Automatic Target Tracking Car with Monocular Camera

Huang Kaicong

22/04/2024

Acknowledgements

Thanks to Zhao Zheng for providing hardware support for this work. Thank you Sun Shuo for providing technical guidance.

Summary

This work designs a real-time automatic target tracking vehicle with a monocular camera. The system integrates YOLOv5 for object detection and TrackerKCF for continuous object tracking, optimizing performance on the NVIDIA Jetson orin nano, a resource-constrained edge computing device. The vehicle control algorithm is crafted to convert visual data into actionable navigation commands, adjusting vehicle speed and steering with PWM based on the object's position and distance. Experiments conducted validate the system's effectiveness in tracking scenarios with both straight-line and curved paths and highlight its robustness through a specialized "searching mode" that re-engages tracking after losing the target. While the vehicle demonstrates promising detection and tracking capabilities, further improvements are suggested, such as enhancing motor control for finer speed regulation and increasing structural rigidity to minimize operational vibrations. Future work may also explore the integration of advanced depth estimation technologies to reduce dependence on prior size information of targets.

Contents

Chapter 1

Introduction

1.1 Background and Problem Description

In recent years, autonomous vehicles have garnered significant attention in both academia and industry due to their potential to revolutionize transportation systems. Despite the strides made in autonomous navigation technologies, one of the enduring challenges is developing cost-effective and efficient systems for real-time object tracking using minimal hardware. This research addresses the specific need for autonomous vehicles capable of dynamically tracking objects with a single camera, which is crucial for applications in unmanned surveillance systems.

The motivation behind this initiative stems from the specific challenges associated with the monitoring and care of mobile entities such as wildlife in zoos and the elderly in assisted living environments. Traditional surveillance methods often require significant human effort and constant attention. Manual tracking and logistics support are labor-intensive and not always efficient or feasible, especially in expansive or densely populated settings. Furthermore, the growing elderly population and the expansion of large-scale wildlife conservation areas necessitate more innovative and sustainable approaches to surveillance and care.

Moreover, the autonomous vehicle is designed to perform dual functions—surveillance and transportation. This dual capability is particularly beneficial in zoo environments where the vehicle can monitor animal behavior and health, and also transport food or medical supplies across the facility. In elderly care contexts, the vehicle can help monitor residents to ensure their safety and well-being, while also delivering essentials like medication and meals directly to individuals, thereby enhancing the quality of care provided.

Deploying an autonomous vehicle equipped with a monocular camera for real-time tracking presents a cost-effective and efficient solution to these challenges. The vehicle is designed to op-

erate autonomously within designated areas, tracking animals or individuals using advanced object detection and tracking technologies. This system reduces the need for constant human presence and can potentially increase the coverage area and accuracy of surveillance operations.

The primary challenge in deploying autonomous tracking vehicles in resource-constrained environments is the computational demand of processing high-resolution video in real time. Traditional systems often rely on sophisticated sensors like LIDAR and stereo cameras, which are not only costly but also require significant computational resources that may not be feasible in smaller, budget-constrained projects. Furthermore, these systems frequently demand complex calibration and maintenance procedures. Thus, there is a compelling need for a simpler, more robust solution that can operate with lower computational requirements and hardware simplicity.

To tackle these challenges, this study explores the possibility of using a monocular camera, which significantly reduces the hardware cost and complexity. However, the use of a single camera introduces its own set of challenges, particularly in accurately detecting and tracking objects due to the lack of depth information, which is readily available in stereo camera setups.

The solution proposed in this report leverages the YOLOv5 object detection algorithm combined with the TrackerKCF tracking algorithm. YOLOv5 is chosen for its efficiency in processing and its ability to perform real-time detection with high accuracy, while TrackerKCF is utilized for its lightweight nature and effectiveness in tracking the detected objects over time. This combination promises to mitigate the computational load by decreasing the frequency of detection operations, relying on tracking to maintain the object's location in the video frame.

The overarching goal of this project is to develop an autonomous vehicle that can reliably follow a target object in a variety of environmental conditions and scenarios without requiring high-end hardware or extensive computational power. By focusing on optimizing the algorithms to run on a low-cost NVIDIA Jetson device, the project aims to make autonomous tracking technology more accessible and practical for a wider range of applications.

In a word, the main challenges addressed in the development of the automatic target tracking car equipped with a monocular camera for real-time tracking are outlined below:

Cost-Effectiveness and Efficiency in Hardware Usage. One of the primary challenges is to create systems that can perform real-time object tracking without relying on expensive and complex hardware. The research aims to develop an autonomous vehicle that uses minimal hardware, specifically a single camera, to reduce costs and complexity.

Need for Continuous and Autonomous Monitoring. In settings such as zoos and elderly care facilities, there is a significant requirement for constant monitoring, which traditionally demands

extensive human effort and attention. The autonomous vehicle must be able to operate autonomously within designated areas, reducing the need for constant human presence and increasing the coverage and accuracy of surveillance operations.

Challenges with Single Camera Systems. Utilizing a monocular camera introduces specific challenges, primarily related to the lack of depth information, which is more readily obtained with stereo camera setups. The system needs to effectively detect and track objects with this limitation, necessitating algorithmic support to compensate for the reduced sensory input.

Algorithm for Low-Cost Devices. The solution must leverage efficient algorithms that can run on low-cost hardware like the NVIDIA Jetson device. This involves apply YOLOv5 for efficient processing and real-time detection, and combining it with TrackerKCF for effective and lightweight tracking over time.

Chapter 2

Literature Review

2.1 Object Detection Technologies

Object detection is a cornerstone of autonomous vehicle navigation, involving the identification of various objects within the camera's visual field. The evolution of object detection algorithms has been significant, transitioning from traditional methods like Haar Cascades and HOG+SVM to more sophisticated deep learning models. Redmon et al. [1] introduced YOLO (You Only Look Once), a groundbreaking real-time object detection system that processes images in a single evaluation, making it markedly faster than region proposal-based methods like R-CNN [2]. The latest iteration, YOLOv5, employed in this research, enhances detection speed and accuracy, proving ideal for real-time applications [3].

2.2 Monocular Camera Systems in Tracking

The use of a single camera to track objects poses unique challenges, primarily related to depth perception, which is more straightforward in stereo vision systems. Nevertheless, researchers have developed various techniques to estimate depth from single images, leveraging geometric methods and machine learning. Engel et al. [4] described an approach using monocular visual odometry to estimate the motion of the camera reliably. This technology has been pivotal in enabling cost-effective solutions for autonomous vehicles, where maintaining a budget-friendly setup is crucial. But such The method requires too high computing power for on-board computing equipment, making it difficult for Jetson to actually run. Therefore, we simplify the depth estimation process and use prior information of camera focal length and object size to estimate depth.

Combining YOLO with tracking algorithms like TrackerKCF [5] optimizes both detection frequency and tracking stability, balancing computational load and tracking accuracy. This hybrid approach is

beneficial in dynamic environments where objects may move unpredictably. Recent studies by Zhao et al. [6] have highlighted the effectiveness of integrating deep learning-based detection with kernelized correlation filters, thus providing a robust framework for real-time tracking in autonomous vehicles. Although this method can satisfy practical-level autonomous driving systems, our task and goals are relatively simple, so there is no need to use such methods.

Chapter 3

Methodologies

3.1 Algorithms

3.1.1 Object Detection

Our visual inspection module uses the YOLOv5 algorithm[1].

YOLOv5

YOLO is a popular real-time object detection system known for its fast detection speeds and high accuracy, widely applied across various scenarios. YOLOv5 is the latest iteration in the YOLO series, optimizing speed and lightness while maintaining high performance. YOLOv5 supports multiple model sizes to accommodate different devices and requirements, boasting excellent scalability and flexibility.

YOLOv5 divides the image into multiple regions through a single neural network and predicts bounding boxes and probability scores for each region. This design allows YOLOv5 to detect multiple objects in the image in one go, significantly increasing processing speed. The model utilizes Convolutional Neural Networks (CNN) to extract features and anchor boxes to predict the size and shape of objects. With end-to-end training, YOLOv5 achieves fast and accurate object recognition. Fig.3.1[2] shows the network of one of its models.

Application in Jetson

In vision-based autonomous target tracking vehicle projects, YOLOv5 is used for real-time identification and tracking of various objects. The vehicle collects visual information from its surroundings through a camera, and the YOLOv5 model processes these images, recognizing target objects and out-

[1] https://github.com/ultralytics/yolov5/blob/master/CITATION.cff
[2] https://blog.csdn.net/qq_37541097/article/details/123594351

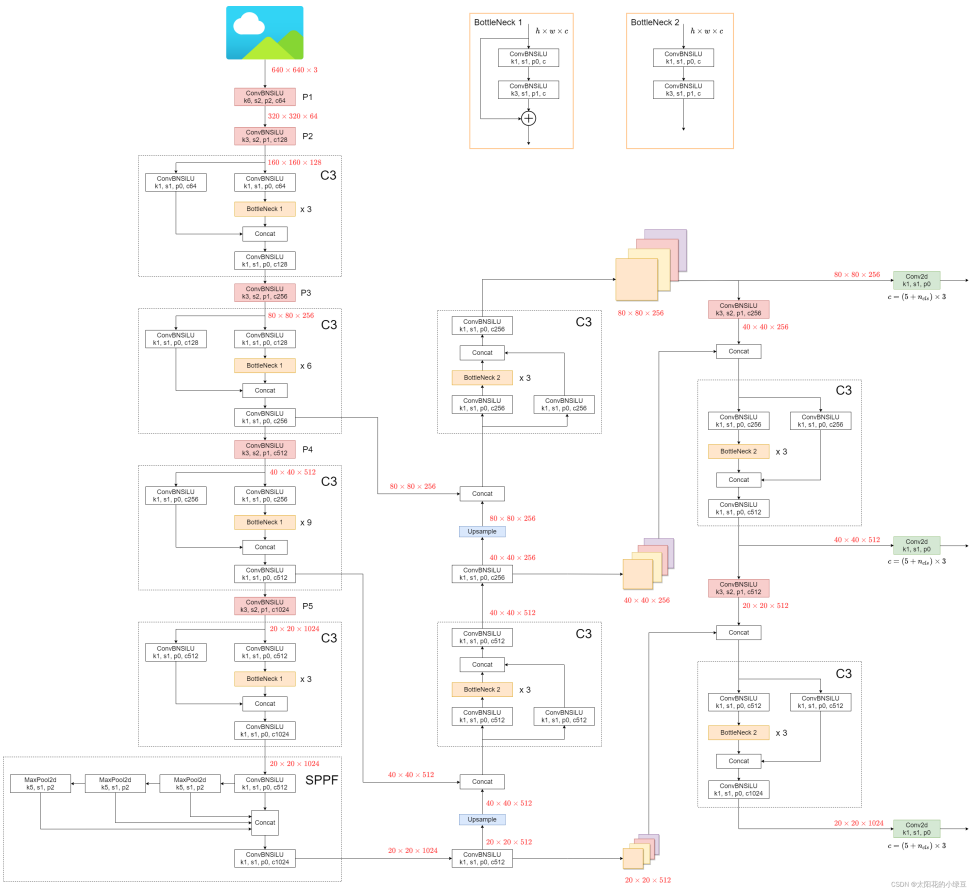

Figure 3.1: Network of YOLOv5l

putting their locations and categories. This information is then used to guide the vehicle's movement, enabling automatic tracking of specific targets.

YOLOv5 offers several pre-trained models that vary in size and performance, mainly including YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. Here is a brief comparison of these models:

-YOLOv5n: The smallest and fastest network, suitable for environments with very limited computing resources.

-YOLOv5s: Faster speed but moderate accuracy and model size, suitable for applications that require high speed and reasonable accuracy.

-YOLOv5m: Balances speed and accuracy, suitable for situations that need higher accuracy but have limited computational resources.

-YOLOv5l: Larger model size and higher accuracy, suitable for scenarios with ample computational resources.

-YOLOv5x: The largest network with the highest accuracy, suitable for applications where accuracy is a critical requirement.

Considering that NVIDIA Jetson is a resource-constrained edge computing device, choosing

YOLOv5n as the object detection model is appropriate. YOLOv5n is the lightest model in the YOLOv5 series, designed to operate on devices with limited performance. Although it sacrifices some accuracy, it significantly reduces the demand for computational resources, ensuring real-time performance on devices like Jetson.

3.1.2 Object Tracking

TrackerKCF and Implementation

TrackerKCF (Kernelized Correlation Filters) is an advanced algorithm for real-time object tracking. It operates based on the principle of correlation filters, which are trained over a cyclic matrix constructed from the initial patch around the detected object. The key idea is to find the optimal filter that yields the highest response at the object's position when correlated with the current frame.

The kernelized version of the correlation filter allows the algorithm to handle the nonlinear transformations of the object, such as changes in appearance due to rotation or illumination shifts, more effectively than basic correlation filters. This is achieved by mapping the data into a higher-dimensional space where linear separability is more feasible.

This work employs TrackerKCF by initializing the tracker with the position and size of the object detected by YOLO. The tracker then maintains the trajectory of the object across subsequent frames until the next detection cycle. This approach not only ensures continuity in tracking but also leverages the strengths of both YOLO and TrackerKCF to maintain high accuracy and efficiency.

Periodic Re-detection and Tracking

To ensure persistent and accurate tracking of a designated object, the system employs a strategy where YOLO detection and TrackerKCF[3] are used in tandem. Initially, YOLO is utilized to detect all potential objects in the first frame of every n-frame sequence. For the subsequent n-1 frames, TrackerKCF takes over to track the selected object. This method helps in minimizing the computational load by reducing the frequency of detection operations, while still ensuring that the target object is kept in focus throughout.

The tracking phase is prone to accumulating errors over time due to various factors like object occlusion, rapid movements, or changes in the environment. To counteract these errors and avoid drifting away from the target object, the system re-initiates the YOLO detection process after every n frames. This re-detection refreshes the tracking algorithm's knowledge of the object's latest position and appearance, thereby recalibrating the tracking process and enhancing accuracy. Fig.3.2 shows

[3]https://github.com/foolwood/KCF

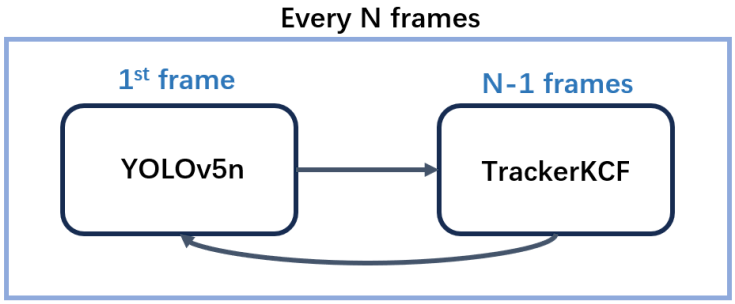

Figure 3.2: Re-detection and Tracking

the re-detection and tracking loop.

3.2 Control Algorithm

In my project, the control algorithm plays a crucial role in translating the target information detected by YOLOv5 into specific control actions for the autonomous vehicle. The algorithm dynamically adjusts the vehicle's movements based on real-time visual data, ensuring effective tracking of a designated target, such as a bottle. Here's a detailed explanation of how this is accomplished.

- **Initialization and Configuration**

  First, the necessary libraries and modules, such as PyTorch, OpenCV, and GPIO for Raspberry Pi, are imported to handle image processing, machine learning computations, and hardware control. I set up the environment for Qt platforms to ensure compatibility with the OpenCV GUI functionalities. PWM (Pulse Width Modulation) pins for controlling the servo (steering) and motor (throttle) are initialized and configured accordingly.

- **Low-Pass Filter Implementation**

  To ensure smooth control signals and mitigate rapid fluctuations in PWM values, I implement a low-pass filter for both the servo and motor controls. The filter smooths out the jitter in PWM signals, allowing for more stable vehicle movements.

- **Periodic Re-detection and Tracking**

  Using the pretrained YOLOv5n model, the system periodically scans the visual field for objects, identifying and localizing bottles with bounding boxes. When a target is detected, a TrackerKCF object tracker is initialized with the target's coordinates. Between detection intervals, the TrackerKCF algorithm tracks the target across frames.

- **Dynamic Control Adjustments**

  The control logic adjusts the servo and motor based on the position and distance of the target relative to the vehicle:

  - Distance Calculation: The distance to the target is estimated using the apparent size of the detected object and a pre-defined actual height, applying simple geometric principles with the camera's focal length.

  - Motor Control: The motor's speed is regulated based on the target's distance. If the target is within a pre-defined range, the speed is adjusted to maintain a safe following distance. This adjustment is interpolated between predefined minimum and maximum PWM values for the motor, smoothed by the low-pass filter.

  - Servo Control: The servo, which controls the steering, adjusts based on the lateral displacement of the target from the center of the camera's view. This is also smoothed using the low-pass filter to ensure gradual and stable steering adjustments.

- **Searching Mode**

  If the tracker loses the target (e.g., no detection after a certain number of frames), the system enters a search mode. In this mode, the servo is set to sweep across possible angles (using a predefined starting PWM value for the servo) and the motor is set to a slow forward speed to aid in relocating the target.

Chapter 4

Experiments

4.1 Experimental Setup

We use the NUS bear as the tracking target. As shown in Fig.4.1, the height of the bear is 27cm. We will calculate the depth of the bear from the camera based on this prior height and the focal length of the camera.

Figure 4.1: NUS bear

The camera model is Logitech c525, with a resolution of 1280*720 pixels, capable of recording 30fps video, and supports USB2.0 port. We connect it to the Jetson's USB port. The camera is installed horizontally facing forward.

Fig.4.2 is a schematic diagram of the hardware connection of the car in this project. Batteries, motor drives, motors, steering gears, etc. are placed on the site. The camera, Jetson and power supply are placed on the second layer of the 3D print. There are several holes between the upper and lower floors for wiring to pass through.

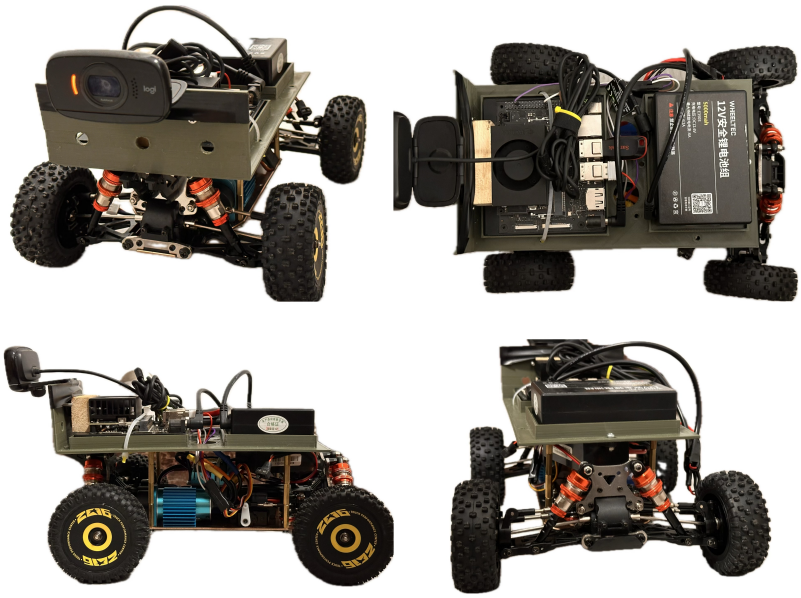

Figure 4.2: Hardware diagram of automatic tracking car

It is worth noting that because our car is rear-wheel steering, when the car needs to turn left, the wheels actually turn right, and vice versa. Fig.4.3 illustrates this logic.

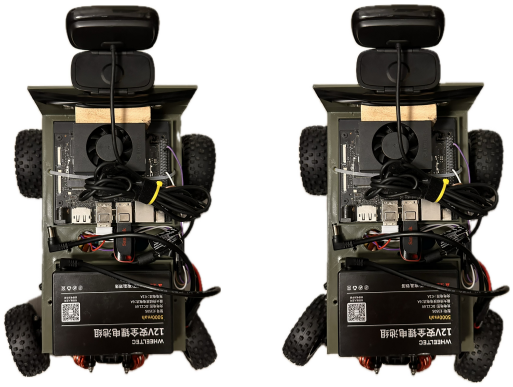

Figure 4.3: Car steering diagram. In the picture on the left, the wheel turning left corresponds to the car turning right. In the picture on the right, the wheel turning right corresponds to the car turning left.

In 'Re-detection and Tracking', set the number of loop frames n=10. And when the camera does not detect the target for 60 consecutive frames, it enters searching mode.

Set the minimum target distance to 70cm. When the target is smaller than this distance, the car will stop. When the target is greater than this distance, the car will start to move forward. The maximum target distance is 2.0m. When the target exceeds this distance, the car will move forward at the maximum speed. When the target is between these two distances, the moving speed of the car (equivalent to the duty cycle of PWM) is proportional to the target distance.

Set the motor PWM range to 60-63% in 1000Hz and the servo PWM range to 3-9% in 50Hz.

4.2 Results and Discussion

4.2.1 Static Testing

During the static test, the cart is suspended on the table to test the integrity of various functions. Fig.4.4 shows the control of the car under different situations. When the bear is on the left side of the screen, the wheel turns to the left, and the steering amplitude is proportional to the distance between the center point of the bear detection frame and the center of the image. When the bear is on the right, the wheels turn to the right. When the bear is greater than the minimum detection distance from the camera, the car wheels start to rotate. When less than this distance, the car will stop.

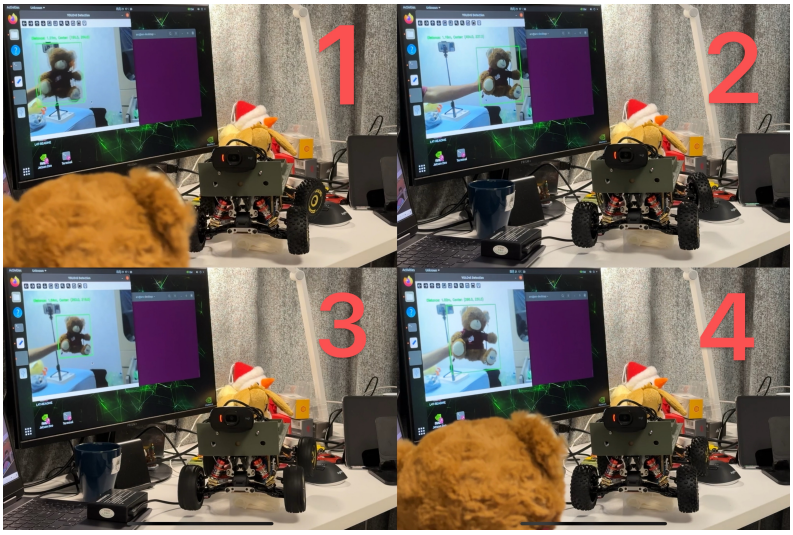

Figure 4.4: Static testing. 1-4 respectively show the situation where the target is on the left side of the screen, the right side of the screen, the target distance is farther, and the target distance is closer.

4.2.2 Tracking the NUS Bear

The car was placed on the ground for the bear tracking test. The following shows two sections of the tracking test, namely straight line tracking and curve tracking. Fig.4.5 shows straight line tracking, with four images arranged in chronological order from left to right. The first image is the initial state. In the second image, we move the bear to a farther distance. It can be seen that the car can move forward and stop at an appropriate location when it finds that the target distance is greater than the minimum distance.

Fig.4.6 shows curve tracking. Tthe car oversteers in the second image. This is because when the car detects that the target is on the right side of the screen, it will move forward to the right, but the accelerator suddenly gives too much. Our algorithm does not correct this problem, but the car is able to return to the correct route in the subsequent time and correct this error. This demonstrates the good robustness of our algorithm.

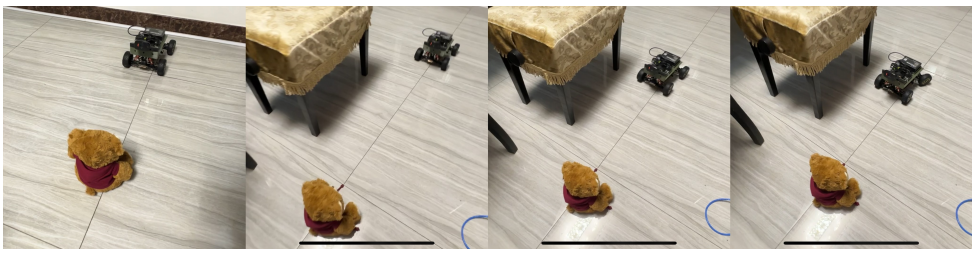

Figure 4.5: Straight line tracking

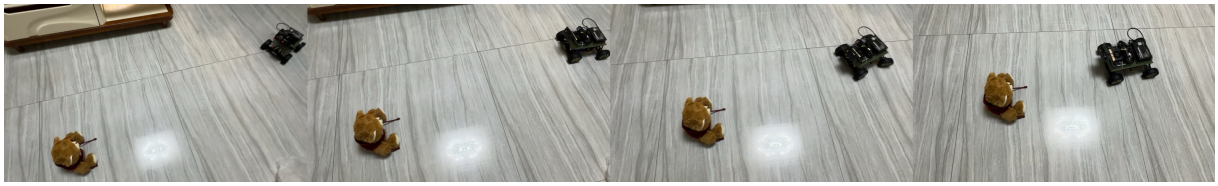

Figure 4.6: Curve tracing

4.2.3 Searching Mode

In the experiment, the number of frames when the car enters the searching mode is set to 60 frames. That is, when the camera cannot detect the target for 60 consecutive frames, the car automatically enters the search mode at a constant speed and the trajectory is a circle. In this process, as long as the camera detects the target at any frame, the car will launch the search and move towards the target. Fig.4.7 shows the experiment in this mode.

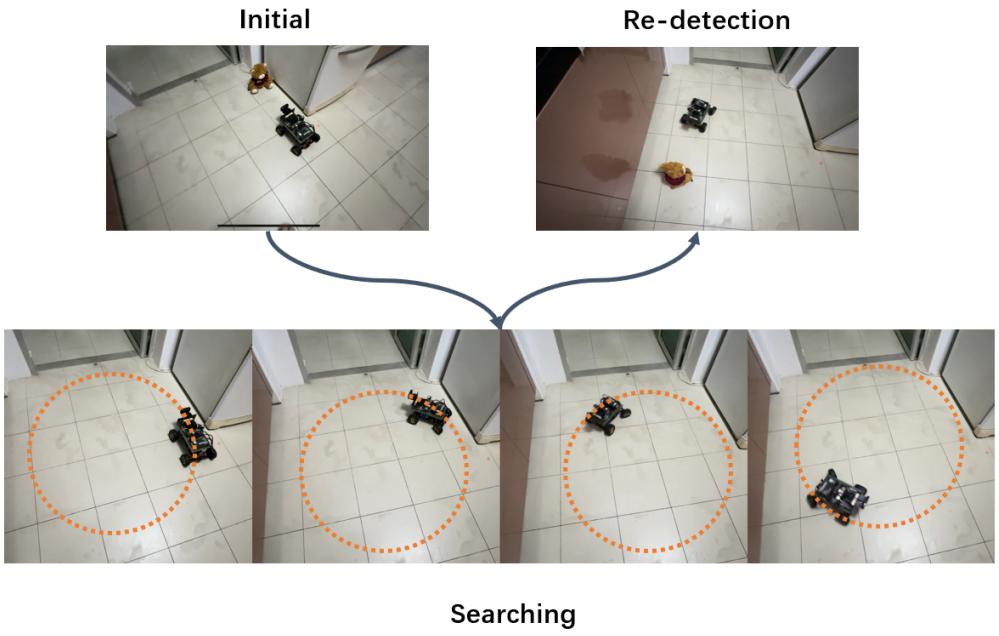

Figure 4.7: Searching Mode. In the initial state, the car can detect the target. After the target is taken away and the camera cannot detect the target for 60 consecutive frames, the car enters the searching mode and searches for the target along the orange dotted circle trajectory until the target is detected again.

Chapter 5

Conclusions

In this work, we design a car based on Automatic target tracking with monocular camera. Only relying on the image input of the RGB camera and the prior information of the target, the car is controlled to track the target in real time.

For the detection part, YOLOv5, a scalable and efficient object detection system, is utilized to identify objects in real-time. It predicts bounding boxes and probability scores across the image, enabling simultaneous detection of multiple objects. Various YOLOv5 models are evaluated for their computational efficiency. Based on the In order to run in real time on Jetson, we chose the more lightweight YOLOv5n.

For the object tracking, YOLOv5 is integrated with TrackerKCF, a tracking algorithm based on kernelized correlation filters, to maintain continuous object tracking. This approach combines initial detection with subsequent frame tracking, minimizing the frequency of detection and reducing computational demands. Periodic re-detection is implemented to recalibrate tracking, correcting any drifts and ensuring consistent accuracy.

For the control algorithm, we translate the positional data from the detected object into vehicle steering and throttle adjustments. Motor and servo PWM are dynamically adjusted based on object distance and lateral position, smoothed by low-pass filters. A "searching mode" is activated if the target is lost, enabling the vehicle to sweep the area and re-detect the target.

These methodologies are validated through various experiments, demonstrating the system's capability in static and dynamic tracking scenarios, including straight-line and curve tracking. The effectiveness of the "searching mode" is showcased, emphasizing the vehicle's ability to reinitiate pursuit after losing the target.

Chapter 6

Recommendations for Future Work

Although the car demonstrated good detection and tracking performance in the experiment, there are still some problems in this work.

The motor runs too fast. Since the car used in this work is modified from a remote control toy off-road vehicle, the original maximum speed of the motor can reach 70km/h, so the extremely small pwm duty cycle can cause the motor to generate a large speed. And the motor has a large dead zone. At 1000Hz, the motor will only rotate when the pwm reaches a duty cycle of 60%, and it can reach a very high speed at 63%. Our tracking task requires very low motor speed, and we hope that the motor PWM adjustment dead zone should be as small as possible. Thus, he motor was quite unsuitable for our task. It can be seen that in the experiment, our car often appeared in the "accelerate, stop" state. This is because the pwm frequently switches between 60-62%. Therefore, other motors may need to be replaced in the future to achieve more stable output.

Insufficient structural rigidity. Due to the poor rigidity of the 3D printed structure and other connecting parts, the car will inevitably vibrate during operation, which is not conducive to the camera shooting. Frequent jitters will increase the error rate of detection, causing the car to receive wrong control information. Therefore, in the future, we may consider replacing the entire vehicle with integrated printed parts to increase structural stability.

Prior information about the target object is required. Since the monocular RGB camera is the only sensor, it cannot obtain depth information in the picture. To this end, it is necessary to know the a priori size of the target object and combine it with the focal length of the camera to calculate the depth, which has weak generalization. In the future, RGBD cameras may be considered, or combined with lidar to directly obtain the depth of the target object and the surrounding environment. Or use the deep learning method of monocular depth estimation, but this is a difficult challenge for the Jetson.

Bibliography

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[2] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jehad Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[3] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, Liu Changyu, Adam Hogan, Laurentiu Diaconu, Kartik Seethepalli, and Michael Foley. Yolov5: State of the art object detection at real-time speed. *GitHub repository*, 2020.

[4] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 40, pages 611–625, 2017.

[5] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 37, pages 583–596, 2015.

[6] Zhe Zhao, Ling Shao, Jian Xie, and Yiyi Lu. Deep learning and kernelized correlation filters for real-time object tracking in autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 21(4):1531–1540, 2019.