A Vision-based Target Tracking Robot System for Embedded Platforms

Yi Liu
Department of Engineering and Design
University of Sussex
Brighton, UK
yl701@sussex.ac.uk

2021 10th Mediterranean Conference on Embedded Computing (MECO) | 978-1-6654-3912-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/MECO52532.2021.9460198

Abstract— **Identification and tracking of dynamic objects are essential concepts in the field of robotic vision. This paper presents the design of a four-wheeled robot that can detect a target object, track the object, and maintain a constant distance from it. In this design, S5P6818 system-on-a-chip (SoC) is used as the core processing unit with embedded Linux OS, and a USB video class (UVC) camera is used for image acquisition. The robot captures real-time images and tracks the object based on a joint tracking algorithm of enhanced Continuously Adaptive Mean Shift (CAMShift) and Kalman filter (KF). A combination of Bang-bang controller and proportional–integral–derivative (PID) controller is used to control the robot's real-time motion. The experimental results show that the robot can track a target object and keep a certain distance in real-time, proving its effectiveness and robustness.**

Keywords- Embedded robotics; robotic vision; embedded system; mobile robot; object tracking; CAMShift

I. INTRODUCTION

At present, autonomous robots are widely used in various fields. An autonomous robot can determine the actions to perform a task, with the assist of a perception system [1]. The development tendency of the future robots is further intelligent.

The basics of autonomous mobile robots consist of the fields of locomotion, perception, cognition, and navigation [2]. In the field of cognition and navigation, artificial intelligence plays an important role in the process of information and perform tasks more efficiently. Among them, vision is one of the most important sensory systems. The vision-based approach not only provides multiple parameters such as color and shape, but also is the simplest and most effective way enabling artificial intelligence technology to reach the goal.

In recent years, with the expansion of the intelligence requirements in industry and research, machine vision has also been rapidly developed and gradually applied to various fields, such as image identification and pattern recognition. Due to the high computation overhead and power consumption, many of the machine vision designs are not suitable for resource-restrained embedded platforms [3]. Driven by the proposition of edge computing [4] and the progress of computing power on embedded processors, some designs are also implemented on embedded systems.

This paper aims at the target-tracking robot system for the Advanced RISC Machines (ARM) platform. Based on the real-time images, the robot is able to track an object and keep a certain distance using a joint tracking algorithm. Users can monitor and control the robot remotely through a desktop application on the PC. The experiments show that the robot has the advantage of low-cost and robustness.

II. SYSTEM STRUCTURE

The system's overall architecture is composed of a desktop application on the host PC and the robot. The desktop application is able to monitor and control the robot remotely. Based on the concept of edge computing, the majority of tasks are executed by the main processing unit on the robot, including image acquisition, object tracking algorithm, and data transmission.

A. Remote-control Application

Presented in Fig. 1, the desktop application provides a graphical user interface (GUI) for users to control and monitor the robot remotely. The wireless communication between the robot and host device opts for the User Datagram Protocol (UDP) in sockets programming to improve real-time performance and reduce latency [5]. Using click-and-drag on the real-time image, a screenshot of the target object can be captured and transmitted to the robot as a sample.

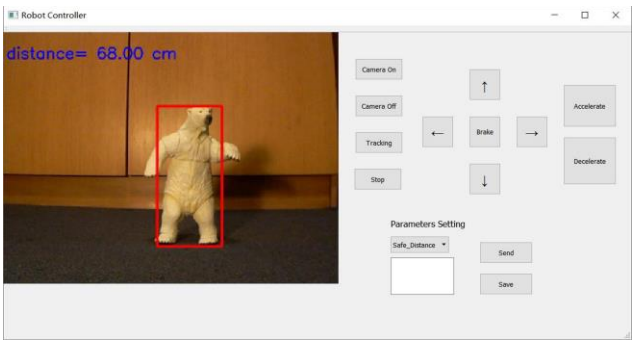

Figure 1. Desktop application GUI, showing that a polar bear model is being tracked

978-0-7381-3361-4/21/$31.00 ©2021 IEEE

B. Robot Hardware Structure

The robot is equipped with embedded control boards, Wi-Fi adapter, UVC camera, servo and DC motors. An image of the robot is displayed in Fig. 2.

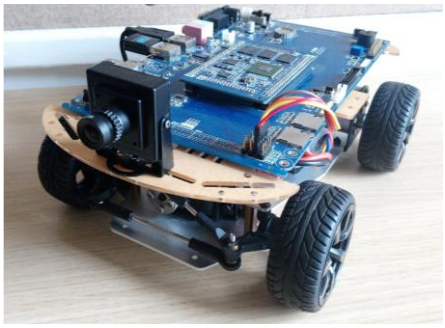

Figure 2. Image of the mobile robot

The robot adopts two controllers, including the S5P6818 SoC and STM32F103 Cortex-M3 microcontroller (MCU). The S5P6818 SoC is an octa-core Cortex-A53 CPU board, with dynamic frequency scaling up to 1.4GHz. Embedded Linux OS provides interaction with the low-level hardware elements. As previously discussed, the S5P6818 SoC performs the core tasks. For image process and wireless communication, the OpenCV library and device driver of the RT5572 USB Wi-Fi adapter are ported to the embedded platform. The forward-facing UVC camera is connected with the CPU board through USB for image acquisition.

The STM32F103 Cortex-M3 MCU is used as a lower controller, responsible for controlling the motion of the robot, connected with the CPU board via the UART. The Cortex-M3 MCU is connected with two DC motors for speed control and a servo for steering control. A closed-loop control algorithm is implemented to control DC motors' real-time speed and rotation angle of the servo.

C. Robot Software Structure

The robot's software framework can be divided into two subsystems: the vision subsystem and the robot control subsystem. The vision subsystem is implemented on the S5P6818 SoC. The subsystem includes four blocks: image acquisition, target tracking, distance estimation, and data transmission. The image acquisition block uses the UVC camera with a fixed view to capture real-time colored images, which have a resolution of 640 × 480 and a frame rate of 30 frames per second (FPS). In the target tracking block, the tracking algorithm is based on an enhanced joint tracking algorithm, which overcomes some disadvantages of traditional CAMShift. After that, the distance between the object and the robot will be estimated based on the object location on the image.

The robot control subsystem is implemented on the Cortex-M3 MCU, which controls the robot's real-time speed and steering. The base algorithm of this subsystem is a combination of PID and Bang-bang controllers. In this way, the robot can follow the target object's motion and keep a certain distance from the target object.

III. TARGET TRACKING ALGORITHM

A. CAMShift Algorithm

For object detection and track, the basic algorithm used in the system is the CAMShift algorithm. The CAMShift was first proposed by Bradski [6] and introduced the idea of adaptively adjusting the tracking window size and the probability distributions of targets. CAMShift is designed for dynamically changing distributions, which can be used to track dynamic objects in a lightweight and robust way. Comparing with other deep-learning-based methods such as Structure-Aware Network (SANet) and Multi-Domain Network (MDNet) trackers, CAMShift has lower complexity and consumption of computation resources while keeping a satisfying level of accuracy [7].

Illustrated in Fig. 3, the process of the traditional CAMShift algorithm can be summarized as follow: First, initialize the tracking window and convert the color model of real-time image from RGB to HSV. Then, extract the Hue (H) component and generate a color histogram within the tracking window. After that, the center of mass in the tracking window will be calculated based on the color probability distribution map derived from the histogram in the tracking window. Finally, move the center of the tracking window to the center of mass and upgrade the size of the tracking window. The iteration will repeat until the center of mass converges or the number of iterations reaches the maximum value. The updated size and position of tracking window will be used as the initial parameters in the next frame of image. The loop of iterations can fulfil the continuous track of an object.

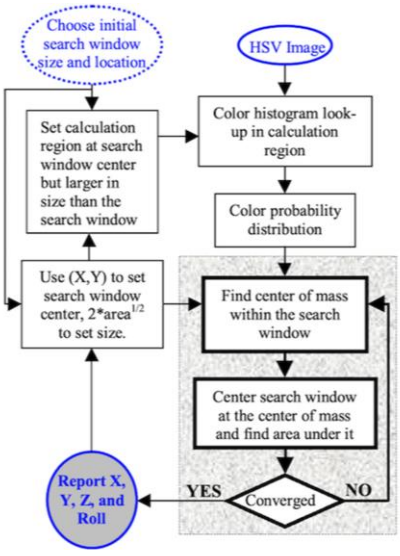

Figure 3. Block diagram of CAMShift algorithm [6]

Although CAMShift is widely used in object tracking because of low complexity and good real-time performance. However, there are still some drawbacks of CAMShift. First, if

other objects block the target, the tracking window is likely to diverge in irregular manners [8]. Similarly, if the target moves too fast, the tracking windows of the previous frame and current frame have no overlapped area. The track is also likely to fail. Additionally, if the track fails, it is unable to restore automatically even if the target returns to the original position.

Moreover, the traditional design requires manual initialization of the tracking window, providing the position of region around the target [9]. However, due to the inevitable latency in communication between the robot and the host device, the data transmission may cost more time than the time interval between two frames. The target's position might have left its initial region while receiving the initial parameters from the host device. It will reduce the quality of detection and causes drifts of the tracking window [10].

B. Enhancements on CAMShift

In order to solve the common issues of CAMShift and improve its stability in the actual environment, several methods are used to enhance the CAMShift algorithm. First, a 2D hue-saturation (HS) histogram replaces the traditional histogram solely based on hue component, improving the accuracy and robustness of CAMShift in complex environments. Second, the Kalman filter is employed to estimate the target's motion, which is used as a supportive tracking method when the target gets occluded. Additionally, the CAMShift algorithm is modified to automatically detect target in the entire frame based on the sample image, which also enables the re-detection when track fails. Combined with the methods above, the enhanced CAMShift algorithm is able to work appropriately on the embedded platform, showing satisfying real-time performance.

The enhanced algorithm combines CAMShift with the Kalman filter and automatic target detection. When the track fails, re-detection can be executed to detect target and restore tracking. Continuous and stable tracks can be performed in a complex environment or when obstacles block the target.

1) *Multi-dimensional Histogram:* The standard CAMShift algorithm only uses the hue component to generate a histogram. It is an efficient way to track the target with a simple appearance. However, the histogram's information is lopsided, and the track is likely to fail when the background is in a similar color [11]. In order to make better use of the color information, the saturation component is taken into consideration to implement the 2D HS histogram, replacing the traditional 1D histogram. The hue and saturation components are divided into 30 and 16 intervals, resulting in a total of 480 bins in the histogram. The number of pixels located in the certain hue and saturation interval will be assigned to the corresponding bin.

2) *Automatic detection:* The target detection is based on the sample image sent from the host PC. At the beginning of the detection process, the sample image will be loaded to generate a histogram, which will be used to detect the target on the entire image. The detection process usually iterates several times. When the tracking windows' size change in two consecutive frames is less than 5%, it is determined as a successful detection. The location, size and histogram of the current tracking window will be used as the initial state for the tracking process.

3) *Kalman filter:* The KF is a popular algorithm in real-time guidance, navigation, and control. The KF observes measurements over time and makes an estimation based on the observation. It only requires the states of a target in previous and current frames. Therefore, the KF is used to predict the center of the tracking window in this design.

KF contains two steps: prediction and update, which are implemented upon the linear system model below [12]:

$$x_{\mathrm{k}} = Fx_{\mathrm{k-1}} + w_{\mathrm{k}}, \tag{1}$$

$$z_{\mathrm{k}} = Hx_{\mathrm{k}} + v_{\mathrm{k}}, \tag{2}$$

where the x_k and x_{k-1} represent the state at k and k-1 respectively. In (1), the term F is the state-transition model, and w_k represents the Gaussian-distributed process noise. In (2), z_k represents the observation at time k, H is the observation model, and v_k is the Gaussian-distributed observation noise.

In the prediction step, the priori state estimate $x_{k|k-1}$ and priori estimate error covariance $P_{k|k-1}$ can be derived by:

$$x_{\mathrm{k|k-1}} = Fx_{\mathrm{k-1|k-1}} + w_{\mathrm{k}}, \tag{3}$$

$$P_{\mathrm{k|k-1}} = Fx_{\mathrm{k}}F^T + Q, \tag{4}$$

where Q is the process noise covariance matrix.

In the upgrade step, the Kalman gain matrix K_k can be calculated by:

$$K_{\mathrm{k}} = P_{\mathrm{k|k-1}}H^T(\mathrm{H}P_{\mathrm{k|k-1}}H^T + \mathrm{R})^{-1}, \tag{5}$$

where R is the covariance of the observation noise.

Then the prediction results can be adjusted by:

$$x_{\mathrm{k|k}} = x_{\mathrm{k|k-1}} + K_{\mathrm{k}}(z_{\mathrm{k}} - \mathrm{H}x_{\mathrm{k|k-1}}), \tag{6}$$

$$P_{\mathrm{k|k}} = (I - K_{\mathrm{k}}H)P_{\mathrm{k|k-1}}. \tag{7}$$

The target's movement on the image can be considered in a uniform speed within the time interval between two frames because the time interval is very short (i.e. around 33 ms). With the assist of the Kalman filter, the tracker is able to work properly when the target suffers occlusion.

4) *Target re-detection:* As mentioned previously, after the failure of tracks, it is unable to regain the tracks. In this paper, the Bhattacharyya distance between the histograms of the initial state and current frame is used to determine if the tracking is failed. The threshold value of the Bhattacharyya distance is set as 0.6 in this design. If the Bhattacharyya distance exceeds the threshold (i.e. less similarity), it will be defined that the CAMShift tracker is failed, and the KF prediction result will be used as the new location of the tracking window. The size of the new tracking window will be the same as the previous state. Otherwise, the CAMShift tracker works properly, and the CAMShift tracking result is used to upgrade the tracking window. If the CAMShift tracker fails consecutively over 60 times (i.e. unable to find the target

within 2 seconds), the target detection procedure will be called to re-detect the target.

C. The Joint Tracking Algorithm

In order to achieve accurate and robust tracking, the joint tracking algorithm is implemented, using CAMShift as the basic algorithm and combining with the enhancements introduced in the previous chapter. Fig. 4 shows the flow chart of the tracking process. The process is listed below:

1) Load a sample image and generate a sample 2D HS histogram.
2) Execute the automatic detection in the entire image. After successful detection, save the location, size and histogram of the tracking window as initial states. The central coordinates of the tracking window are also used as the initial states of the KF.
3) Make the KF prediction and compute CAMShift tracking process. Compute the Bhattacharyya distance between the histograms of the current tracking window and initial state. If the Bhattacharyya distance is over 0.6, the CAMShift tracker is failed, and execute step 4 and 5. Otherwise, execute step 6.
4) If the number of consecutive CAMShift failures is over 60, restart from step 2. Otherwise, use the KF prediction result as the tracking window's central coordinates and keep the window size unchanged. Update the tracking window based on the KF prediction.
5) The CAMShift tracking result is accurate. Update the tracking window based on CAMShift tracking result.
6) Update the KF using the current central coordinates of the tracking window as observation results. The current position and size of tracking window will be used on the next frame.

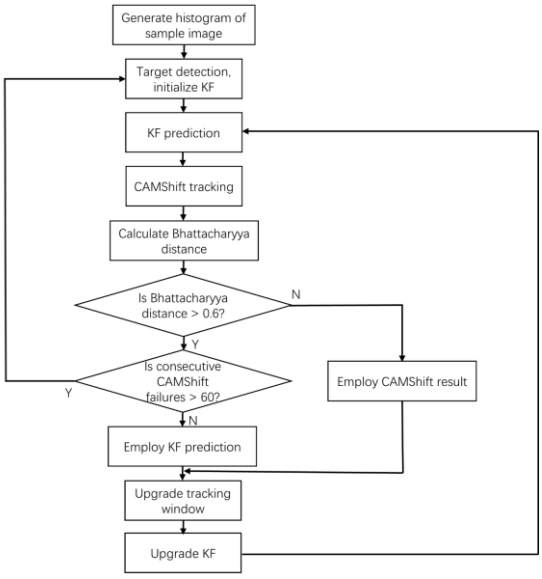

Figure 4. Flowchart of the joint tracking algorithm

IV. DISTANCE ESTIMATION ALGORITHM

When objects are being tracked, it is essential to determine their position and orientation with respect to the robot to navigate the object. In vision-based methods, a method uses the stereo camera systems that require at least two cameras or changing camera position [13]. Despite providing high-accuracy measurements, it requires higher hardware cost and system power consumption. In order to provide effective range estimation on a resource-strained embedded platform, monocular cameras are preferred for fulfilling the task.

Previous designs show that for monocular camera systems, it is currently unavailable to accurately estimate the distance that suits all cases without external sensors assist [14]. However, the accuracy of estimation can be improved to a satisfying level for specified environments within an effective range. In this paper, the distance estimation method is based on land objects located on the same ground surface as the robot. With zero roll and yaw angles of the camera, distance can be estimated through object position in the image based on pinhole camera geometry and similarity of triangles [15], shown in Fig. 5.

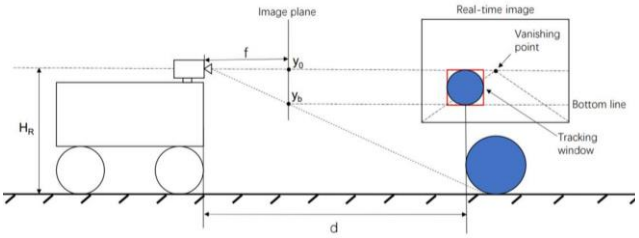

Figure 5. Diagram of imaging geometry

The center of the entire image is approximated as the vanishing point in the image, and the bottom edge of the tracking window is considered as the bottom line of the target object. If the object moves far away from the robot, the bottom line will approach the horizon, which passes through the vanishing point. The vertical distance in pixels between the bottom line of the tracking window and the horizon is inversely proportional to the real distance between the object and robot, with a factor between pixels and centimeters. The estimated distance d can be calculated as:

$$d = \frac{f\,H_R}{y_b - y_0}, \tag{8}$$

where f represents the camera focal length, H_R represents the height of the robot, and y_b and y_0 are the vertical coordinates of the tracking window bottom line and vanishing point in pixel.

V. ROBOT CONTROL ALGORITHM

The robot's speed and steering are controlled by DC motors and a DS3119 servo, respectively. In this paper, a PID controller is applied on the servo, and Bang-bang + PID two-

mode controller is used to control the real-time speed. PID controllers are widely used in industrial applications due to their robustness and functional simplicity [16]. In order to reduce the computing complexity, a simplified proportional-plus-derivative (PD) controller is used to control the steering. The purpose of applying the PD controller is to keep the target object within the central region in the field of view.

The horizontal central coordinate of the tracking window x is the current state, and the expected state x_0 is the horizontal coordinate of the center in the entire image. In the 640 × 480 images, the value of x_0 is 320. Hence the error of horizontal coordinate (e_x) can be obtained by:

$$e_x = x - x_0. \tag{9}$$

The output of the PD controller can be calculated by:

$$u_x = K_{Px} e_x + K_{Ix} e_x. \tag{10}$$

The DC motors are controlled by PID controllers. The purpose of using the PID controller is to keep a desired safe distance between the robot and the target object, by adjusting real time speed. The current distance d is estimated in cm through the method stated in the previous chapter. The expected state d_0 is the desired safe distance (100 cm). The error of distance (e_d) can be obtained by:

$$e_d = d - d_0. \tag{11}$$

The output the PID controller can be calculated by:

$$u_d = K_{Pd} e_d + K_{Id} \int e_d \, \mathrm{dt} + K_{Dd} \frac{de_d}{dt}. \tag{12}$$

Under extreme circumstances, the Bang-bang controller is used to control the motor. The Bang-bang controller has two thresholds. In this paper, the two thresholds are set at 300 cm and 48 cm, respectively. When the estimated distance exceeds 300 cm, the robot is too far from the target. The motor will apply maximum positive output. On the contrary, when the distance is less than 48 cm, the object is too close to the object. The motor will reverse with maximum output to avoid collisions. The Bang-bang controller can reduce the system response time under extreme circumstances, improving the efficiency and robustness of the system.

VI. EXPERIMENT RESULTS

A. Occlusion Experiment

In the occlusion experiment, the tracking performances of the joint tracking algorithm and standard CAMShift are tested, presented in Fig. 6, where the tracking windows of the joint tracking algorithm and standard CAMShift are illustrated in red and blue respectively. A blue notebook is the target object, moving from right to left and passing through an obstacle. When the target gets fully occluded, the standard CAMShift tracking window wrongly tracks the black area due to color interference. The tracking cannot be restored when the target re-appears in the field of view. However, the Kalman filter is able to predict an approximate position of the target for the joint tracking algorithm. When the target appears again, it can be tracked continuously and stably. The result proves that the joint tracking algorithm has better robustness to short-term occlusion and color interference, overperforming the standard CAMShift.

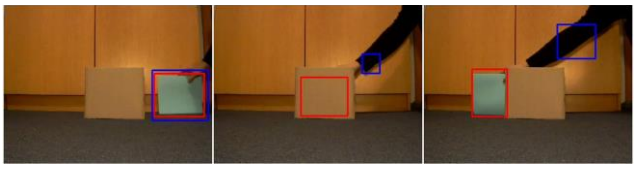

Figure 6. Occlusion experiment

B. Distance Measurement Experiment

In this experiment, the accuracy of the distance estimation algorithm is tested by measuring the same object placed at different distances, as presented in Fig.7. The real distances were pre-measured. The estimation was performed 20 times at each distance, and the average estimation value was calculated and recorded. The results are presented in Table I.

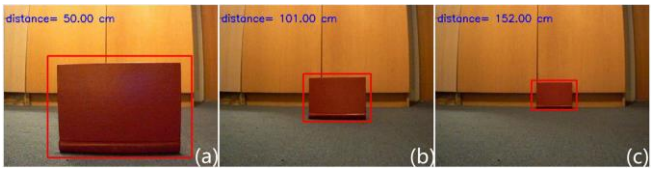

Figure 7. The same object placed at distances of 50 cm (a), 100 cm (b) and 150 cm (c) from the robot, with estimates displayed on the top-left

TABLE I. AVERAGE ESTIMATION RESULTS IN DIFFERENT DISTANCES

Real Distance (cm)	***50***	***100***	***150***	***200***	***250***	***300***	***350***	***400***
Average Estimation (cm)	51.1	100.3	147.8	196.1	249.2	293.9	316.6	357.4
Error rate (%)	-2.2	-0.3	1.9	1.9	0.3	2.0	9.5	10.7

It can be observed that the estimation error is able to maintain a relatively low level for measurements within 300 cm. At distances over 300 cm, the error rates go up to around 10%. Hence, the effective range of the distance estimation algorithm is set as 3 meters.

C. Real-time Tracking Experiment

In the real-time tracking experiment, the robot was tested in an indoor environment sized 3m × 3m. The route includes straight paths and turns. A red notebook carried by the operator was set as the target object. The safe distance was set as 1m. Fig. 8 presents the tracking path result, illustrating the paths of target and robot in red and blue respectively.

The result shows that, in linear tracking, the algorithm has satisfying tracking accuracy. During the experiment, the robot and the target keep a distance of 1m. During steering, the robot is able to select the optimal path to steer. In some cases, although the robot loses the target for a while, it can still re-locate and continue tracking after re-detection. Overall, the processing speed of the tracking system is fast enough for real-time applications, with the average video frame rate around 30

FPS. The movement of the robot is recorded by a camera, and some of the images are displayed in Fig. 9.

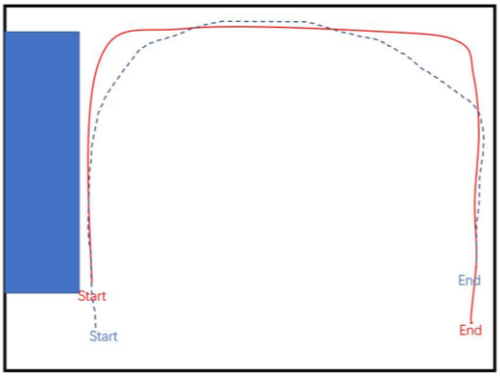

Figure 8. Plotted paths of target (red solid line) and robot (blue dotted line)

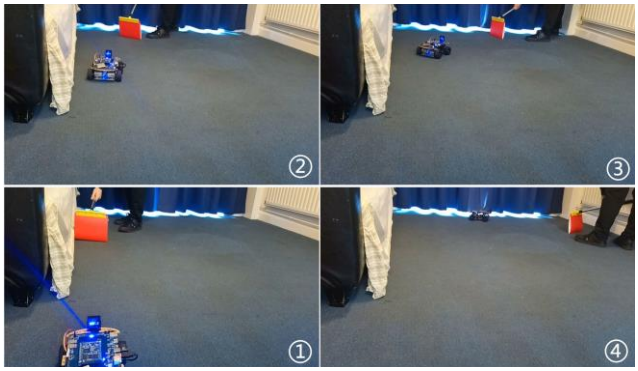

Figure 9. The robot in operation for target tracking

VII. CONCLUSION

In this paper, the target tracking robot system based on embedded platform is proposed and investigated. Integrated with CAMShift, Kalman filter and additional measures, the joint tracking algorithm is used for target tracking, increases the real-time performance and robustness of the tracking algorithm. Proved by the experimental results, the robot can estimate the real-time distance with a satisfying accuracy within the effective range, and the target is tracked properly. The robustness of the tracking algorithm is enhanced, preventing it from losing control due to occlusion and color interferences. The future work will consider combining other tracking algorithms for more complex environments.

REFERENCES

[1] F. Rubio, F. Valero, and C. Llopis-Albert, “A review of mobile robots: Concepts, methods, theoretical framework, and applications,” International Journal of Advanced Robotic Systems, vol. 16, no. 2, p. 172988141983959, 2019.

[2] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, Introduction to autonomous mobile robots. Cambridge, MA: MIT Press, 2011.

[3] S. Alyamkin et al., “Low-power computer vision: Status, challenges, and opportunities,” IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 9, no. 2, pp. 411–421, 2019.

[4] A. Davis, J. Parikh, and W. E. Weihl, “EdgeComputing: Extending enterprise applications to the edge of the internet,” in Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters - WWW Alt. ‘04, 2004.

[5] A. Milanovic, S. Srbljic, and V. Sruk, “Performance of UDP and TCP communication on personal computers,” in 2000 10th Mediterranean Electrotechnical Conference. Information Technology and Electrotechnology for the Mediterranean Countries. Proceedings. MeleCon 2000 (Cat. No.00CH37099), 2002.

[6] G. Bradski, “Real time face and object tracking as a component of a perceptual user interface,” Proceedings Fourth IEEE Workshop on Applications of Computer Vision. WACV98 (Cat. No.98EX201), 1998.

[7] P. Chen and Y. Zhou, “The review of target tracking for UAV,” in 2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2019.

[8] J. Cai, Y. Chen, J. Zhu, X. Zheng, and H. Wu, “CAMShift tracking algorithm for metro entrance and exit security,” 2019 Chinese Automation Congress (CAC), 2019

[9] S. Boubou, A. Kouno, and E. Suzuki, “Implementing camshift on a mobile robot for person tracking and pursuit,” in 2011 IEEE 11th International Conference on Data Mining Workshops, 2011.

[10] E. Emami and M. Fathy, “Object tracking using improved CAMShift algorithm combined with motion segmentation,” in 2011 7th Iranian Conference on Machine Vision and Image Processing, 2011.

[11] N. Papanikolopoulos and C. E. Smith, “Issues and experimental results in vision - guided robotic grasping of static or moving objects,” Industrial Robot, vol. 25, no. 2, pp. 134–140, 1998.

[12] M. Roth, G. Hendeby, C. Fritsche, and F. Gustafsson, “The Ensemble Kalman filter: a signal processing perspective,” EURASIP J. Adv. Signal Process., vol. 2017, no. 1, 2017.

[13] J. G. Allen, R. Y. D. Xu, and J. S. Jin. Object tracking using camshift algorithm and multiple quantized feature spaces. In VIP ’05: Proceedings of the Pan-Sydney area workshop on Visual information processing, pages 3–7, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[14] K.-S. Hsu, K.-C. Chen, T.-H. Li, and M.-C. Chiu, “Development and Application of the Single-Camera Vision Measuring System,” Journal of Applied Sciences, vol. 8, no. 13, pp. 2357–2368, 2008.

[15] K. Park and S. Hwang, "Robust Range Estimation with a Monocular Camera for Vision-Based Forward Collision Warning System", The Scientific World Journal, vol. 2014, pp. 1-9, 2014.

[16] R. H. Bishop and R. C. Dorf, Modern Control Systems: International Edition, 10th ed. Upper Saddle River, NJ: Pearson, 2005.