429–1441. https://doi.org/10.1109/tcyb.2013.2275291

[17] Sun Z, Chen J, Chao L, et al., 2021, A Survey of Multiple Pedestrian Tracking Based on Tracking-by Detection Framework. IEEE Transactions on Circuits and Systems for Video Technology, 31(5): 1819–1833. https://doi.org/10.1109/tcsvt.2020.3009717

[18] Hossain S, Lee D-J, 2019, Deep Learning-Based Real-Time Multiple-Object Detection and Tracking from Aerial Imagery via a Flying Robot with GPU-Based Embedded Devices. Sensors, 19(15): 3371. https://doi.org/10.3390/s19153371

[19] Thang DN, Nguyen LA, Dung PT, et al. 2018 5th NAFOSTED Conference on Information and Computer Science (NICS), November 23–24, 2018: Deep Learning-Based Multiple Objects Detection and Tracking System for Socially Aware Mobile Robot Navigation Framework. 2018, Ho Chi Minh City, 436–441. https://doi.org/10.1109/nics.2018.8606878

[20] Carvalho G de S. Kalman Filter-Based Object Tracking Techniques for Indoor Robotic Applications. 2021, Universidade de Coimbra. https://estudogeral.sib.uc.pt/handle/10316/98163

[21] Koch G. Siamese Neural Networks for One-Shot Image Recognition. 2015, University of Toronto. http://www.cs.toronto.edu/~gkoch/files/msc-thesis.pdf

[22] Bolme DS, Beveridge JR, Draper BA, et al. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June 13–18, 2010: Visual Object Tracking Using Adaptive Correlation Filters. 2010, San Francisco, 2544–2550. https://doi.org/10.1109/ cvpr.2010.5539960

[23] Wojke N, Bewley A, Paulus D. 2017 IEEE International Conference on Image Processing (ICIP), September 17–20, 2017: Simple Online and Realtime Tracking with a Deep Association Metric. 2017, Beijing, 3645–3649. https://doi.org/10.1109/icip.2017.8296962

[24] Scaramuzza D, Fraundorfer F, 2011, Visual Odometry [Tutorial]. IEEE Robotics & Automation Magazine, 18(4): 80–92. https://doi.org/10.1109/mra.2011.943233

[25] Nejad ZZ, Ahmadabadian AH, 2019, ARM-VO: An Efficient Monocular Visual Odometry from Ground Vehicles on ARM CPUs. Machine Vision and Applications, 30: 1061–1070. https://doi.org/10.1007/s00138-019-01037-5

[26] Ristani E, Solera F, Zou R, et al. Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking. In Computer Vision – ECCV 2016. 2016, Springer, Cham. https://doi.org/10.1007/978-3-319-48881-3_2

[27] Redmon J, Divvala S, Girshick R, et al. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 27–30, 2016: You Only Look Once: Unified, Real-Time Object Detection. 2016, Las Vegas, 779–788. https://doi.org/10.1109/cvpr.2016.91

[28] Liu W, Anguelov D, Erhan D, et al. SSD: Single Shot MultiBox Detector. In Computer Vision – ECCV 2016. 2016, Springer, Cham. https://doi.org/10.1007/978-3-319-46448-0_2

[29] Bochkovskiy A, Wang C-Y, Liao H-YM, 2020, YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprints, arXiv: 2004.10934. https://doi.org/10.48550/arXiv.2004.10934

[30] Wang C-Y, Bochkovskiy A, Liao H-YM. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 20–25, 2021: Scaled-YOLOv4: Scaling Cross Stage Partial Network. 2021, Nashville, 13024–13033. https://doi.org/10.1109/cvpr46437.2021.01283

[31] Sandler M, Howard A, Zhu M, et al. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, June 18–23, 2018: MobileNetV2: Inverted Residuals and Linear Bottlenecks. 2018, Salt Lake City, 4510–4520. https://doi.org/10.1109/cvpr.2018.00474

[32] Milan A, Leal-Taixé L, Reid I, et al., 2016, MOT16: A Benchmark for Multi-Object Tracking. arXiv preprint, arXiv: 1603.00831. https://doi.org/10.48550/arXiv.1603.00831