International Core Journal of Engineering Volume 12 Issue 2, 2026
ISSN: 2414-1895 DOI: 10.6919/ICJE.202602_12(2).0015

Design and Implementation of a Laser Combat Smart Car Integrating Target Tracking and Obstacle Avoidance

Long Yang, Zhigang Di

College of Electrical Engineering, North China University of Science and Technology, Tangshan 063210, China

Abstract

Laser combat tasks require high dynamic target-tracking accuracy, fast real-time response, and reliable obstacle-avoidance capability in complex environments. However, in conventional intelligent vehicle systems, coordination among target tracking, motion control, and obstacle-avoidance decision-making is often degraded by response delays and control conflicts, thereby reducing pointing stability and task completion rate. To address these issues, a laser combat smart car system integrating target tracking and obstacle avoidance was designed and implemented. A hierarchical cooperative architecture based on a Raspberry Pi 5B and an STM32F407 was adopted: grayscale preprocessing was performed on the Raspberry Pi side, and target detection was completed using a YOLOv8 model to output the pixel deviation of the target center; the deviation data were received by the STM32 via a serial interface, and an "obstacle-avoidance-first" motion-control strategy was implemented by fusing ultrasonic ranging, while a dual-axis pan–tilt mechanism was driven to coordinate laser pointing with vehicle motion. Experimental results showed that the system operated stably at a resolution of 640 × 480 and achieved real-time target tracking and pointing control. In multi-obstacle scenario tests, an obstacle-avoidance success rate of over 95% was obtained, and the laser pointing deviation was maintained within a small range. These results indicated that good real-time performance and stability were achieved, providing a reference for the engineering implementation of laser combat mobile platforms.

Keywords

Laser Countermeasure; Intelligent Vehicle; YOLOv8; Target Tracking; Obstacle Avoidance Control; Gimbal Control.

1. Introduction

In recent years, laser countermeasure and intelligent control technology have become one of the research hotspots in the field of intelligent equipment, especially with broad application prospects in military training, directional jamming, unmanned driving, express sorting and other fields[1]. With the development of mobile platforms and intelligent control technology[2], integrating laser actuators with ground intelligent vehicles to achieve rapid pointing and stable action on dynamic targets through perception, decision-making and control can reduce the burden of manual aiming and tracking, and improve the efficiency of task response and execution consistency, which has certain practical significance.

However, in actual systems, "being visible" does not equal "being accurate": the output of visual recognition is image coordinate error, while the execution end needs to convert the error into chassis movement and gimbal pointing; at the same time, the mobile platform also needs to balance safe obstacle avoidance and real-time performance in complex environments. Therefore, the collaboration

ISSN: 2414-1895 DOI: 10.6919/ICJE.202602_12(2).0015

between visual tracking, attitude control and obstacle avoidance strategies has become a key issue in system implementation.

Based on the above requirements, a "layered architecture, hardware-software collaboration" intelligent vehicle system is designed and implemented. While ensuring hardware reliability, it improves intelligent control efficiency, which has important practical value for enhancing the task completion rate of laser countermeasure intelligent equipment. The system uses Raspberry Pi as the visual processing unit to perform target recognition on camera images through YOLOv8, and extracts the center deviation of the white diffuse reflection area in the image as the control quantity; uses STM32F407VGT6 as the motion and execution control unit to complete functions such as four-wheel chassis driving, two-dimensional pan-tilt servo control, and ultrasonic obstacle avoidance.

2. Overall System Design

To meet the requirements of laser combat scenarios, the system needs to implement three core functions: target recognition and positioning (vision module), omnidirectional movement (chassis module), laser aiming and emission (gimbal + laser), and an obstacle avoidance (ultrasonic) guarantee function. For this purpose, a three-layer hierarchical architecture of "perception-control-execution" is adopted, and the engineering design concept of modular decoupling is integrated throughout. The perception layer includes an ultrasonic obstacle avoidance module responsible for obstacle detection and distance signal transmission, and a camera module controlled by Raspberry Pi 5B to achieve target recognition and positioning. The control layer is centered on the STM32 module, which is responsible for receiving data from the perception layer and calculating control commands. The execution layer includes a motor module driven by TB6612 to control motor rotation, a two-dimensional gimbal module to drive the camera rotation, and a laser emitter module for laser combat. Each layer module has independent functions and works collaboratively through standardized interfaces[3], realizing the optimization of system performance and the improvement of maintainability. The system structure block diagram is shown in Figure 1.

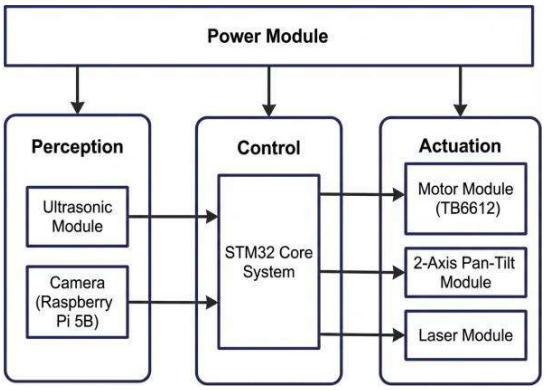

Figure 1. System Structure Block Diagram

3. Hardware System Design

3.1 Main Control Board

This system adopts the hardware integration method of "self-developed mainboard (carrier board) + STM32F407VGT6 core board" as shown in Figure 2: the STM32 core board is responsible for computing and peripheral control, and the self-developed mainboard is used to complete power management, interface expansion and module integration. The mainboard integrates the MP1584EN step-down power supply and reserves pin headers/interfaces for functional module connection; two TB6612 motor driver boards are fixed through the mainboard pin headers and uniformly routed by the mainboard to the PWM and direction control pins of STM32 to realize PWM speed regulation and forward/reverse control of the four-wheel motor. The Raspberry Pi and STM32 are connected

ISSN: 2414-1895 DOI: 10.6919/ICJE.202602_12(2).0015

through the UART TTL serial port pin headers on the mainboard to realize data transmission from the visual end recognition results to the control end. Through the above method, the mainboard plays the role of a "power supply + interface + wiring" platform, making the connection between the core board, driver board and peripherals more reliable, and improving the convenience of vehicle assembly and maintenance.

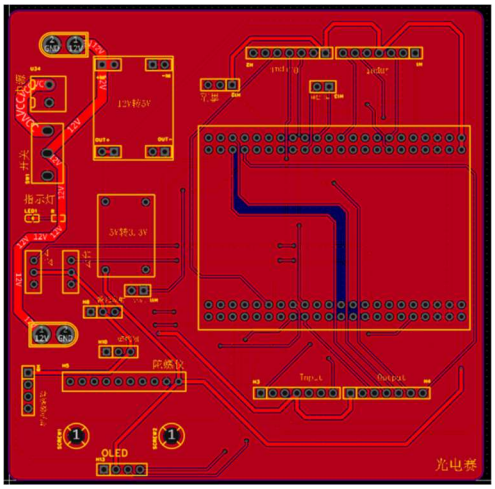

Figure 2. Mainboard PCB

For the buck circuit, the main board uses the 12V bus as input and adopts two-stage MP1584EN synchronous rectification buck: the first stage steps down 12V to 5V to supply peripherals such as Raspberry Pi and TB6612; the second stage further steps down 5V to 3.3V to supply STM32 logic circuits and related peripherals. Its core chip MP1584EN supports a wide input range of 4.5V to 28V and a maximum output current of 3A[4], which meets the current demand of the system when the motor and peripherals work simultaneously.

For the main control board's core chip STM32F407VGT6 (ARM Cortex-M4 core), the chip integrates various types of timers and common bus interfaces internally[5]. This system mainly uses its timer PWM output to realize motor and servo control, uses the timer encoder mode to complete wheel speed counting, uses UART to realize data communication with the Raspberry Pi, and completes ultrasonic ranging through GPIO/timer timing to support obstacle avoidance control. The block diagram of its core board is shown in Figure 3.The above-mentioned peripheral resources can meet the real-time requirements of the system in tasks such as visual pointing, chassis motion control and obstacle avoidance, and provide an interface foundation for subsequent function expansion.

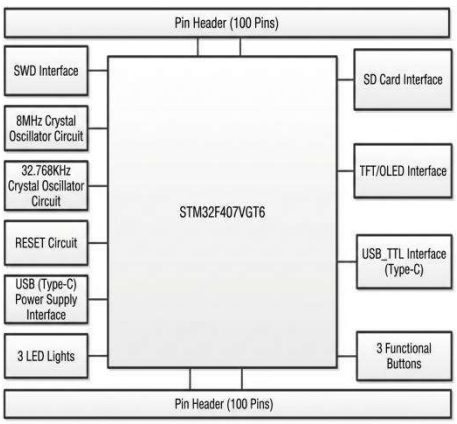

Figure 3. Block Diagram of STM32F407VGT6 Core Board

ISSN: 2414-1895 DOI: 10.6919/ICJE.202602_12(2).0015

3.2 Gear Motor

The chassis drive unit uses a 520-series DC motor with a gearbox (equipped with an incremental encoder), and the selection meets the needs of the laser combat scenario: (1) Torque and acceleration margin: The Mecanum wheel chassis has higher requirements for wheel-end torque under lateral/oblique movement conditions. A reduction ratio of approximately 1:50 is selected to balance the output speed while increasing the wheel-end torque margin, so that stable starting, braking and steering can still be maintained after mounting upper components such as a pan-tilt and camera[6]. (2) Structural and installation adaptation: The 520 motor has a compact shape and a universal installation form, which is convenient for four-wheel symmetrical arrangement and subsequent maintenance and replacement. (3) Closed-loop control scalability: The incremental encoder provides wheel speed feedback, which can be used for wheel speed closed-loop control, odometer estimation and kinematic solution, improving the speed consistency and trajectory stability of omnidirectional movement.

The motor is connected to the TB6612FNG motor driver board via a 6PIN interface, where the two wires of the motor are driven by the TB6612 H-bridge output; the encoder A/B signals are led out to independent Output IO pins on the driver board and connected to the STM32 timer encoder interface to realize wheel speed and direction measurement[7].

In addition, this system uses the A/B signals output by the incremental encoder to read the timer counter increment ΔN within a fixed sampling window Δt, and the rotational speed is calculated as:

$$n = \frac{\Delta N}{k \cdot PPR \cdot \Delta t} \times 60 \quad (1)$$

where ΔN is the count increment within the sampling window Δt; PPR denotes the number of base pulses per revolution of the encoder; and K is the counting/decoding factor: k=1 when only the rising edge of channel A is counted, k=2 when both edges of channel A are counted, and k=4 when 4× quadrature decoding of channels A/B is used. In this system, the STM32 timer encoder mode is adopted and thus k=4. The phase lead/lag relationship between channels A and B is used to determine the rotation direction, thereby enabling closed-loop wheel-speed control and odometry accumulation.

3.3 TB6612 Motor Driver

The chassis of this system adopts 4 DC gear motors with incremental encoders. To achieve independent speed regulation and forward/reverse control for the four wheels, two TB6612FNG dual H-bridge motor driver boards are selected: each driver board drives two motors simultaneously, and the two boards drive the four wheels in total. The control method of TB6612 is as follows: PWM pulse input determines the speed (duty cycle speed regulation), IN1/IN2 combination determines the direction, and STBY is the enable terminal[8]. This control logic is consistent with the description in Table 1 of the TB6612 working truth table, and can realize modes such as forward rotation, reverse rotation, braking and stopping.

Table 1. TB6612FNG Truth Table

PWM	IN1	IN2	Function
1	0	0	Brake stop
1	1	0	Forward rotation
1	0	1	Reverse
1	1	1	Brake stop
0	X	X	stop

ISSN: 2414-1895 DOI: 10.6919/ICJE.202602_12(2).0015

To avoid function multiplexing conflicts such as PWM/encoder/communication, this design allocates pins according to the principle of "PWM → timer channel, encoder → timer encoder interface, direction → GPIO": Driver Board 1 is responsible for motors A/B, with PWMA → PB1, PWMB → PB0; direction terminals AIN1 → PB14, AIN2 → PB15, BIN1 → PB13, BIN2 → PB12; encoder inputs are Motor A: PB6/PB7, Motor B: PA15/PB3. Driver Board 2 is responsible for motors C/D, with PWMC → PA11, PWMD → PA10; direction terminals AIN3 → PD14, AIN4 → PD15, BIN3 → PD13, BIN4 → PD12; encoder inputs are Motor C: PA0/PA1, Motor D: PC6/PC7. The STBY pins of both driver boards are connected to 3.3V to maintain enablement. In addition, ADC sampling of the driver input voltage is performed through PA6 and PA7 for status monitoring. AO1/AO2 and BO1/BO2 are two motor output terminals, which are respectively connected to both ends of the corresponding DC motors.

3.4 Raspberry Pi 5B

In the visual processing section, the system uses Raspberry Pi 5B + USB camera as the upper computer visual processing unit for image acquisition and processing. The Raspberry Pi collects 640×480 resolution images through the USB interface as visual input; it is cross-connected with the lower computer STM32 via UART TTL serial port (Raspberry Pi TX→STM32 RX, Raspberry Pi RX→STM32 TX), and the relevant signals are led out by the motherboard pins. The Raspberry Pi is powered by the system's 5V power supply and shares the ground with the STM32 and drive modules to ensure consistent communication level reference and stable operation of the entire vehicle.

Raspberry Pi 5B is based on the Linux system, with good edge computing capabilities and peripheral expansion capabilities, meeting the operational requirements of visual algorithms[9]. The Raspberry Pi runs a YOLOv8-based visual processing process, outputs the deviation (dx, dy) of the center of the white diffuse reflection area relative to the image center, and sends it to STM32 via serial port as control input.

3.5 MG996R 2D Gimbal

This system installs a 2D gimbal on the top of the vehicle body to carry the camera and laser, realizing the pointing adjustment of two degrees of freedom: horizontal (Yaw) and pitch (Pitch), so as to align with the white diffuse reflection area in cooperation with the visual target deviation. The gimbal is driven by an MG996R digital servo, which can achieve approximately 180° horizontal rotation and 180° pitch adjustment, meeting the rapid pointing requirements in laser countermeasures.

In terms of hardware connection, the servo uses a three-wire system: the red wire connects to 5V, the brown wire connects to GND, and the signal wire is controlled by PWM output from STM32; in this design, the Yaw signal is connected to PE9 and the Pitch signal to PE11. The servo adopts the standard 50Hz control method (20ms period), and the target angle is determined by the high-level pulse width. The pulse width-angle relationship is converted and limited according to the range given in Table 2[10].

Table 2. Relationship between PWM Pulse Width and Servo Rotation Angle

Pulse width (duration of high level)	Corresponding angle
0.5ms	0°
1ms	45°
1.5ms	90°
2.0ms	135°
2.5ms	180°

ISSN: 2414-1895 DOI: 10.6919/ICJE.202602_12(2).0015

3.6 Ultrasonic Module

The system uses the HC-SR04 shown in Figure 4 as a short-range obstacle avoidance sensor. HC-SR04 consists of transmitting and receiving transducers, which can measure distance by emitting ultrasonic waves and receiving echoes. It has the advantages of low cost, simple implementation, and stable short-range ranging, making it suitable for indoor obstacle avoidance scenarios[11].

The main pins of the HC-SR04 module include VCC, GND, Trig, and Echo. The connection between the module and the main controller is as follows: Trig, as the trigger terminal, is connected to the PA3 pin (the trigger pulse is output by STM32), Echo, as the echo output terminal, is connected to the PE6 pin (the high-level pulse width of the echo is measured by STM32), and they share the ground with the system's 5V/GND.

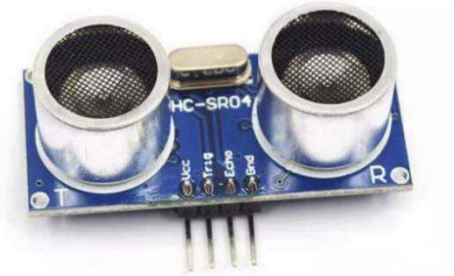

Figure 4. HC-SR04 Ultrasonic Module

4. Software System Design

To achieve the function of "continuous target tracking, countermeasure, and obstacle avoidance" for the entire system, the software system adopts a hierarchical collaborative architecture of "Raspberry Pi visual processing + STM32 execution control". The Raspberry Pi side is responsible for image acquisition and target recognition, extracting the deviation (dx, dy) of the center of the white diffuse reflection area relative to the image center, and sending it to the STM32 as a control input; after receiving the visual deviation, the STM32 side completes the pan-tilt pointing and chassis motion control by combining the encoder wheel speed feedback and ultrasonic ranging information. The system as a whole takes visual tracking as the main control basis. When an obstacle ahead is detected, obstacle avoidance control is prioritized, and visual tracking is resumed after obstacle avoidance is completed, thereby achieving coordination between target pointing and driving safety.

4.1 Visual Part

For visual processing, the Raspberry Pi uses Python for programming and data processing, and the specific visual processing flow is shown in Figure 5. After the program starts, initialization is completed, including USB camera acquisition parameters (OpenCV library normalization)[12], grayscale threshold parameters, YOLOv8 (ONNX) inference model loading, and UART TTL serial communication configuration with STM32 (TX/RX cross-connected and common ground). During operation, a frame of image is read cyclically, and first, grayscale threshold pre-screening is performed using the characteristic that the white diffuse reflection area has high brightness: generate a mask to suppress non-white areas; if the proportion of effective highlight areas is too small, skip this frame and continue acquisition. The images that pass the pre-screening are subjected to YOLOv8 inference, and this frame is discarded when the confidence is insufficient[13]; calculate the center coordinates (x_c, y_c) of the white area from the valid detection results, and compare them with the image center $(W/2, H/2)$ to obtain the pixel deviation:

$$dx = x_c - \frac{W}{2} \tag{2}$$

ISSN: 2414-1895 DOI: 10.6919/ICJE.202602_12(2).0015

$$dy = y_c - \frac{H}{2} \tag{3}$$

Finally, send (dx,dy) to STM32 via UART TTL serial port as the control input for the gimbal and chassis.

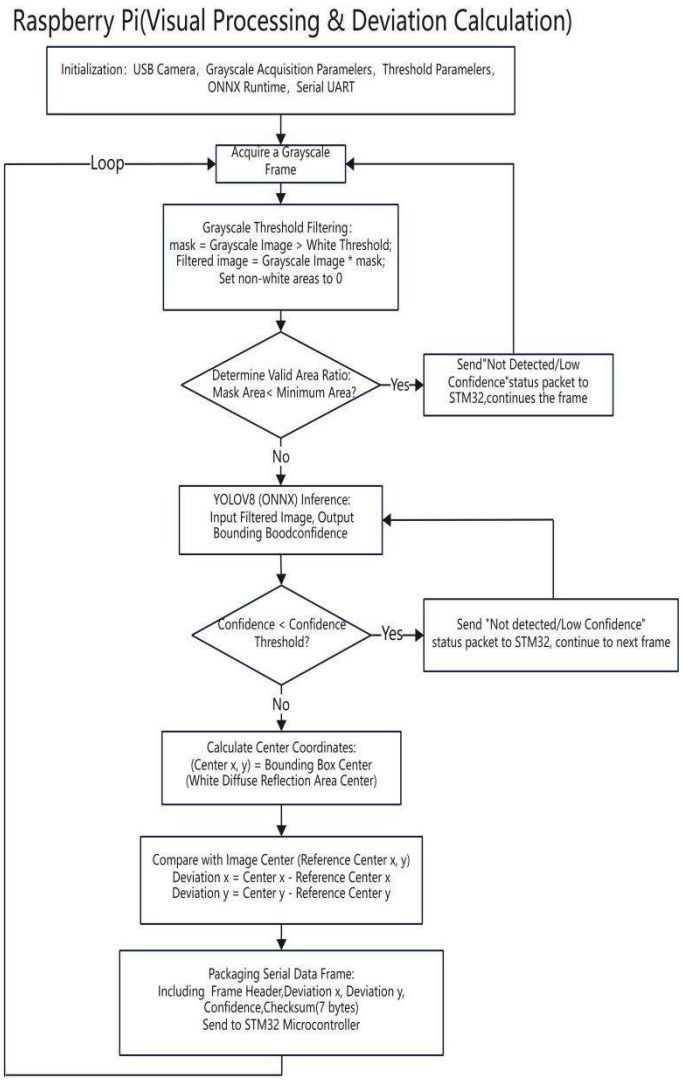

Figure 5. Flowchart of the Visual Part

4.1.1 Gray Threshold Filtering

The white diffuse reflection area has the characteristics of high brightness and large gray value. To reduce the interference of complex backgrounds on detection and decrease the model's attention to invalid areas, this system adds a pre-screening based on gray threshold before target detection: converting the input image into a grayscale image G(x,y), generating a binary mask M(x,y) through threshold T, and then applying the mask to the original three-channel image to black out non-white areas, thereby highlighting suspected white target areas[14]. The mask is defined as:

$$M(x,y) = \begin{cases} 1, & G(x,y) \geq T \\ 0, & G(x,y) < T \end{cases} \tag{4}$$

On this basis, the filtered image is represented as:

ISSN: 2414-1895 DOI: 10.6919/ICJE.202602_12(2).0015

$$I_f(x,y) = I(x,y) \cdot M(x,y) \quad (5)$$

Where $I(x,y)$ is the pixel of the original color image. This preprocessing belongs to per-pixel threshold calculation, with a time complexity of O(H,W) (H and W are the height and width of the image). It is simple in operation and suitable for real-time processing.

In implementation, to ensure consistency with the YOLOv8 inference input format, this paper applies a mask to the three-channel image (maintaining the BGR/RGB three channels) instead of directly using the single-channel grayscale image as the model input. The key code is as follows:

```
def threshold_white_area(bgr, t_white: int = 200):
# 1 Grayscale Conversion
gray = cv2.cvtColor(bgr, cv2.COLOR_BGR2GRAY)
# 2 Threshold segmentation to obtain mask
_, mask = cv2.threshold(gray, t_white, 255, cv2.THRESH_BINARY)
# 3 Apply a mask to the three-channel image (ensure consistency with YOLO input)
filtered = cv2.bitwise_and(bgr, bgr, mask=mask)
return filtered, mask
```

4.1.2 Single-class YOLOv8 Training and Inference

This system uses YOLOv8 to detect single-target white diffuse reflection areas. The training phase completes dataset annotation and model training on the PC, with the input size set to 416×416, batch size to 4, 100 iterations, initial learning rate lr0 to 0.001, and the ONNX model is exported after training[15]. The key code is shown below:

```
model.train(
        lr0=0.001,
        data=DATA_YAML,
        imgsz=416,
        epochs=100,
        batch=4,
        device=DEVICE,
        workers=0,
        project="runs_white_reflect",
        name="exp",
               )
```

During the inference phase, the Raspberry Pi uses Python to call the ONNX model for forward inference, outputting candidate detection boxes and their confidence levels; the results are filtered using a confidence threshold to obtain the final target box center coordinates(x_c, y_c), and then the pixel deviations (dx,dy) are calculated according to equations (2) and (3) before being sent via serial port.

4.2 STM32 Control and Execution Software Design

The STM32F407VGT6 core chip is the key to the hardware control part. Pin assignment is carried out based on STM32CubeMx, and the system clock, external clock, and interrupt can be configured in sequence, and initialization code can be generated[16]. At the same time, the keil μVision5 environment is used for specific program writing, and modular programming and design are performed on the hardware part of the smart car, including the ultrasonic obstacle avoidance module, motor drive module, pan-tilt rotation module, serial communication module with Raspberry Pi, and main program module. In code implementation, the priority logic of "obstacle avoidance first, continuous tracking and striking" is adopted, continuously issuing instructions to the motor and 2D pan-tilt, and adjusting the running speed and state of the smart car at all times. The specific flow chart is shown in Figure 6.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202602_12(2).0015

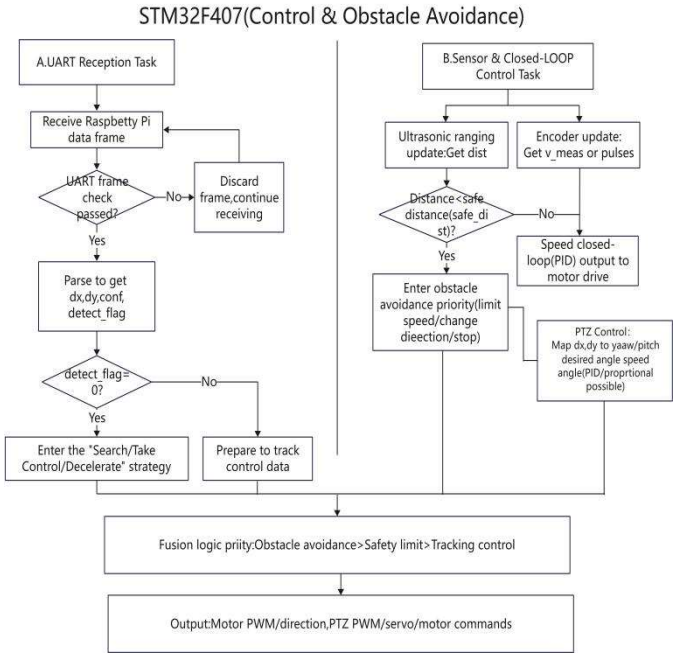

Figure 6. Control and Execution Software Design Flowchart

4.2.1 Serial Communication Program

STM32F407 and Raspberry Pi communicate via UART TTL serial port (baud rate 115200, 8N1)[17], and the communication data adopts a fixed-length binary frame format. Each frame is 7 bytes long, with the frame header being 0xAA and 0x55, followed by the low/high bytes of dx and dy in sequence, and a 1-byte checksum (sum of the first 6 bytes). The byte corresponding relationship is shown in Table 3.

Table 3. Byte Position and Variable Correspondence

Byte position	Variable Correspondence
buf[0]	Frame Header 1
buf[1]	Frame Header 2
buf[2]	dx Lower 8 Bits
buf[3]	dx Higher 8 Bits
buf[4]	dy Lower 8 Bits
buf[5]	dy Higher 8 Bits
buf[6]	Checksum

The STM32 side uses UART receive interrupt to obtain data, and the protocol parsing adopts a frame header matching state machine: first search and lock the 0xAA and 0x55 frame headers, then cache subsequent bytes until 7 bytes are satisfied before verification; when verification fails, discard current data and re-search the frame header to complete resynchronization. dx and dy are transmitted in int16 format, with the unit being pixel deviation. Part of the code for the parsing function is shown below:

```
VisionMsg_t Proto_TryParse(void)
{
    VisionMsg_t o = {0};
    if (idx < 7) return o;  //Frame Header Check
    uint8_t sum = 0;
```

ISSN: 2414-1895 DOI: 10.6919/ICJE.202602_12(2).0015

```
    for (int i=0;i<6;i++) sum += buf[i];
    if (sum != buf[6]) { idx = 0; return o; }   //Checksum Calculation and Verification
    o.dx = (int16_t)((buf[3]<<8) | buf[2]);          //dx Analysis
    o.dy = (int16_t)((buf[5]<<8) | buf[4]);          //dy Analysis
    o.valid = 1;
    idx = 0;
    return o;
}
```

4.2.2 Motor Drive Program

The direction of motor drive is determined by the signals of IN1/IN2, and PWM controls the duty cycle to adjust the motor speed. For interface unification, this article defines the control variable as signed PWM: the sign is used to select the direction combination of IN1/IN2, and the amplitude is used to set the PWM duty cycle. When PWM=0, the motor enters stop/brake mode to avoid meaningless switching of direction pins at zero duty cycle. The key code is as follows:

```
if (pwm > 0)      //Forward rotation
{
HAL_GPIO_WritePin(m->in1_port,m->in1_pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(m->in2_port,m->in2_pin, GPIO_PIN_RESET);
}
else if(pwm < 0)       //Reverse
{
HAL_GPIO_WritePin(m->in1_port,m->in1_pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(m->in2_port,m->in2_pin, GPIO_PIN_SET);
pwm=-pwm;
}
else          //pwm==0,Brake Stop
{
HAL_GPIO_WritePin(m->in1_port, m->in1_pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(m->in2_port, m->in2_pin, GPIO_PIN_RESET);
}
```

4.2.3 PTZ Rotation Program

PTZ control takes the target pixel deviation (dx, dy) output by the vision module as input and maps it proportionally to the PTZ angle adjustment amount:

$$\Delta\theta_{pan} = k_x \cdot dx \tag{6}$$

$$\Delta\theta_{tilt} = k_y \cdot dy \tag{7}$$

and further linearly mapped to the servo control pulse width:

$$t_{pan} = t_{pan,0} + \alpha \cdot \Delta\theta_{pan} \tag{8}$$

$$t_{tilt} = t_{tilt,0} + \alpha \cdot \Delta\theta_{tilt} \tag{9}$$

ISSN: 2414-1895 DOI: 10.6919/ICJE.202602_12(2).0015

Among them k_x, k_y, α is the proportional coefficient, which is adjusted through experiments;$t_{pan,0}, t_{tilt,0}$ is the neutral position pulse width of the steering gear.

When the target deviates from the center of the image, the PTZ rotates accordingly based on the direction and magnitude of the deviation, gradually bringing the target back to the center of the image. Since the servo motor is controlled by PWM signals, and angle adjustment is achieved by adjusting the high-level pulse width, the system uses TIM6 as the time base and generates PWM control signals through counting in the timer interrupt: at the start of each control cycle, the pins of the horizontal axis and pitch axis servos are set high; when the count values reach the target pulse widths t_{pan}, t_{tilt} respectively, the corresponding pins are set low; after the cycle ends, the count is cleared and the next cycle begins, completing continuous servo control output.

To prevent the servo from overstepping and jittering,t_{pan}, t_{tilt} limit the pulse width to the range of 0.5-2.5 ms.[18].

4.2.4 Ultrasonic Ranging Obstacle Avoidance Program

The working process of the ultrasonic ranging module is shown in Figure 7. The system first outputs a trigger pulse by controlling the Trig pin to start ultrasonic transmission; then the module outputs a high-level signal proportional to the echo flight time on the Echo pin.

During the ranging process, the system measures the high-level pulse width of the Echo pin, records the duration of the Echo signal from high to low through a timer, and calculates the target distance based on the measured pulse width. When the calculated distance is less than the set threshold, the system determines that there is an obstacle ahead and enters the obstacle avoidance control logic; otherwise, it maintains the normal motion state.

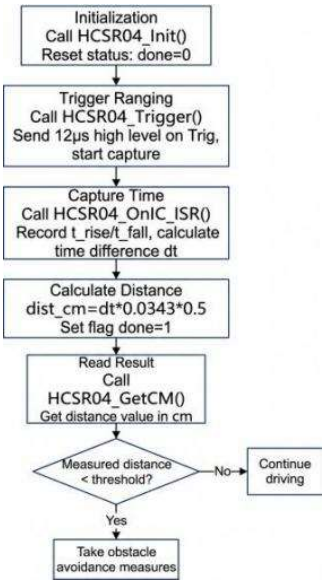

Figure 7. Flowchart of HC-SR04 Ranging and Obstacle Avoidance

4.2.5 Main Function Program

The main program flow is shown in Figure 8. The system's main function adopts a cyclic execution mode, and in each control cycle, tasks such as visual data parsing, ultrasonic ranging and obstacle avoidance judgment, and chassis and pan-tilt control output are completed in sequence.

In the main loop, the system first receives and parses the serial port data from the visual module to obtain the pixel deviation of the target in the image; then reads the ultrasonic ranging result to determine whether there is an obstacle ahead; finally, according to the results of visual tracking and obstacle avoidance judgment, outputs the corresponding chassis motion control commands and pan-tilt control signals to realize target tracking and autonomous obstacle avoidance functions.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202602_12(2).0015

In terms of control logic, the system adopts an arbitration strategy with obstacle avoidance priority: when the ultrasonic ranging result is less than the set threshold, the chassis prioritizes obstacle avoidance control, and the pan-tilt tracking output is temporarily frozen at this time; when the obstacle avoidance condition is lifted, the system resumes visual tracking control, and limits the control quantity during the recovery phase to avoid system jitter caused by sudden changes in control commands.

When no new visual data is received or the visual result is invalid, the system keeps the pan-tilt control quantity of the previous cycle unchanged, thereby avoiding pan-tilt jitter or misoperation caused by the lack of visual information, and ensuring the continuity and stability of the system operation.

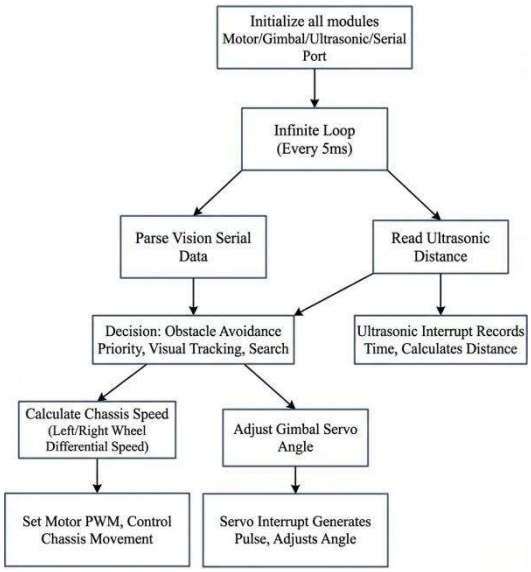

Figure 8. Main Program Flowchart

5. Experimental Results

5.1 Overall Structure of the Smart Car

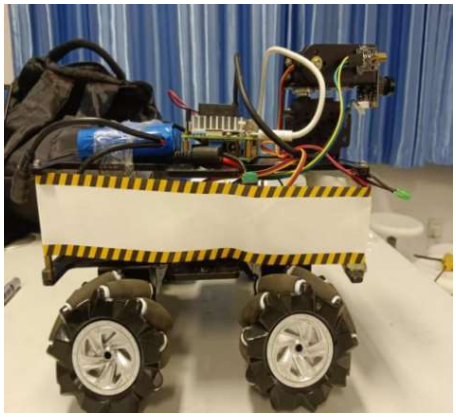

Figure 9. Overall Structure of the Smart Car

The smart car adopts a two-layer acrylic board to construct a layered three-tier installation structure. (Lower layer/Middle layer/Upper layer). The lower layer is the motion execution layer, equipped with Mecanum wheels and 520 geared motors to achieve omnidirectional movement; the middle layer is the control and power supply layer, integrating the main control board, TB6612 motor driver board and power module, and an HC-SR04 ultrasonic module is installed at the front of the car body for forward distance detection and obstacle avoidance; the upper layer is the perception and pointing

ISSN: 2414-1895 DOI: 10.6919/ICJE.202602_12(2).0015

layer, deploying Raspberry Pi 5B, and carrying a camera and laser emitter through a two-dimensional pan-tilt to realize target recognition and pointing control. The overall structure of the car is shown in Figure 9.

5.2 White Diffuse Reflection Area Recognition

To verify the recognition effect of the Raspberry Pi-side vision module on white diffuse reflection areas, this paper performs online inference on the collected images and outputs detection results. The model provides the confidence level and the center coordinates of the target box for the target area, based on which the pixel deviation(dx,dy) of the target center relative to the image center can be calculated, providing input for the subsequent control module. Figure 10 shows the typical recognition results displayed on the PC after processing by the Raspberry Pi, indicating that the system can correctly locate the white diffuse reflection area in this scenario.

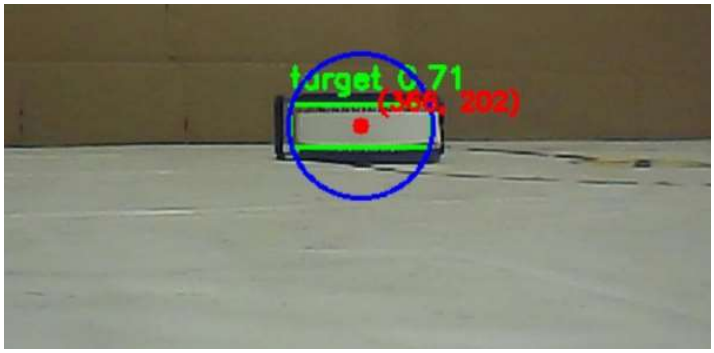

Figure 10. Identification of White Diffuse Reflection Area

5.3 Obstacle Avoidance Effect Demonstration

Obstacle avoidance tests were conducted in an indoor environment with a flat ground. Obstacles were set in front, and the car was made to move straight. The ultrasonic obstacle avoidance threshold was set to Dth = 25 cm, and the chassis speed was in low gear (approximately 0.25 m/s). The success criterion was defined as "no collision after triggering obstacle avoidance and successful escape from the obstacle area". A total of N = 20 tests were conducted, with M = 19 successes, resulting in an obstacle avoidance success rate of M/N = 95%. Figure 11 shows the typical results of the obstacle avoidance process.

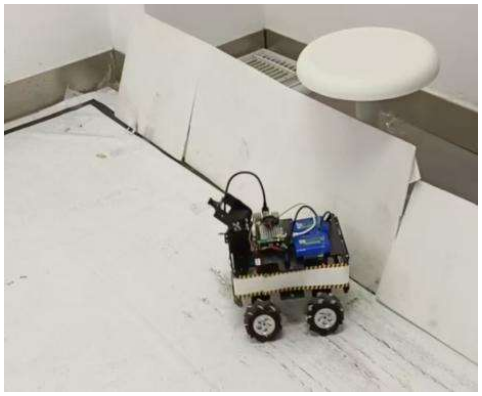

Figure 11. Obstacle Avoidance Effect Demonstration

5.4 Laser Countermeasure Effect Demonstration

Through a series of designs, the intelligent vehicle finally accomplished the laser combat task, and the specific effect diagram is shown in Figure 12.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202602_12(2).0015

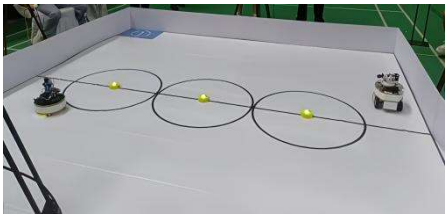

Figure 12. Laser Countermeasure Effect Demonstration

6. Conclusion

This paper focuses on the core tasks of laser countermeasure, obstacle avoidance, and continuous target tracking, and completes the design and implementation of an intelligent vehicle hardware system with STM32F407 as the control core and Raspberry Pi 5B as the visual unit. Through the collaborative work of the Mecanum wheel omnidirectional chassis, two-axis servo pan-tilt, and HC-SR04 ultrasonic module, combined with the "perception-control-execution" layered architecture and obstacle avoidance priority control logic, the system achieves stable tracking, efficient obstacle avoidance, and precise pointing of dynamic targets. In terms of hardware selection, reliability and economy are balanced; in software design, the real-time performance of visual recognition is improved through grayscale threshold filtering and YOLOv8 model, and modular programming reduces the system maintenance cost. Experimental results show that the intelligent vehicle can meet the requirements of laser countermeasure scenarios in terms of target tracking accuracy, obstacle avoidance response speed, and system stability.

In the future, the anti-interference ability of the visual algorithm can be further optimized, and the IMU inertial measurement unit can be introduced to improve the attitude control accuracy, so as to provide more complete technical support for laser countermeasure tasks in complex environments.

References

[1] Zhu Yueqi, Jie Fengxue, Sun Kexin, et al. Design and Implementation of an Intelligent Vehicle Laser Combat System Based on Fourier Series Fitting and YOLOv8[J]. Micro/Nano Electronics and Intelligent Manufacturing, 2025, 7(01):25-31. DOI:10.19816/j.cnki.10-1594/tn.2025.01.025.

[2] Chen Quanfu. Design of Control System for Intelligent Mobile Robot Platform[D]. Harbin Engineering University, 2006.

[3] Han Qiang, Yang Xiaohua, Yu Ruxing. Design of Intelligent Car Based on STM32 Microcontroller[J]. Automobile Applied Technology, 2026, 51(02): 21-26. DOI: 10.16638/j.cnki.1671-7988.2026.002.004.

[4] Zeng Zhilin, Ding Jie. Design of an Intelligent Express Delivery Cart System Based on STM32 and FreeRTOS[J]. Electronic Production, 2025, 33(15):7-12. DOI:10.16589/j.cnki.cn11-3571/tn.2025.15.014.

[5] Liu Hao. Design of Solenoid Valve Controller Based on STM32F407[J]. Colliery Mechanical & Electrical Technology, 2022, 43(02): 69-73. DOI: 10.16545/j.cnki.cmet.2022.02.017.

[6] Zhang Qiangzhi, Huang Jiexian, Zhang Ming, et al. Intelligent Self-balancing Vehicle Based on STM32 and PID Control[J]. Internet of Things Technology, 2025, 15(24):78-82. DOI:10.16667/j.issn.2095-1302.2025.24.016.

[7] Mo Ming, Yu Jiayue, Zhang Lin, et al. Design of a Multifunctional Smart Car Based on Raspberry Pi Microcontroller[J]. Applications of IC, 2022, 39(10):12-13.DOI:10.19339/j.issn.1674-2583.2022.10.005.

[8] Guo Junyu, Bai Yingkai, Fang Zixuan, et al. Design of Multi-sensor Intelligent Car Control System Based on STM32[J]. Auto Electric Parts, 2026, (01): 54-56. DOI: 10.13273/j.cnki.qcdq.2026.01.012.

[9] Ji Shijun, Di Weiguo. Design of Automatic Obstacle Avoidance Navigation Car Based on Raspberry Pi Main Control[J]. Electronic Technology, 2025, 54(02): 380-381.

[10] Luo Hao, Liang Fazhou, Lai Mingfa, et al. Design and Implementation of a Mechanical Arm Based on STM32[J]. Southern Agricultural Machinery, 2025, 56(17): 129-131.

[11] Su Lin. Design of Ultrasonic Range Finder Based on HC-SR04[J]. Science and Technology Information, 2012,(09):125+124.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202602_12(2).0015

[12] Hu Jiao, Huang Gengsheng, Yuan Weihua. Design of Mask Detection System Based on YOLOv5 and Research on Raspberry Pi Deployment[J]. Computer Programming Skills & Maintenance, 2025, (11): 150-153. DOI: 10.16184/j.cnki.comprg.2025.11.048

[13] Ji Xiangyi, Xiong Jie, Fang Jinyuan, et al. Design of a Garbage Cleaning Robot Based on Raspberry Pi[J]. Electronic Production, 2025, 33(24): 3-7. DOI: 10.16589/j.cnki.cn11-3571/tn.2025.24.010.

[14] Wang Yuxiang, Zhang Xihong. Design of Machine Vision Training Camera Based on Flask and Opencv[J]. Journal of Heilongjiang University of Technology (Comprehensive Edition), 2025, 25(09): 151-156. DOI: 10.16792/j.cnki.1672-6758.2025.09.029.

[15] Wang Qinhai, Sun Yuguo. Visual Object Detection and Tracking for Unmanned Surface Vessels Based on YOLOv8 and RK3588[J]. Journal of Chengdu Technological University, 2026, 29(01): 44-51. DOI: 10.13542/j.cnki.51-1747/tn.2026.01.008.

[16] Chen Qijian, Liang Taohua, Liu Hongtao. Comparison between STM32CubeMX Graphical Configuration Mode and MDK-ARM Code Development Mode[J]. Information Technology and Informatization, 2023, (12):59-62.

[17] Ren Wenfeng, Li Qing, Song Xiaohua. Design and Implementation of an Intelligent Trash Bin Based on YOLOv11[J]. Electronic Quality, 2025, (12): 8-12.

[18] Zhao Huandi, Tan Yufei, Qin Yunbai, et al. Design of Obstacle Avoidance and Line Tracking for Smart Car Based on STM32F407ZET6[J]. Electronic Production, 2023, 31(13):19-21+29.DOI:10.16589/j.cnki.cn11-3571/tn.2023.13.029.