Technical Report

For The ICRA 2019 RoboMaster AI Chenllage

Author
Yi-Kang Gu
Zhao-Yi Meng

Dongguan University of Technology
Electronics and IT

1 Hardware

1.1 Framework

We are going to use the standard robot as the development platform (hereinafter referred to as "Infantry") which developed by DJI TECHNOLOGY CO.

Benefits of using Infantry includes high stability, low development costs. Cons like low personality, bad mechanical design that would probably cuts off the LIDAR scanning range also exist.

The hardware parts mainly includes mechanical part and device part.

1.2 Mechanical part

The mechanical part mainly consist of two modules; chassis module and gimbal module.

1.2.1 Chassis module

By using Mecanum wheels, Infantry are enable active omnidirectional movement.

Four 3508 Brushless DC gear motor with governors provide power to push the robot and payload.

We will be using RoboMaster Development Board Type A(STM32F427IIH6) as main control unit.

1.2.2 Gimbal module

Two dimensions gimbal drive by GM6020 DC brushless motor.

loading system drive by M2006 P36 Brushless DC Gear Motor.

Firing system drive by DJI Snail 2305 Racing motor.

We will be using RoboMaster Development Board Type A(STM32F427IIH6) as main control unit.

1.3 Device part

1.3.1 Slave computer

Two RM development board type A will be using as slave computer on chassis module and gimbal module separately.

It features various ports for further development such as UART port and CAN port that could allow it communicate with master computer, PWM port that could drive servo and brushless motors, and it also supports UWB localization and SDK development.

Multiple protection mechanisms including a built-in ESD on PWM interfaces prevent reverse connection, over-voltage and over-current.

1.3.2 Host computer

So far we have two choices for master computer, we could either simply buying the Nvidia TX2, it is expensive but will save a lots of time or just DIY an IPC, which will be obviously cheaper than the previous choice but time-consuming.

For the first choice, Jetson TX2 is a tiny little board built around Nvidia Pascal-family GPU with 256 CUDA cores which means faster speed on matrix multiplication. The CPU complex consists of a Quad core A57 ARM processor connected to a dual core Denver processor, way slower than the I7, but it is ok to use.

For the DIY IPC choice, we would using the chassis with dimension of 18.5X4.5X19.75CM(W*H*D) to meet the Infantry mechanical space where chassis are going to be locate.

After all, we are going to remain both choice of the master computer, simply because a higher CPU performance is needed for ROS, so that it would be better choice if we developing on IPC platform and then transfer it to the TX2 after the developed.

1.3.3 Camera

Based on our previous experiences, it is highly import to have a camera that can reduce motion blur. What's more, the parameter adjustment is also one of the necessary functions.

And the convenience. As mentioned above we finally chosen a camera that supports USB3.0, parameters adjustment and high frame rate for motion blur reduction. The resolution of the camera is not that satisfactory, with only 640x480 resolution rate, but the competition does not requires the resolution rate that much. It is quite enough to get image features such as amour light bar on this resolution rate. In short, the camera we have chosen is quite enough for the competition.

1.3.4 LIDAR

LIDAR is an extremely important device for localization and navigation.

Since the LIDAR works like shine a small light at a surface and measure the time it takes to return to its source, Therefore the adaptive scanning frequency and range sample frequency would be the high priorities for choosing a LIDAR.

The LIDAR we are going to use is the G4 LIDAR which developed by YDLIDAR CO, with range sample frequency about 9000hz, scanning range around 16m and maximum 12Hz adaptive scanning frequency.

The reason why we are choosing this LIDAR is because it has the highest adaptive scanning frequency and range sample frequency among same type of product, and the price is also quite reasonable.

1.3.5 Ultrasonic sensor

Ultrasonic sensor will be using as a auxiliary positioning device of compensation and calibration for LiDAR positioning during the robot reloading process.

HY-SRF05 ultrasonic sensor can work from 2cm to 3m with 15 degrees measuring angle and ranging accuracy is reaching no more than 3mm. Since it is a relatively mature product on the market therefore we chose it as auxiliary positioning device.

2 Embedded System

2.1 Framework

2.2 Chassis Control

2.3 Gimbal Control

3 Software

3.1 Enviroment

Consider the development time we have and the comprehensive requires to the robots of the competition, an existing /open-source toolkits would be use during the development to avoid a bunch of repetitive works.

So far we have abandoned the choice of building a framework ourselves, instead, choosing the Kinect version of ROS as the development environment for the host computer to build an algorithm framework.

- OpenCV library (For visual processing, including armors detection, robot orientation recognition) - CUDA toolkit (For speed up the GPU processing speed, that include visual and neural network processing) - Localization and the Navigation modules (Other corresponding toolkit)

3.2 Framework

Software framework(includes Embedded system) on above.

IPC(etc Nividia TX2) stands as master computer and takes change of higher level calculation such as object detection, object relative distance calculation.

Perception including:

Localization, that contents robot localization, object detection and object recognition.

Decision, base on the behavior tree or other machine learning framework.

Planning, global path planning and local path planning.

3.3 Function Modules

Function modules are the modules that processing the sensory input and transform the lower-level information to higher-level information for the host to take further actions.

3.3.1 Detection module

Visual detection is one of the most important method for autonomous robots in perceiving the environment. In this project, visual detection will be using on enemy identification, armors detection and obstacles highlight and so on.

We would use different algorithms on different objects detection.

The visual model of the armor is composed of two LED strips, each of them has the same length, parallel to each other and vertical to the ground.

The armor has two different colors, both of the brightness of blue and red are extremely high. Therefore, we need to extract the corresponding color blocks inside the image first.

Then, we are going to filter out the non-conforming color blocks according to the shape of the light bar, parallel features, length features, and direction features. We are going to match each other bar at the same time.

Lastly, filtered out all the remaining noise, leaving many pairs of color blocks that conform to the features.

TODO::two pictures

We had already tested both algorithms based on RGB and HSV gamut extraction.

Generally, the image storage in OpenCV defaults to BGR three-channel storage, the R and B for different channels. The highest value for pure color and the remaining values are below 50 on average, which also proves the effectiveness of the RGB separation method.

By using well-design pixel pointer, we could achieve the fastest processor speed on OpenCV, so the RGB separation method should be considered more effective than HSV separation method.

In actual test, the color brightness and the purity are extremely high, which would cause the normal camera to be overexposed so that the imaging result of the center color of the light bar get white.

Therefore, we would use an industrial camera with adjustable parameters.

By lowering the brightness and exposure, increasing the contrast and slightly adjusting the saturation to optimize the algorithm on hardware level.

After the adjusting, the red color block recognition rate would greatly increase as well as the identification rate of armor.

3.3.2 Localization module

Localization is the process of determining where a robot is located with respect to its environment. It is one of the fundamental competencies required by an autonomous robot.

We are going to use AMCL algorithm, which known as Adaptive Monte Carlo Localization, this algorithm needs high precision odometer data for localization. Factors that leads to robot wheels slipping or idling could cause increase of odometer and by the time the cumulative error reaching a certain level, serious deviation of the positioning might occur, which leads to the robot's inability to estimate its own position over time.

So, we are considering by adding a global localization system to help with the data correction, for that, we would like to use the Robomasters UWB positioning module. The previous test results from the DJI summer camp of the program of using this UWB positioning module as global localization system is not very satisfactory; the robot frequent collisions with obstacles during the dynamic navigation. It might because the module needs precisely calibration, further tests will be needed.

3.3.3 Navigation module

Path planning is one of the essential functions of autonomous mobile robots, which given the ability of a robot to find the shortest path between two points.

This time, we are going to use A* algorithm; it is an informed search algorithm that widely used in pathfinding and graph traversal, it finding the path by maintaining a tree of paths originating at the start node and extending those paths one edge at the time until its termination criterion is satisfied. The algorithm determines which of its paths to extend based on the cost of the path and an estimate of the cost required to extend the path all the way to the goal; then it selects the path that minimises.

We are going to use the TEB local planner for local planning, TEB local planner able to optimize the robot trajectory online and produce alternative trajectories in distinctive topologies, moreover, it supports forward and backwards driving of robot, consider the mercurial environment of the battlefield(stage) the TEB would be the best choice for local planning.

Planning processes:

Assume that point A(green) wants to get to point B(red) and there is a wall(blue) separates the two points.

Firstly, algorithm add point A into an ‘open list’.

Secondly, it starts to look at all the reachable squares adjacent to the point inside the open list, ignoring squares with illegal terrain like walls. Add them to the open list too. For each of these squares, save point A as ‘parent square’ which be used to trace our path later.

Lastly, algorithm move the starting square (point A) from the open list to ‘close list’ after reachable squares were added into open list and start to determine which square to use when figuring out the path using the equation:

$$f(n) = g(n) + h(n)$$

Where G is the movement cost to move from the given location to the goal square on the grid, following the path generated to get there.

H is the estimated movement cost to move from the given square on the gird to the final destination.

Choice the square with the smallest f index and repeat step 1,2,3 until the final goal is inside an open list.

3.4 Behavior tree

Type of nodes in the behavior tree:

Selector node: Select the child-node from highest-priority to lowest-priority, once there are more than one child-node satisfied the condition, the child-node with the highest-priority will be chosen.

Condition node: Judging the condition, once it is not satisfied, return None, Otherwise points to the actions below the condition.

Action node: Specific function that controls robot action.

Abbreviation words in the behavior tree:

FA* Action means Frontal attack, both Master and Slave robots will start to frontal attack with the target.

SA* Action: Sneak attack, robot will attack the target from behind.

FF* Action: Focus fire, both Master and Slave robots will focus fire on one target.

GAB* Action: Go and activate buff, robot will go and activate the buff.

MBS* Action: Master robot fellow behind the Slave robot.

SBM* Action: Slave robot fellow behind the Master robot.

UTF* Condition: unable to fire, robot is not loaded.

RTF* Condition: ready to fire, robot is loaded.

Starting Condition:

Judgment:time left greater than 150 second.

Behavior: Slave robot start reload and Master robot will cover it.

Effect:Robot will rush to reload after game starts.

GG Condition:

Judgment:Only one robot surviving and the HP should below 600. Behavior:Avoid contact, try to reserve.

Effect:Once there seems on chance for us to winning the game, robot will try to avoid contact as much as possible.

UTF Condition:

Judgment:There is no bullet left inside the bullet tank.

Behavior:Reload

Effect:Robots will reload the bullet when the bullet tank is empty.

RTF Condition:

Buff Activated Condition:

Judgment:either one robot stayed in buff zone for more than 5 seconds.

Behavior:All robot will switch buff activated condition.

Effect:Robots will take more aggressive actions.

Buff Deactivated Condition:

Judgment:Neither robot stayed in buff zone for more than 5 seconds.

Behavior:All robot will remain buff deactivated condition.

Effect:Robots will take normal actions.

Under attack Condition:

Judgment:Any damage detected by referee system.

Behavior:Under attack Condition will last in the next eight second.

Effect:Robot would take action to avoid damage in this condition.

Attacking Condition:

Judgment:Robot is fire on the target

Behavior:Attacking condition will last in the next three seconds.

Effect:Robot would choose different kind of way to attack the enemy