Glasgow College, UESTC
Team Design Project and Skills

Team 30 Final Report

"A Certain UESTC Railgun"

Team Members:

Yukai Song 2357612S
Zihao Huang 2357611H
Zichao Liu 2357617L
Ze Sheng 2289050S
Dingte Yang 2357909Y
Zhiheng Lin 2289052L
Weihao Xiong 2357623X
Jingyi Ran 2357641R
Jiajun Huang 2357654H
Fangze Xu 2357602X
Instructor: Wasim Ahmad, Abdullah Al-Khalidi

Abstract

In this TDPS (Team Design Project and Skills) project, we need to construct the patio based on the size given by the professor, choose or make a robot, add sensors, robotic arm or cameras to our robot if necessary and program our robot controller to enable our robot to complete the task 1 to task 5 given by the instructors.

Environment building is mainly done by adding the nodes from the Webots library but we also do some modifications to decrease the physical complexity, overcome some physical problems and modify our environment to make it look better and more realistic. The box can add a lot physical difficulty of environment and thus we put the box into the image (image is produced through the photoshop) that is put in the texture image of floor and this method solves the line and box putting problem as well as physical complexity issue. Friction and shape of the bridge can cause a lot of problems when our robot is crossing the bridge. We design our bridge from 3DS MAX software (add an auxiliary ramp and arc bridge floor) and attributes a larger coulomb friction coefficient to the bridge to avoid sliding. 3DS MAX also helps us design the arch with a better looking. By disabling background light and material brightness, we can make the map much darker and create the night mode of the environment, which simulates the condition of bad weather in real TDPS.

Following the lines, color recognition, object detection and control, main logic to link them together and ensuring safety are main functions that the robot needs to acquire in order to finish the task 1 to task 5. In this project, we develop the Mid-value with PID control algorithm for task 1 line following and least square with PID control algorithm to tackle the right angle turns in the task 4 line. Color recognition algorithm experiences a progress from traditional threshold method to ROI (Region of interest) algorithm that is more stable and is more efficient compared with the traditional threshold method. Besides, ROI is a close-loop control while traditional method is open-loop control and close-loop control works much better in reality. Robotic arm control utilizes position set, middle state and mechanical claw propulsion method to increase the stability. Object detection is achieved through the camera API object detection function, which provides the object ID, relative distance and orientation of objects which helps the control of robots. Similarly, distance sensor will reflect the

relative distance and orientation of the forest to help the robot avoid colliding with the forest. PID control is utilized for the line tracing that requires high accuracy while the proportional control is used for object detection control where the computing speed is highlighted. The main logic structure of the robot is developed through the usage of switch case structure. In each task state, the robot is doing the task and detecting whether the robot meets the condition of the case as well. If the robot meets the condition of case, it will change into the next mode to complete the next task. Safety of robot is always considered in each task and it is mainly done by adjusting parameters and using PID control to help the robot always keep its velocity within its maximum speed.

Besides the technical problem, project management is another important part of this project. Firstly, the team leader Yukai Song worked out the project planning that includes Work Breakdown Structure, Workload Distribution, Project Network Diagram and Gantt Chart with the help of team. Next, the initial design was designed out by Fangze Xu and then the whole team cooperated with each other to finish tasks from patio building to task 5 according to the project plan. In this process, we adapted the teamwork skills of breaking a huge task into small parts and brainstorming to ensure our teamwork. Time management is another important part and we ensure that our time meet the project plan through communicating, weekly meetings and feedback from each member of the team.

The result of our work was shown in the final presentation and received admiration from the professor Abdullah. This should be thanked to the good project management and effort from each team member.

Table of Contents

Catalogue

List of Figures and Tables

Acknowledgement

First and foremost, we would like to express our deepest gratitude to DR. Wasim Ahmad and DR. Abdullah Al-Khalidi, who are with extraordinary patience as well as consistent encouragement and have provided us with valuable suggestions and guidance in every stage of our design. Secondly, we are grateful for each member's contributions to this project. We learn to cooperate and communicate and thanks to the clear division of work and teamwork spirit, every part is completed in order.

Introduction

Nowadays，artificial intelligence and automation system have been widely applied in our daily life, robots can do loads of tedious work for us human beings. For instance, we can see robot waiters serve guests according to specific route, robot dustman can help cleaner to clean the city and do some work of garbage sorting. We can see intelligence and automation system have been an indispensable part of our daily life. One important application of automation system is automatic transportation, just like many technology companies are researching on driverless technology, transporting safety can be improved by automation system, In reality, the number of traffic accident increases year by year and this increasing figure is largely resulted from unavoidable mistakes of drivers, like furious driving and fatigue driving. Automatic transportation can solve these problems to some extent.

In this project, we focus on automatic transportation, using Webots to design a multifunctional robot, who can run in a specific route, recognize specific colors and avoid obstacles etc. The designed robot moves along the required routes and finish different tasks from 1 to 5. Task 1 focuses on line tracing, task 2 mainly shows the function of color recognition and robotic arm control, task 3 concentrates on bridge crossing, task 4 is about arch detection and task 5 is consisted of color box detection as well as path following. 5 tasks can test the powerful functions of the robot.

As far as project plan is concerned, 4 key algorithms are deeply researched：PID for line tracing, color recognition, robot arm control and path planning based on object recognition and distance sensor. Every algorithm is divided into 2 parts: various algorithms (code writing according to theory) and packaging code of software. In this

project, team members study and test algorithms and packaging code of software respectively and judge which one is the best and can save researching & developing time. Optimization work focuses on the improvement of 4 key algorithms to achieve a better result.

Besides the modification of algorithms, we also highlight the decoration of the environment to make our simulation more realistic. Firstly, based on the basic environment, we add many other staff in the environment, such as cars, advertising boards and houses. Secondly, we develop two modes of environment: day mode and night mode and the night mode help our robot develop the ability to finish all tasks even when the lighting condition is not satisfactory.

1 Overall System Design Approach – Proposed Design

1.1 Technical Design

1.1.1 Hardware Part

In this section, we will introduce our general method to design the robot to finish 5 tasks. Firstly, we will introduce the overall system design, then we will explain its basic working logic, finally, we will show the team members distribution on each sub tasks

1.1.1.1 Environment Design

1. Map frame design based on the requirements of instructors
2. Fish food in task 2
3. Other buildings like houses, bridge and arch
4. Construction of the night mode

For further details about how we construct the environment, you can look through the details in the 2.1 Patio Building.

1.1.1.2 Robot System Design

The design of robot can be illustrated as a combination of various structures and it is demonstrated below.

Device:

1. Camera (x3)
2. Distance sensor (x1)
3. Youbot from Kuka (x1)
    1. wheels (x4)
    2. Robotic arm (x1)
4. Controller

1.1.1.3 Robot System Structure

In order to explain how we integrate those devices to our robot, we draw the figure which can tell you the intrinsic logic about integration of our robot.

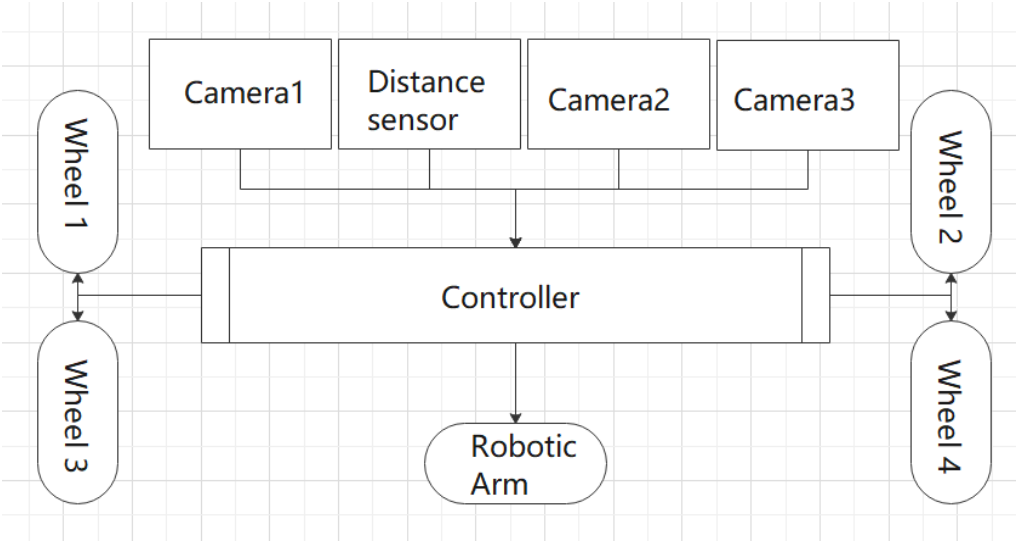


Figure. 1: Robot System Structure

1.1.1.4 Subsystem Design

From the robot system structure shown above, we can see that the four sensors (Camera1,2,3 and distance sensor) can provide information to the controller to make decision. And we need to design the sub system according to each sensor.

Camera1: Observe the line to make robot keep close to the traced line and

recognize the colour in front (Task5), which are **PID trace and colour recognition**.

Camera2: Recognize the colour on the right (**Task2**), which is **colour recognition**.

Camera3: Recognize the object in front (bridge and arch), which is **Object recognition**.

Distance sensor: Detect the distance between the obstacle in front of the robot.

1.1.2 Code Design

The code of the controller is the core part of the project, which drives the robot to act based on requirements of different tasks. In the project, we divided the involved functions into four subsystems: line tracing, color recognition, object recognition and distance detection (in this part robotic arm control is integrated in color recognition).

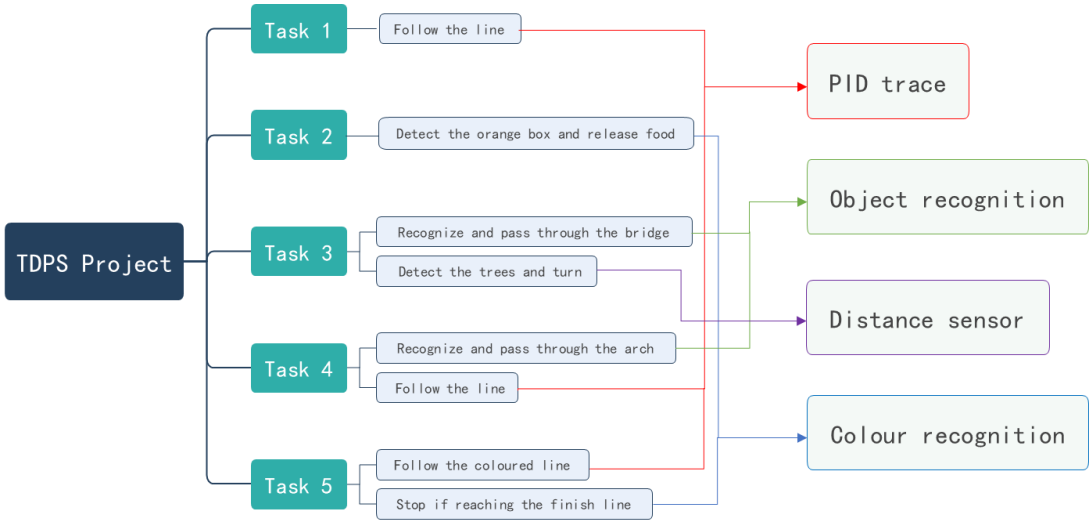


Figure. 2: Code design diagram

1.1.2.1 Algorithms:

1. PID tracing: Applied in task 1&4&5, it controls the car to move along the line on the floor. The ground information detected by Camera1 is analyzed and the middle point of line can be calculated. Then the tires can adjust speeds to control the car moving the direction of the middle point. Feedback system should be introduced to ensure the car can keep in correct route when turning or encountering interference.

2. Object recognition: Applied in task 3&4, it controls the car to get close to the bridge or the arch and pass through it. With recognition color, the bridge and the arch

can be recognized by the Camera3 and data about relative orientation and position can be obtained and processed to adjust route. the Feedback system should be introduced to ensure the car can move as required without line.

3. Distance sensor: Applied in task 3, it ensures the car can detect the forest and avoid trees. The distance information can be obtained when the sensor is activated. The car can determine whether to do next according to the value of it.

4. Color recognition: Applied in task 2&5, it helps the car to recognize the fish-food box and the colored box. Camera 2 can detect each pixel and judge its color, if the pixel number of interested color exceeds a value, that color is successfully detected. In task 2, the robotic control algorithm will come into use once the yellow color of the box is recognized.

1.2 Project Planning

1.2.1 Statement of Work

Introduction:

The project is an online project which requires running a simulation on software. It needs members to accomplish 5 different tasks on the simulation software (Webots). You can find further information about task requirements and membership in the following report.

Purpose:

The project is both small scale and non-profitable. It aims to practice students' skills in co-working by building the environment. Secondly, the project also train us with project management skills meanwhile check abilities in engineering and programming.

Scope of work:

The project can mainly be divided into three parts:

Preparation: Project requirement reading and understanding; Work distribution; Software and hardware knowledge learning. This part will take 2 weeks.

Project programming: This is the biggest part of the project, which includes patio building robot choosing and testing, sensor installing and programming and other simulation requirements satisfying.

Notebook and report: Members need to draw a conclusion after the simulation program is tested. The report should include abstract, plan, SOW, Work breakdown structure, project network diagram technical details and further work.

Schedule and Milestone:

The project will be submitted on week16, and we take the time to finish each project task as a milestone. You can see the Gantt chart below to get more details. We reassign tasks and learning Webots on week 6, and summary of learning webots choosing the proper programming language on week 7. Then do the environment building during week7 and week9.

Deliverables:

Considered the fact that the project does not involve real hardware building, the output of the project consists of only two parts: the simulation program and the project report.

1.2.2 Work Breakdown Structure

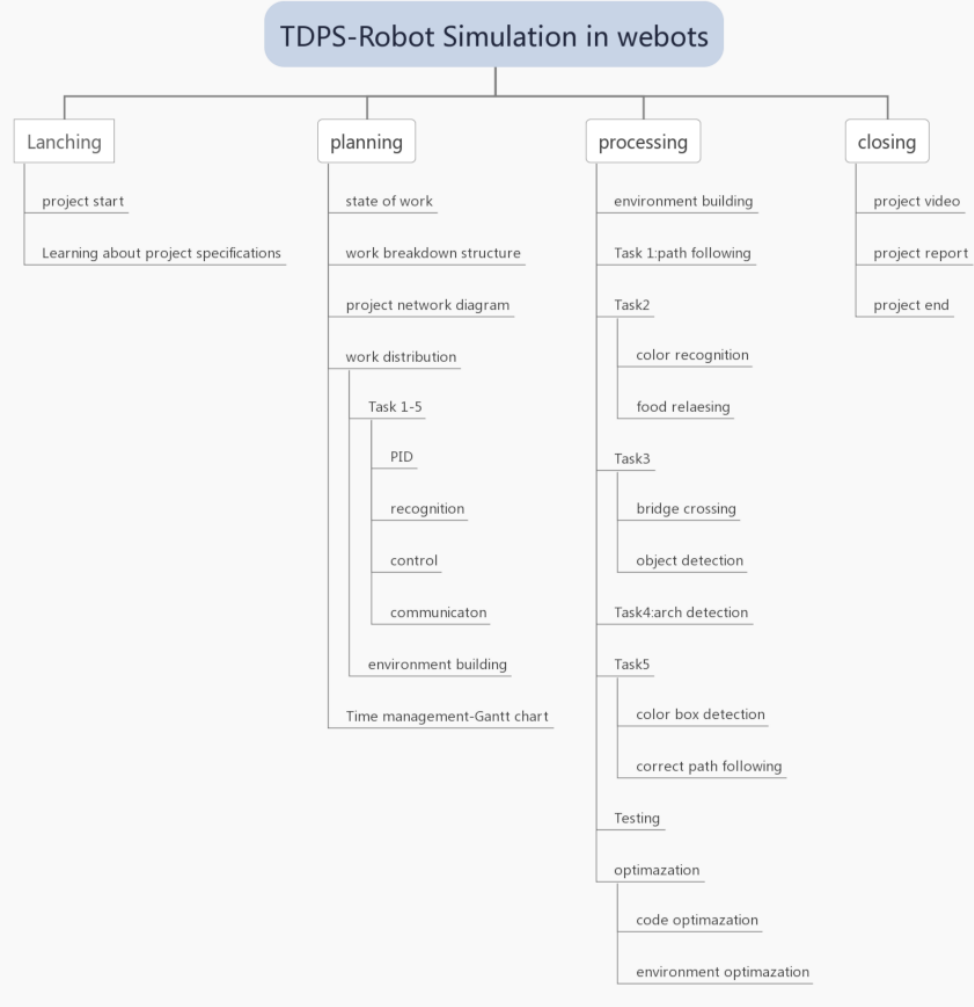


Figure. 3: Work Breakdown Structure

At the beginning of the project, we collected the specifications about tdps on the Internet. When it was not announced how to make this project, we divided into two groups according to the control and identification lecturers to collect relevant information. After determining to use webot for production, we decided to use C language as the main assembly language for this project after group discussion. And divided into five groups according to tasks 1 to 5, each of them participated in these five tasks. As the project progressed, we divided the tasks more carefully to ensure that the experiment could proceed well. After completing the preliminary environment construction, we have further planned the five tasks. After the completion of the five tasks, we successfully passed the preliminary test and we completed the robot design. Then we performed code optimization and environment optimization to ensure that our

robots completed the task more smoothly and the environment was more beautiful. After Xiong Weihao produced a cool demo video, we completed the project report together, and this tdps project ended successfully.

1.2.3 Workload Distribution

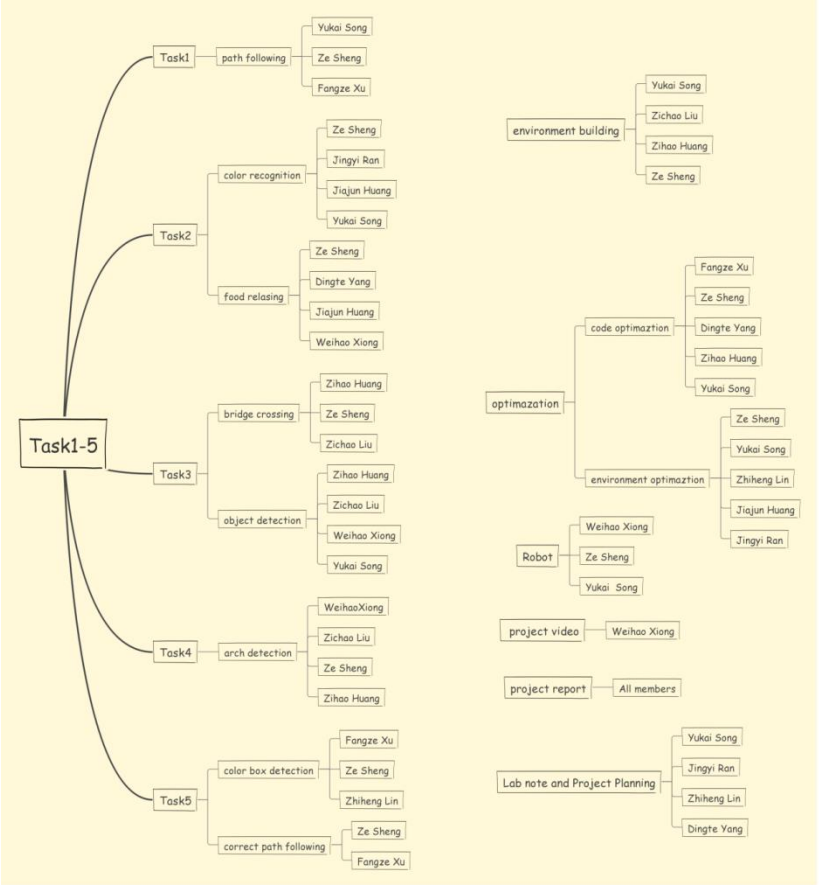


Figure. 4: Workload Distribution

This picture is our Workload Distribution, and In order to ensure that the picture is clean and tidy, it may not have marked all the projects that some people participated in. Take the environment optimization as an example. The demo video shows that our environment has not only night mode, but also many new items on the map. Many students have put forward their own suggestions for this environment optimization, such as poster design, adding street lamps and so on. So, the best shining point of our group is that each of us is responsible for the production of multiple tasks. We have carried out good teamwork and gained new knowledge and new friendship.

1.2.4 Project Network Diagram

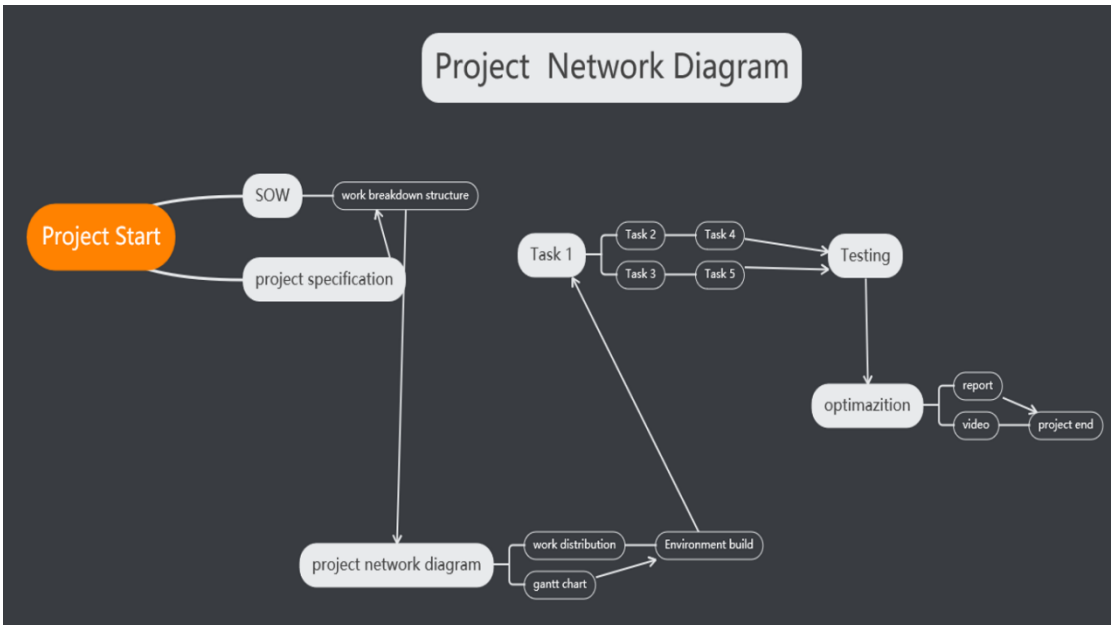


Figure. 5: Project network diagram

As shown in the chart, the project is divided into 4 part, when the project started, the first step is to focus on the requirement specifications of the project, be clear about the goal, time, what we will do,according to these questions, SOW and work breakdown diagram was worked out. Then we turned to the planning part.The work was distributed due to everyone's strengths. After this part, the team was looking on the task1-5. The first thing is basic environment building, which took a week. Then we started to do task1-5, after analysed these tasks , our team found that there are some similarities. For example, line tracing is both used in task1,task4 and 5, so we did task from1 to 5 modularly and parallelly. After finishing the basic requirement of tasks, testing and optimizing work started.Some creative factors was added into the project environment. Other team members turned to code optimization for better efficiency and stability. Last but not the least, we made videos, and wrote notebook as well as report.

1.2.5 Time Management: Gantt Chart

	Task	Start	Duration
1	Task allocation&pre-thinking discussion	2/26/2020 02:20 PM	18d2h40m
2	Reassign tasks&learning webots	3/15/2020 11:40 PM	8d2h20m
3	Environment building	3/24/2020 02:00 AM	10d21h45m
4	Task1: robot's path following	4/1/2020 04:20 AM	14d18h40m
5	Task2: robot's color recognition & food releasing	4/9/2020 12:00 AM	17d0h35m
6	Task3:robot's bridge crossing & object detection	4/20/2020 11:55 PM	16d0h5m
7	Task4: robot's arch detection & line following	5/3/2020 11:20 PM	19d0h45m
8	Task5:robot's color box detection & correct path following	5/9/2020 10:40 PM	14d20h20m
9	Testing	5/24/2020 11:50 PM	2d1h5m
10	Code optimazation	5/27/2020 03:40 AM	6d10h55m
11	Environment optimazation	5/27/2020 08:40 AM	6d15h5m
12	summary & report & video	6/3/2020 12:55 AM	11d21h30m

Figure. 6: Gantt Chart

Gantt chart shows project's time management, Green color represents planning and preparation that took lots of time since our team need to learn about new knowledge, spending 26 days in preparation and 10 days in environment building. As for task1 to task5, we spent about 2 months overall to finish it. You can see more time was used on task 1 to 3 rather than task 4 and 5. That is because there are some same techniques used in these tasks. As for optimization, which is shown in yellow, code and environment optimization were on processing at the same time,and it took 6 days. The last thing is video, report, notebook, and PPT , this part spent about 15 days and then we finish the whole project.

2 Subsystem Design and Solution(s) (Individual)

2.1 Patio Building

2.1.1 Lines and Advertising Board (Yukai Song)

Technical Content

1. Organize the design and construction of initial environment construction in

April by assigning tasks building the bridge, constructing the arch, adding the walls to different members in the group.

2. During the brainstorm of how to construct the line of the environment, I give out the way to put an image “textures/Yellow.jpg” in the “Floor:\ appearance Appearance\texture ImageTexture \url” and this successfully solves the problem of thickness of the line perfectly and reduces the physical difficulty. If we adopt the way of putting a lot of boxes to represent the line, it will have some thickness and adds physical complexity to the environment, which might slow the computing speed down. This method of adding image to the object in Webots becomes a fundamental part of our later environment modification.

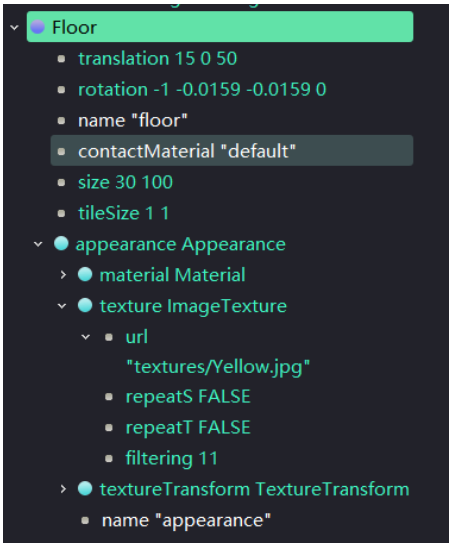


Figure. 7: Path of the floor texture

3. Attend the modificaton of environment and design the advertising board with Zhiheng Lin, Ze Sheng and Fangze Xu. The main technical part of this advertising board is in its front texture and we adopt the same method as we did in putting the line. We design our team logo together and add this logo to the “AdvertisingBoard:\frontTexture” and the detail of how we put the photo of team logo is show in the figure below.

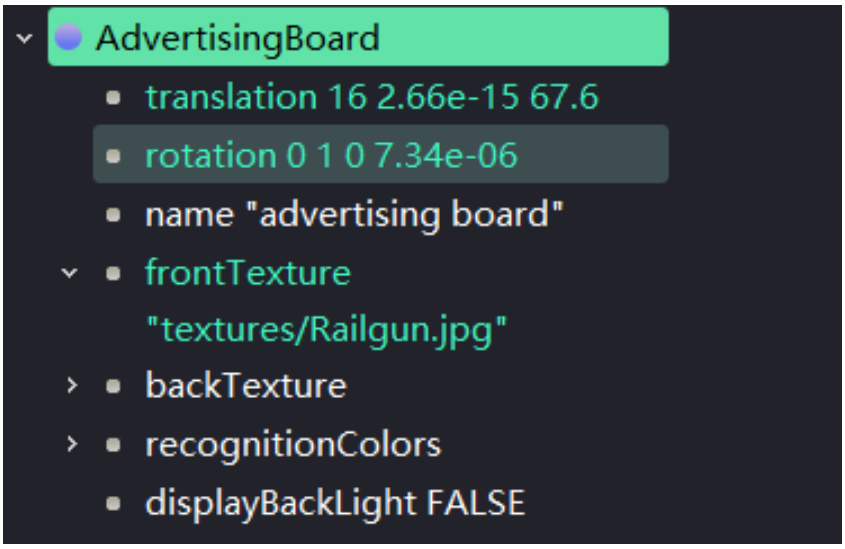


Figure. 8: Path of the ad board texture

Analysis and Discussion

1. After working with other team members for nearly a week, we work out the result of initial environment. I explain about the size of each building in this map and manage the time to finish each task and the result is shown in the photo below.

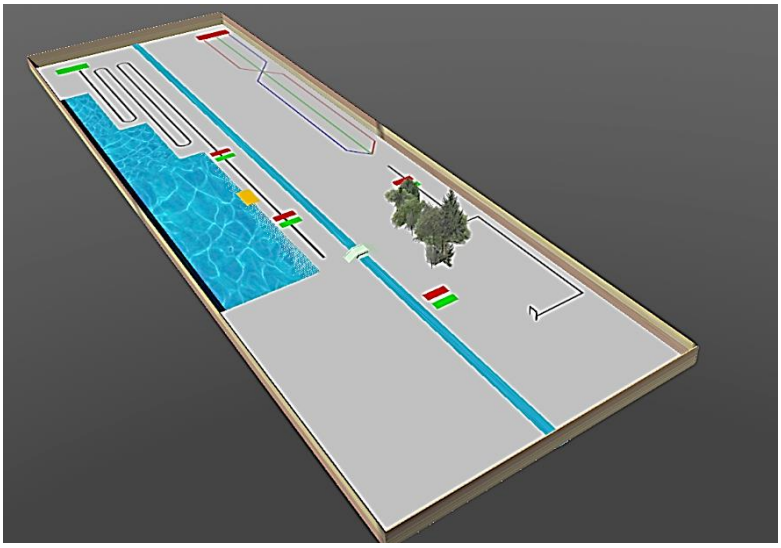


Figure. 9: Basic map of our project

2. I mainly discussed about how to put the lines into the environment during the environment construction. Some members of our team thought the lines could be substituted by the boxes. However, the thickness is a huge problem in this idea as in the real world, there is no thickness of lines while the boxes in the Webot has its thick. Besides, boxes can add the physical complexity of the environment and leads the computing speed to slow down. Through searching the tutorial of Webots, I found out

that adopting an image in the Floor would be a perfect way to solve this problem. However, the size of the photo and shape of the line became another issue. Through discussion with Fangze Xu, we decided to use Photoshop to design and adjust our line image. The result of one of our line images is shown above and there are two more images for the color red and purple in the task 5, where we change the color of the box manually.

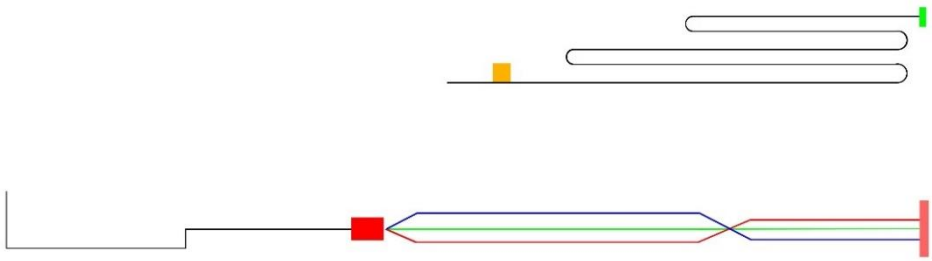


Figure. 10: Basic line map

3. We firstly chose the advertising board from the library of Webots and then discussed how to design our team logo. Some suggested that we could just put our team name in the advertising board but I did think we could do better than it. Thus, I found a wonderful photo, with the help of Fangze Xu's photoshop skill to modify the photo, we finally created a great team logo with our team name on it. Finally, I adjusted it and put the photo in "AdvertisingBoard:\frontTexture\"textures/Railgun.jpg"". The result of advertising board is shown in the following pictures below.

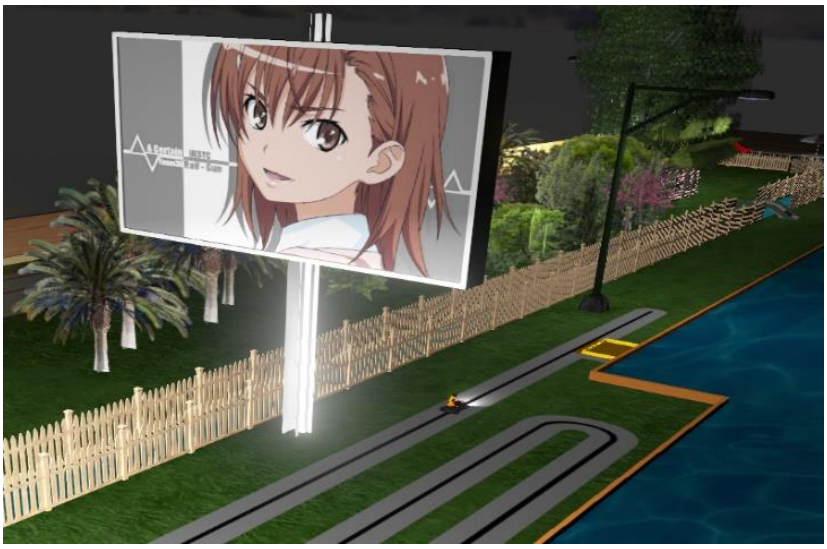


Figure. 11: The advertising board

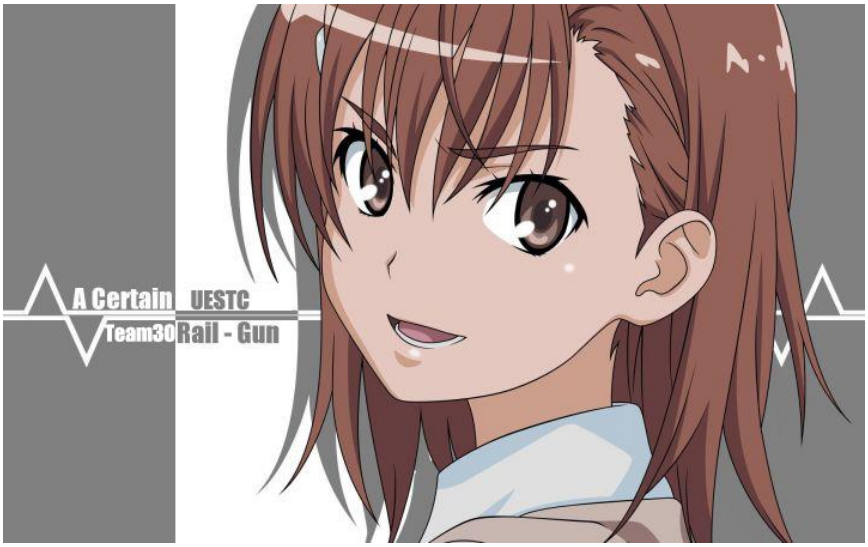


Figure. 12: The LOGO of our team

2.1.2 Frame Design (Zihao huang)

Technical Content

1. I constructed the frame of the patio before starting to build the patio, such as setting the appropriate coordinate system, ground size and background light. Then I designed the general layout in the southern part of the patio and built it as planned.

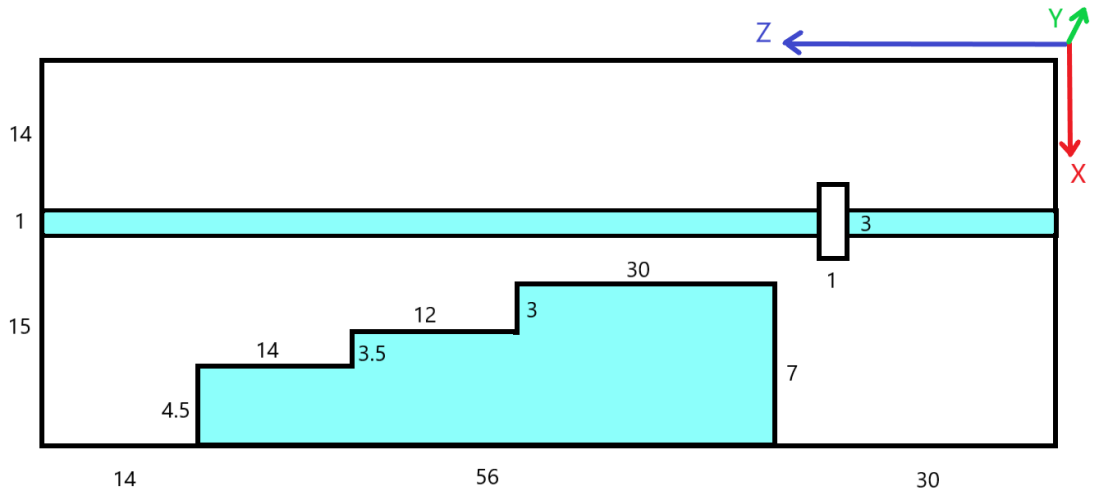


Figure. 13: Layout of the frame and south part of the patio

2. Built the south part of the environment and the bridge model in the earliest version.

2.1.3 Abstract of Environment Building (Ze Sheng)

Environment is a very important part of the whole project. A good environment can not only reflect the superiority of its own algorithm, but also bring user visualized and nice experience. As a professional team, our responsibility is to create the environment as beautiful and real as possible. In addition, because we will encounter a variety of weather conditions in reality, we need to take this condition into account, so we designed a night mode to simulate the night situation. All in all, the environment is an essential part of the project.

2.1.4 The Construction of Fundamental Environment (Ze Sheng)

In the construction of fundamental environment, I am in charge for the construction of walls, the design of dynamic water and the line optimization algorithm.

2.1.4.1 The Construction of Walls (Ze Sheng)

I used four boxes with wood textures as walls, enable collision volume but disable physical properties (otherwise it will fall out of this world).

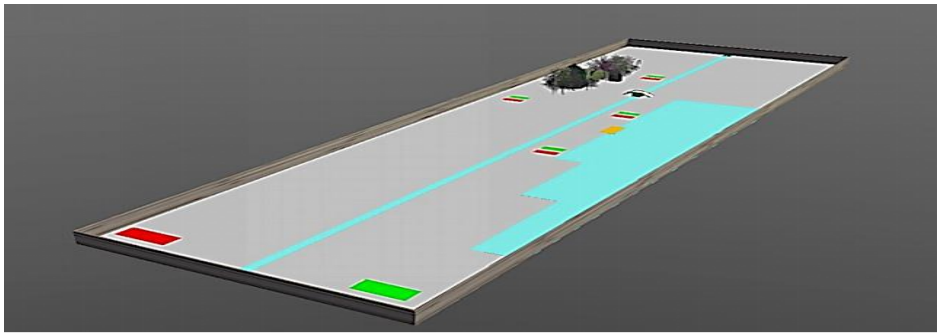


Figure. 14: Basic environment

2.1.4.2 The Design of Dynamic Water (Ze Sheng)

In order to make the environment real, it is very amateurish to use a solid box instead of water simply. In this case we used **Fluid** node to create the liquid areas.

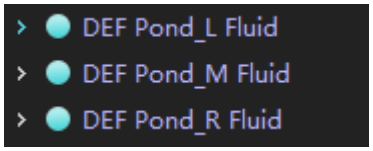


Figure. 15: Three fluid nodes

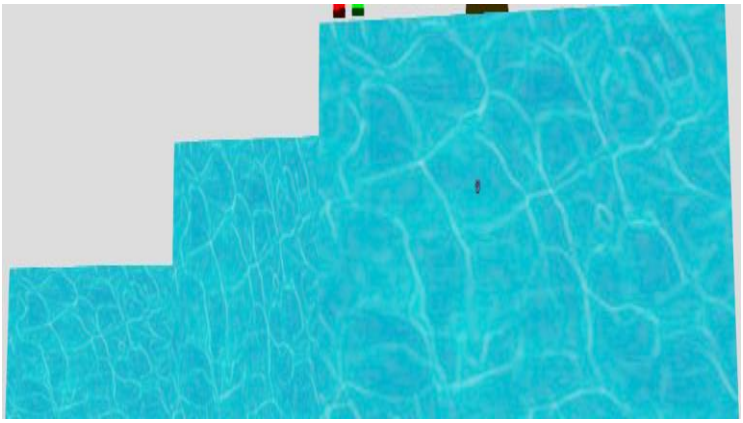


Figure. 16: Three parts of the pond

After setting the shape and position of water, we also need to input the parameters of water. Here we use standard atmospheric pressure (1.01kpa), the density and dynamic viscosity of water at 20℃. As shown in the picture below:

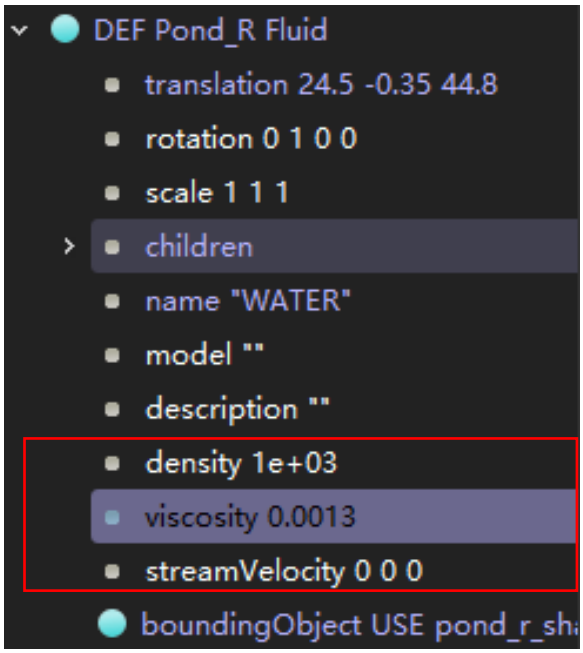


Figure. 17: Dynamic parameters of water

The density of water is 1000kg/m^3, and the dynamic viscosity is 0.0013Pa·s. So far, we have set the parameters of water (don't forget to enable the physical properties of fluid node). Next is the test, we need to show the buoyancy of the water to our client, so we need to make some objects that can float on the water. In this project, rubber ducks are the best objects, their density is about 100-400 kg/m^3. According to the buoyancy formula:

$$\rho vg > mg$$

The buoyancy of the rubber ducks is greater than its gravity, so the duck can float, and it turns out that this scheme is feasible. As shown in the figure below, these ducks floating on the water prove that the dynamic water we designed is available and successful.

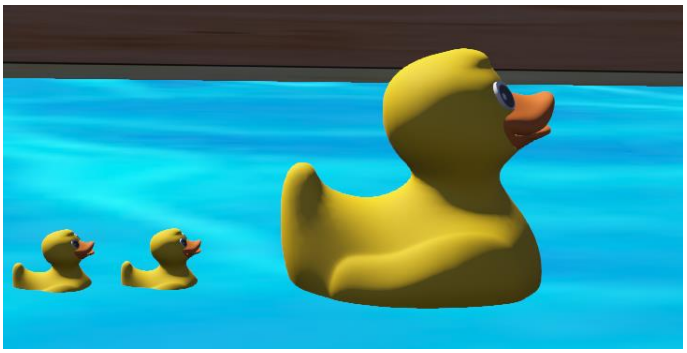


Figure. 18: The swimming ducks

So far, the design of water is over. Actually, this part is not very tough, but the idea

is very innovative. As a professional team, we need a more realistic environment, so dynamic water is essential.

2.1.4.3 Line Width Optimization Algorithm Based on Canny and Sobel Operators (Ze Sheng)

Because the design of the algorithm requires the robot to have a correct and accurate recognition of lines, so optimization is a necessary step. The main problem now is that our youbot runs across the line sometimes (caused by the excess of the line or the unevenness at the curve), so the first optimizing target is the lines in the map. In short, it is to make the lines more appropriate with the algorithm design by edge extraction and edge removal.

Canny[1][2], Sobel[3] and Prewitt[4], three popular image edge extraction algorithms at present, are used to directly optimize our map instead of writing into the line patrol algorithm. In this way, a large number of image convolution operations in the robot's image processing process can be avoided, which slows down the simulation speed.

Canny edge detection algorithm is a very powerful algorithm to extract the image edge. In filter the gray value of the image through gradient and double threshold of the image, We can obtained the edge of image. Compared with Sobel and Roberts algorithm, it has the advantages of weak edge extraction, strong adaptability and wide application.

Canny algorithm was originally proposed by John canny. To put it simply, Canny algorithm uses various operators (such as Sobel) to find out the region with the strongest gray value of the image, and retains the maximum value of the local gradient, that is, to achieve the edge extraction with non-maximum suppression. In addition, Canny algorithm uses two thresholds to predict the potential edge, which makes it more accurate and adaptive than other algorithms in theory. The following is the detailed analysis of Canny method in this paper.

The map in this part doesn’t need to be filtered, so it will not be described more in detail. First, calculate the gradient and direction. The following formula gives the calculation method of these two parameters.

$$D_x = \frac{\partial f}{\partial x}, D_y = \frac{\partial f}{\partial y}$$
$$|\nabla f(x, y)| = \sqrt{D_x{}^2 + D_y{}^2}$$
$$\theta = \arctan(\frac{D_y}{D_x})$$

Where f (x, y) is the input image function, D_x, D_y are the first derivative of a single pixel point in the direction x and direction y, which can be obtained by the way of Sobel operator convolution image. Then we can calculate the gradient direction and amplitude of this pixel point through D_x, D_y, which are represented by |f (XY) |, and θ respectively. Here, we give an example to brief description how to obtain D_x and D_y by Sobel operator. First, suppose a picture f (x, y) with a size of 3*3, and the gray value is replaced by a1 to a9. As shown in formula below.

$$f(x, y) = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix}$$

Formula above input Sobel operator S_x and S_y in direction x and direction y namely.

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$
$$S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

By convoluting two operators with the image, we can get following formula, that is, the calculation process of D_x and D_y.

$$D_x = \begin{bmatrix} -(2a_2 + a_5) & 2a_1 - 2a_3 + a_4 - a_6 & 2a_2 + a_5 \\ -(a_2 + 2a_5 + a_8) & a_1 - a_3 + 2a_4 - 2a_6 + a_7 - a_9 & a_2 + 2a_5 + a_8 \\ -(a_5 + 2a_8) & a_4 + 2a_7 - a_6 - 2a_9 & a_5 + 2a_8 \end{bmatrix}$$

$$D_y = \begin{bmatrix} 2a_4 + a_5 & a_4 + 2a_5 + a_6 & a_5 + a_6 \\ -2a_1 - a_2 + 2a_7 + a_8 & -a_1 - 2a_2 - a_3 + a_7 + 2a_8 + a_9 & -a_2 - 2a_3 + a_8 + 2a_9 \\ -(2a_4 + a_5) & -(a_4 + 2a_5 + a_6) & -(a_5 + a_6) \end{bmatrix}$$

Since Canny has non-maximum inhibition and double threshold algorithms, the accuracy is theoretically better than Sobel. To intuitively reflect the rigor and accuracy

of Canny system, we designed a small experiment to directly compare the edge extraction of a simple grayscale image by two algorithms. *Figure. 19* is an example grayscale image used to show the non-maximum inhibition performance of Canny. In the following experiment, Sobel and Canny will extract the edge of this image respectively

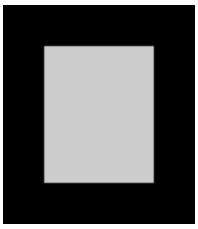


Figure. 19: White box with black edge (greyscale)

After the above calculation, we can get the grayscale amplitude and direction (Angle) of the image. At this time, the processing results are displayed and the edges extracted by Sobel algorithm are shown in *Figure. 20*.

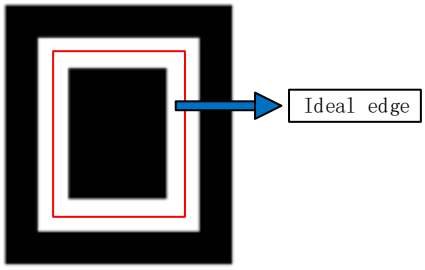


Figure. 20: Edge detection and extraction

The white part is the extracted edge, and the inside of the red box is the ideal edge of the image. It can be clearly seen that the edge extracted by Sobel algorithm is too thick, which will cause some errors in the edge extraction of irregular graphics. The Canny algorithm solves this problem more perfectly by filtering out a part of the edge through the non-maximum inhibition in the local area. To put it simply, the Canny algorithm only preserves the maximum value of the local region in the same gradient direction, meaning that the other values will become 0 (becoming the background). *Figure. 21* is the result of the same image processing by the Canny algorithm.

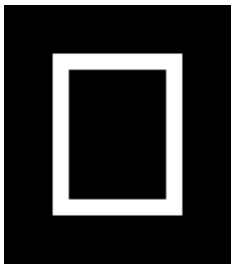


Figure. 21: Only the edge of white area

It can be clearly seen that the originally thick part is eliminated, proving that the non-maximum suppression algorithm does have a very positive impact on the precision of image edge extraction. As mentioned in, Canny double threshold algorithm can screen some 'false' edges by screening strong and weak edges, which to some extent improves the problem that Sobel algorithm is too sensitive to noise. However, because the images obtained in this paper (the maps) are less affected by noise and have steep edges, no further discussion will be made. The following is the result of processing the map for this project

Figure. 22: Canny edge extraction

Figure. 23: Sobel edge extraction

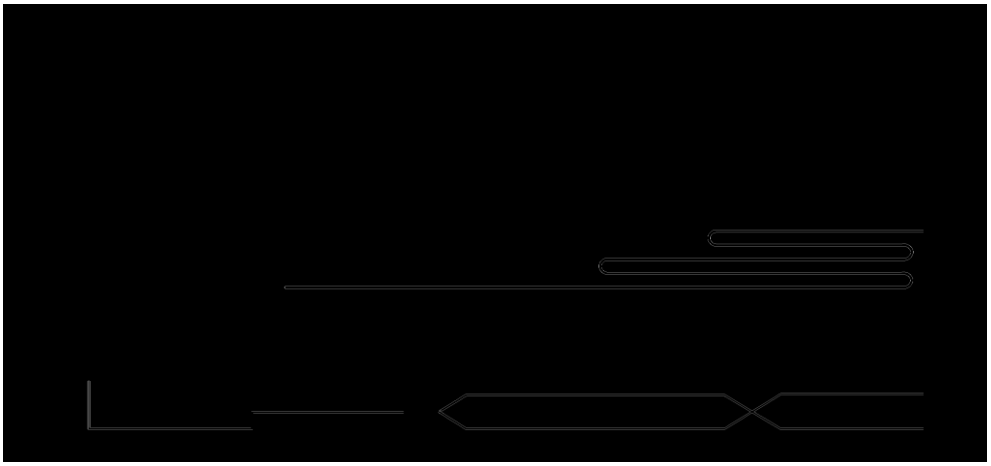


Figure. 24: Prewitt edge extraction

Note that the Robert129[5] algorithm is not discussed because its performance are inferior to Sobel's. It can be seen that the degree of Canny extraction to the edge is the best, most of the path edges can be extracted correctly. But that doesn't mean we have to just use the Canny algorithm. Because the curves in the original picture are jagged and uneven, the optimization for the curves is greater than for the straights. The following picture is original and the result of a canny extraction and optimization:

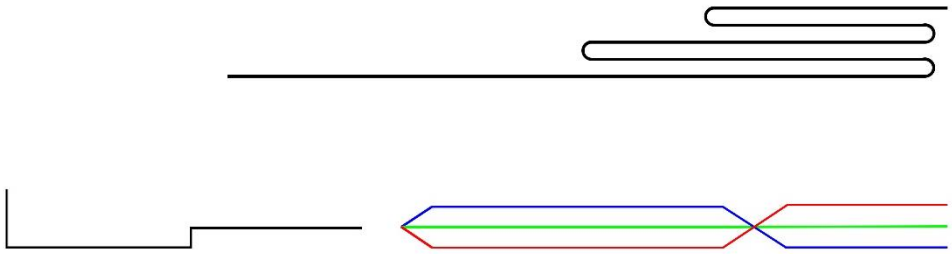


Figure. 25: Original map

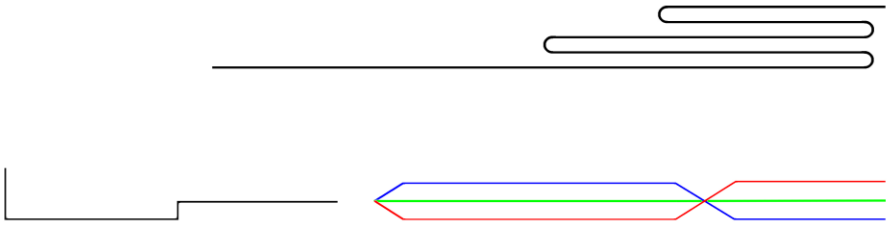


Figure. 26: Result of Canny opitimization

A significant reduction in the width of the runway can be observed. However, there are still some uneven edges, and we don't want the straight to be too fine, so we use Sobel method to optimize the curves (Sobel edge extraction is not sensitive enough, so the straight will not be affected, but the corners will be).After one Canny and two Sobel optimizations, the new runway we got completely meets the requirements of the algorithm and is more beautiful!

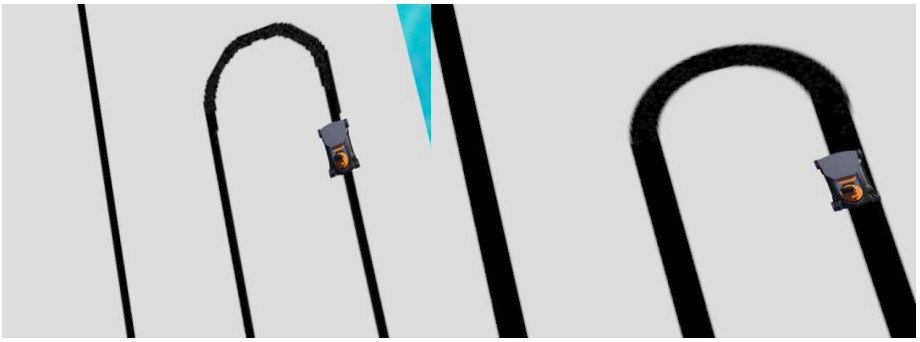


Figure. 27: Comparison between optimized (left) and original (right) path

2.1.5 Environmental Optimization (Ze Sheng, Zichao Liu)

2.1.5.1 Environmental Planning (Ze Sheng)

Just like urban planning and environmental planning in the real world, we also need a reasonable blueprint to arrange the space on our map. As the leader of the environment beautification group, the following is my design drawing of environment beautification:

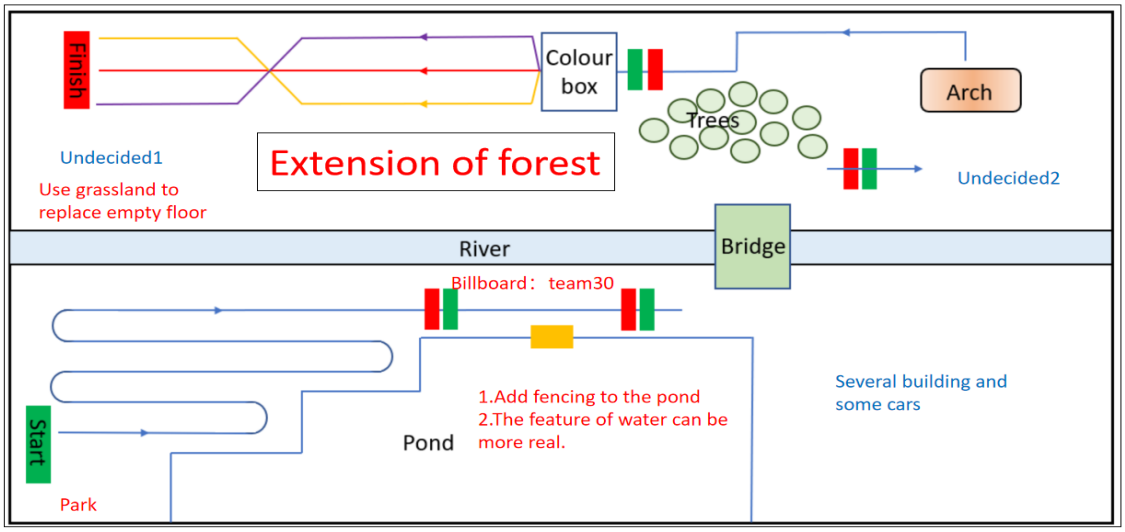


Figure. 28: Planning diagram of the new environment

Based on this map, we wrote a short dialogue of tour guide and tourists to introduce this map. Let us look at the final result of our environment:

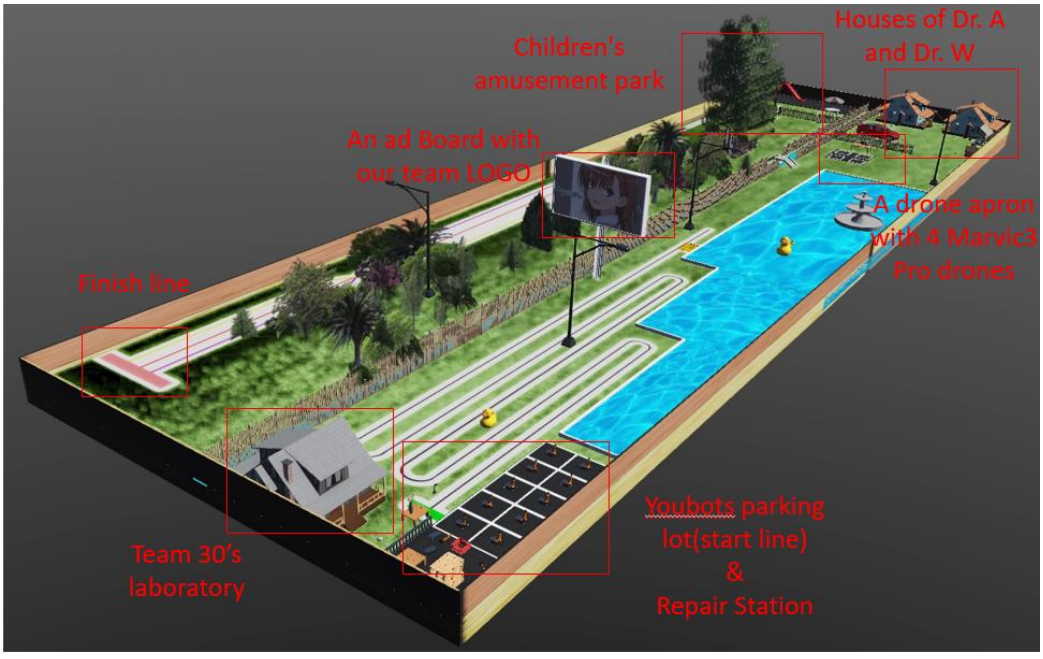


Figure. 29: Our new Environment

2.1.5.2 The Establishment of Facilities & Performance Analysis (Ze Sheng)

First of all, I used the grass texture of Photoshop to process the original floor. The following picture shows the specific material I used:

Figure. 30: The grass texture

The picture below shows the comparison between the original floor and the new floor. It is obvious that the new floor is much more beautiful:

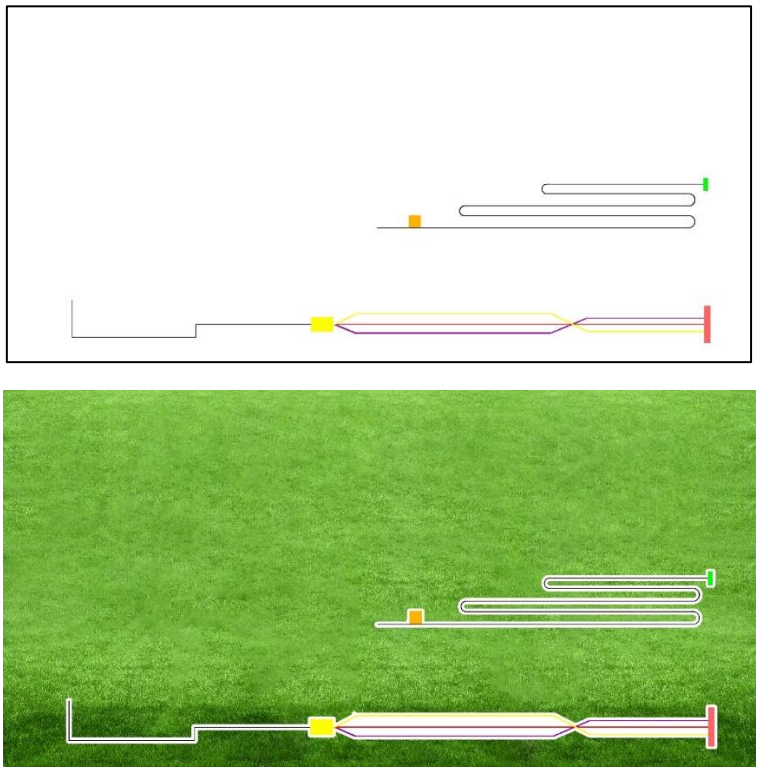


Figure. 31: Original and new floor textures

About all the ground facilities, this part of the work is quite heavy, I personally built about 95% of the environment. Here is a list of all the parts I built:

1. Youbots parking lot and repairing station.
2. Team 30's laboratory.

3. Children's amusement park.
4. The new trees and the extension part.
5. Tesla car and barbecue grill.
6. Advertising board.
7. All the streetlights.
8. Drones apron and four Marvic2 Pros

It is worth noting that in order to avoid the computer will be overloaded (running less efficiently) caused by too much calculation. For all robots, we should set its Controller to 'none' or 'void' and set its synchronization to 'False'. This ensures that the computer will not be burdened with unnecessary actions during operation. If you want to use these robots, you just give them controllers. The following figure shows an example of the parameters of the robot in the environment:

Figure. 32: The settings of empty youbots

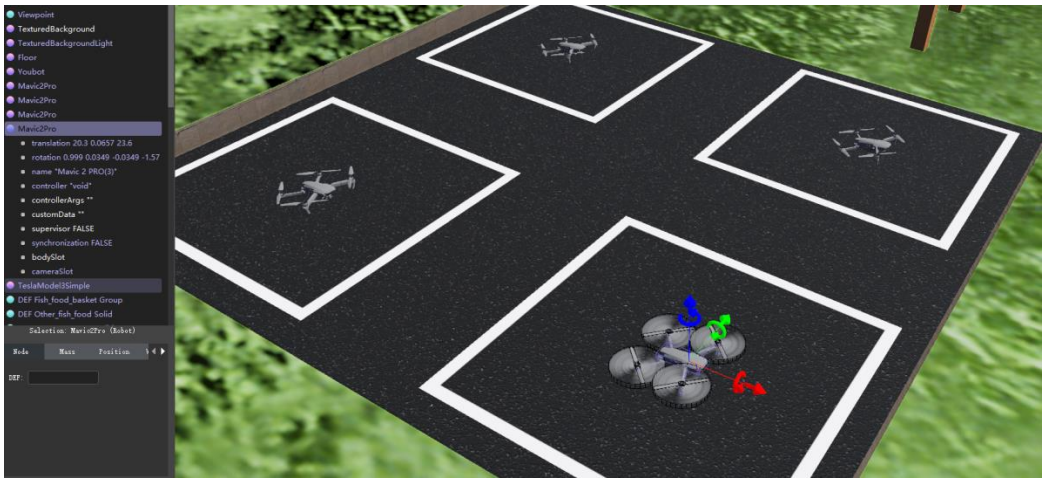


Figure. 33: The settings of empty Marvic2 Pros

In a concession to the tradeoff between environment and operational efficiency, we disable the physical properties of most objects, otherwise it will be very slow and even crash the program. So all of the above setup is built into the running capability of

a properly configured game computer. The specific configuration is as follows:

Table 1: Ordinary Configuration

CPU	I5-3450
GPU	GTX 960
Memory	DDR3 16G

In the above configuration, the running times of the project are as follows: 0.8-1.0 (real time), 1.8-2.3 (Simulation Time). In the final simulation, we used a well-configured computer considering the video conference. The following table shows the advanced configuration we used:

Table 2: Advanced Configuration

CPU	I7-8700k
GPU	GTX 1080 Ti
Memory	DDR4 32G

In the above configuration, the running times of the project are as follows: 0.90-1.0 (real time) and 3.8-4.3 (Simulation Time). Through the test of the map without environment optimization, we can quantify the specific impact of the environment on the project. The following table shows the test results of operation speed before and after environment optimization under ordinary and Advance setting (In Simulation time)

Table 3: Comparison between ordinary and advanced configuration

	Ordinary Configuration	Advanced Configuration
Original Map	2.5-2.9	4.7-5.2
Optimized Map	1.8-2.3	3.8-4.3

Here we introduce the impact ratio to quantify the impact of the environment on us. The impact ratio is the ratio of the number of real simulation multiples in the current computer configuration between the non-environment and the optimized environment. The impact ratio is defined by the following formula:

$$\varepsilon(\textit{Influence Ratio}) = \frac{M_{\text{Optimized}}}{M_{\text{Original}}} \times 100\%$$

Where M_{origin} represents the real simulation multiples running under the original map. $M_{optimized}$ is the true simulation multiplier of the project running in the environment optimized map.

The real simulation multiples are the denoising average of all simulation multiples, but sometimes the simulation multiples will suddenly decrease because of slack, which will cause a certain amount of burrs. For example, under the common configuration, the simulation multiples in the original map are distributed between 2.5-2.9. However, due to the operation of the computer, sometimes the multiples drop sharply (such as 0.8, 1.2), and these points can be regarded as image noises. Therefore, these image noises need to be removed when calculating the average. Since Webots cannot export the simulation multiples, we record the simulation multiples every 10s of the simulation time, and Appendix A is the simulation multiples recorded in all the above cases. Take the common configuration as an example to conduct a complete calculation. The following figure respectively records the simulation multiples of the system in the case of no environmental factors and optimized environment:

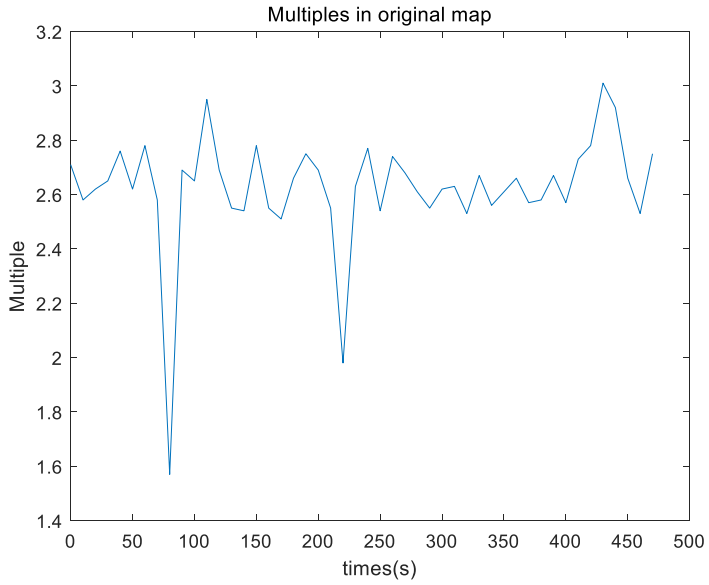


Figure. 34: Multiples in original map with noise

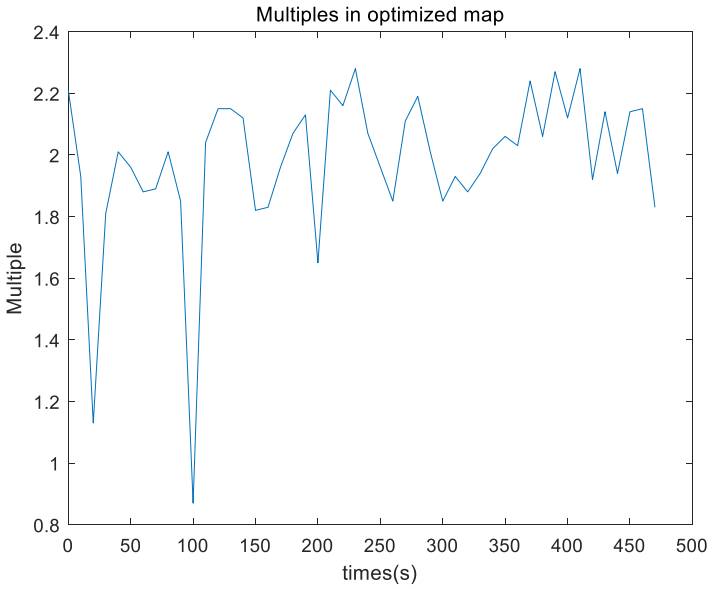


Figure. 35: Multiples in optimized map with noise

It can be seen that there are a lot of image noises in the obtained data, especially in the first half, which is make sense, because the computer does cause some unstable factors when it just runs the simulation. Multiple will gradually smooth until the computer stabilizes. The next step is to remove the image noises. Here we use average filtering. Here, the simplified mean value filtering is defined as:

$$M_i = \frac{M_{i-1} + M_{i+1}}{2}$$

In other words, the average of the before and after values is replaced by the burr value to achieve smooth processing and eliminate the noise. After apply this algorithm, we get a new simulation multiple, which can be regarded as the real simulation multiple, as shown in the figure below:

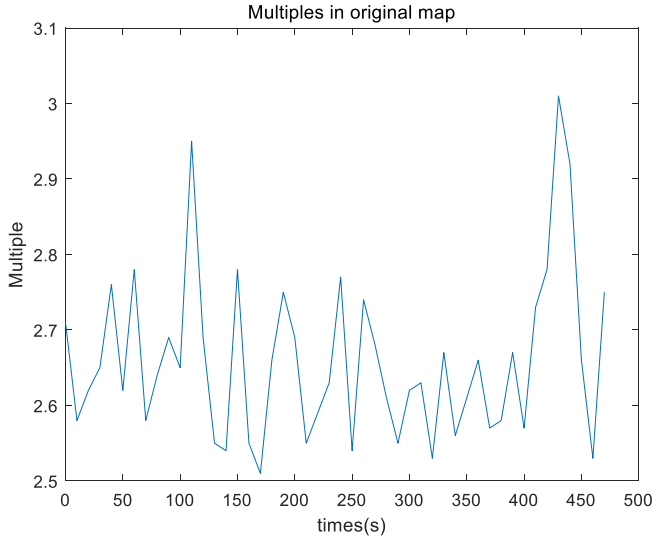


Figure. 36: Filter out the noise for original map

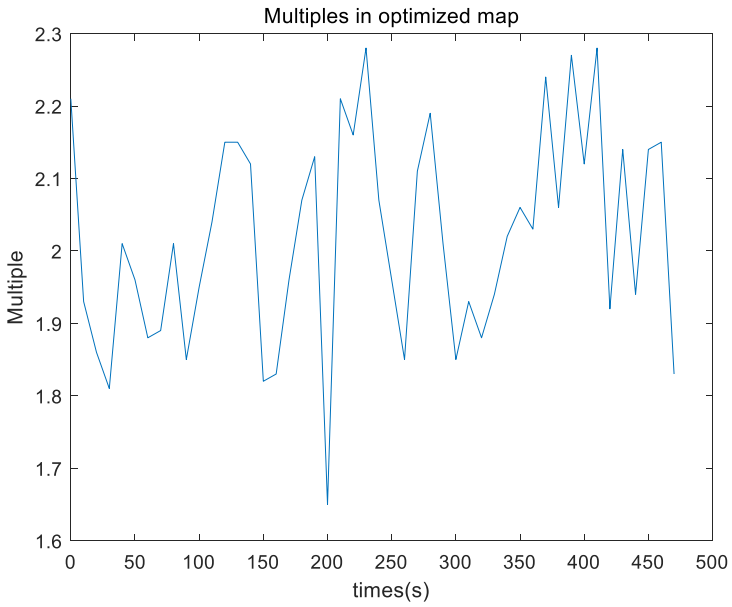


Figure. 37: Filter out the noise for optimized map

Now we can calculate the average of the simulation multiples in these two cases, as shown in the following formula:

$$M_{\text{Original}} = \frac{\sum_{i=1}^{48} M_{1i}}{48} \approx 2.66$$

$$M_{\text{Optimized}} = \frac{\sum_{i=1}^{48} M_{2i}}{48} \approx 2.02$$

Then, the influence ratio is calculated as follows to quantify the performance burden brought by environmental factors to the system:

$$\varepsilon(\textit{Influence Ratio}) = \frac{M_{\text{Optimized}}}{M_{\text{Original}}} \times 100\% = \frac{2.02}{2.66} \times 100\% = 75.94\%$$

This means that our system can be simulated at 76% of the original map performance with multiple environmental factors. This value must be controlled above 50%, otherwise the environment will place excessive burden on the system. And 76% above this indicates that our system can still be simulated with sufficient speed and efficiency in a large number of environments!

To sum up, we need a comprehensive update of the environment at the same time, we can't put too much burden on our computers because of too many environmental factors, so we need to introduce the impact ratio to quantify the performance degradation caused by environmental factors. After consideration, we sacrificed about 24% of our performance to create a beautiful, functional and reasonable environment.

In the end, it was all worth it.

2.1.5.3 Texture Library (Ze Sheng)

Because our map size is 100m * 30m, it is too small for some facilities. For example, the House structure encapsulated inside Webots has a basic size of about 5m, which is too big for our map. As shown below, bungalow Style House is too large to put in our environment normally:

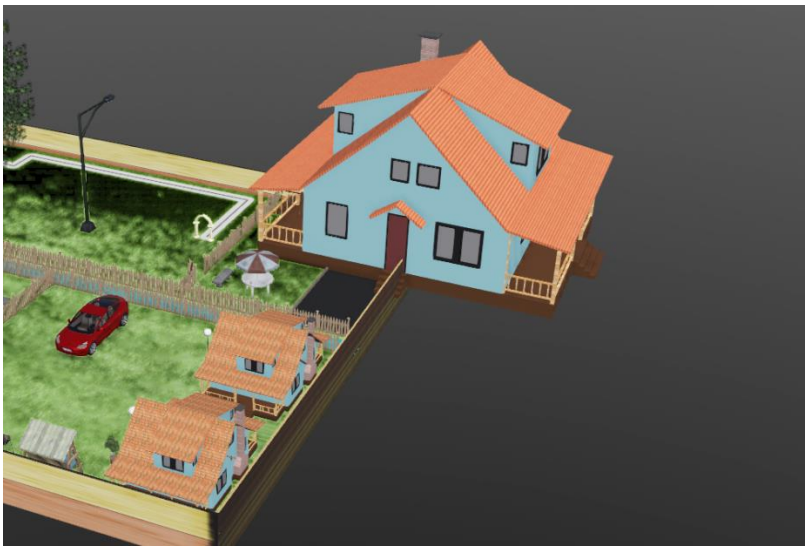


Figure. 38: The house is too large

Therefore, we need to use the function inside Webots: convert into base nodes, as shown in the figure below:

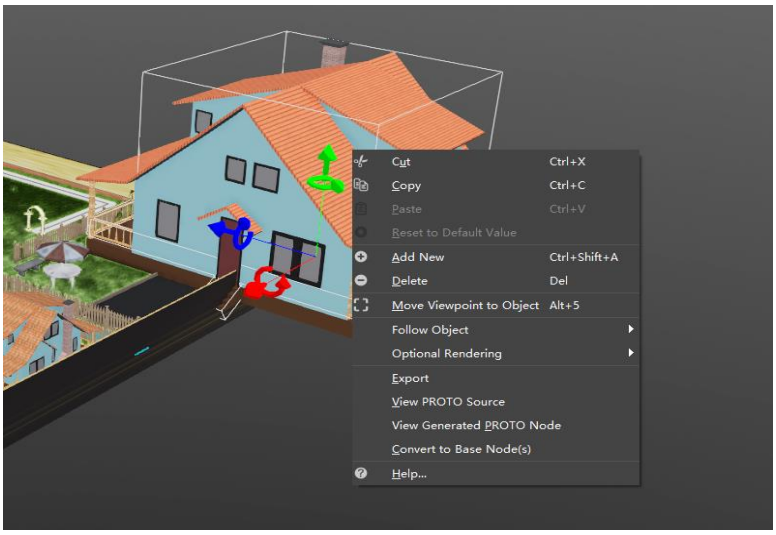


Figure. 39: Convert to Base Nodes

Then the house will be divided into many basic parts, as shown in the following picture:

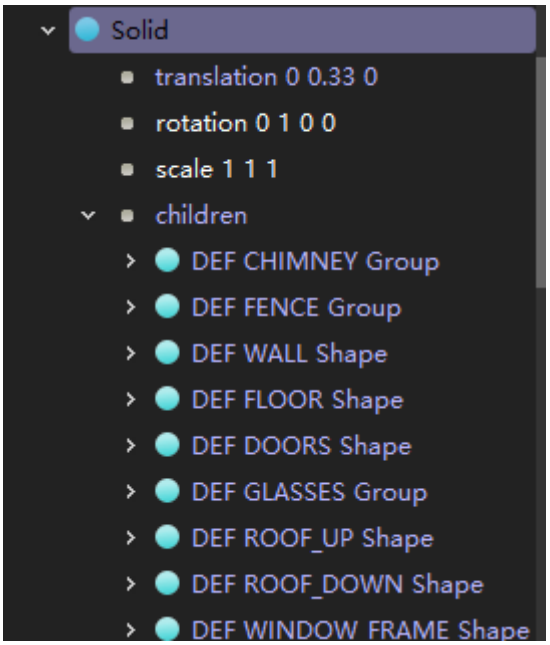


Figure. 40: The base parts of the house

The name of the house changes from 'Bungalowstylehouse' to 'Solid', which means that we can see the parts of the house. As Figure 14 shows, we can change the scale to adjust the size until we are satisfied. What's more we can change every part of the house. There are many parts such as chimney, fence, wall and floor etc.

That is how we change the scale of the house, it's fast and convenient. But here comes a question —— what about the textures? Unfortunately, when we change the node of the house, the textures of each part are saved in the installation path of Webots software. It is really tricky that for different computers, the path will be different, which means that we need to build our own texture library so as to solve this issue. The following image shows the error messages caused by the paths of textures.

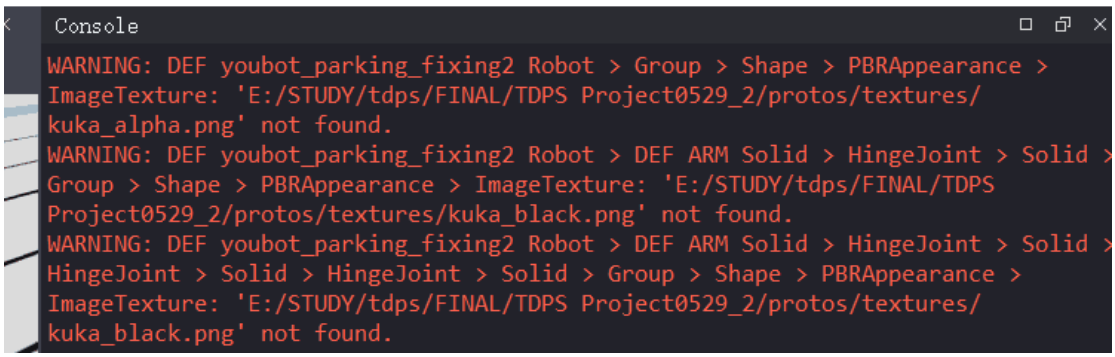


Figure. 41: Errors caused by missing texture

These errors appear when the operation 'convert to base nodes' happens. Not only for the house, but also for our broken youbots in the repairing station.

To tackle this issue, I built our own texture library to store all the textures we need.

Next is to set all the item textures' path to 'texture\...', it was a huge work because there are considerable elements need to be changed. The following picture shows our new path which is built in our world file.

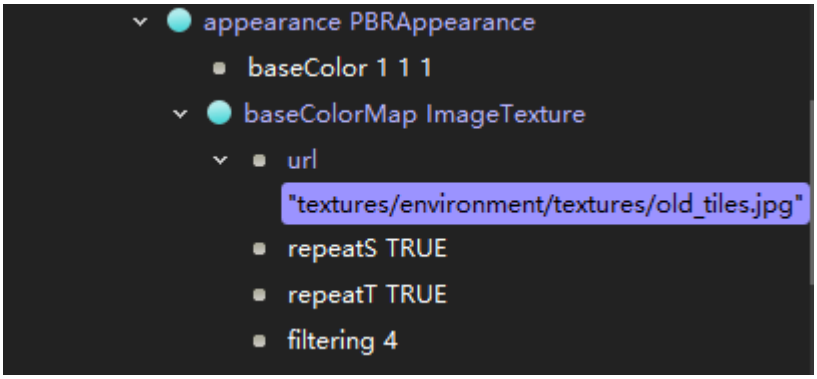


Figure. 42: The path of our library

In this way, we solved the texture missing errors perfectly. What's more, we can add any kind of texture we like and this is an advantage of textures library. However, with the building of the library, the size of our project file is much more larger than that of any other group. But we limited it within 100mb, so it is not too large for downloading.

In general, our own textures library is used to solve textures missing errors. It does work perfectly and by using it we can change different textures we like to decorate our world. So, this is all about the texture library.

2.1.5.4 Import of External Models (Zichao Liu)

2.1.5.4.1 Constructions of Bridge and Arch (Zichao Liu)

1. Setting coefficient of friction on ground and the bridge

Testing the performance of KUKA with a series of friction coefficients and selecting the one that enables it to move stably at a constant speed given a safe power output.

2. Structural analysis of KUKA and redesigning the bridge

Testing the performance of KUKA with different terrains, identifying obstacles for crossing the bridge, and redesign the bridge to allow KUKA to cross it at a constant speed.

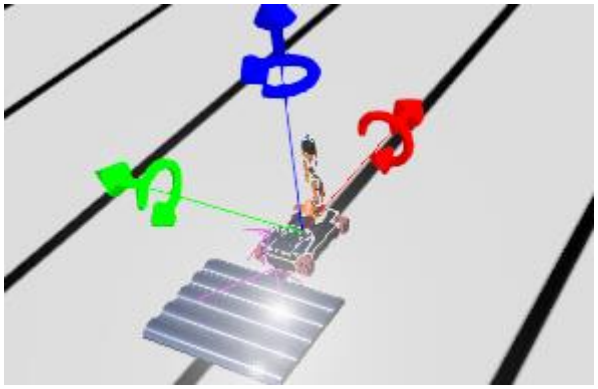


Figure. 43: Test an undulating terrain

3. Model crafting and importation

Crafting the bridge and arch with 3d modeling software and import them into webots.

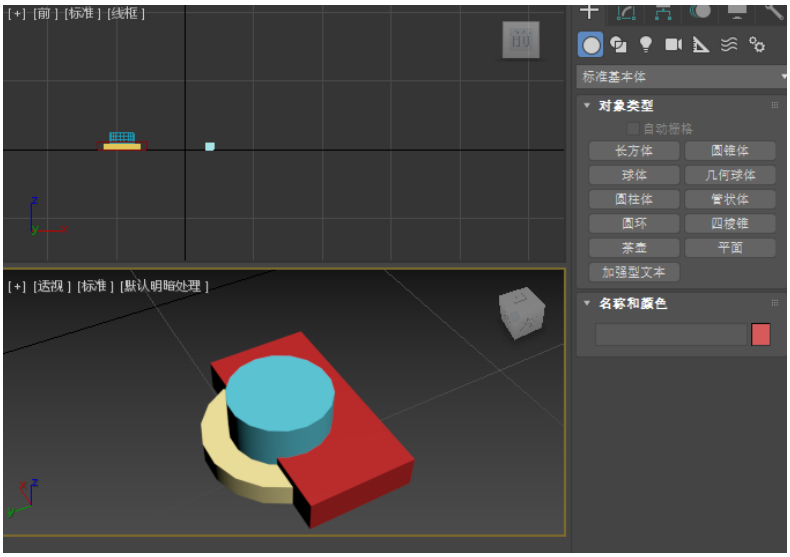


Figure. 44: Craft the bridge in 3DS MAX

Design a 50% closed-loop control based on camera recognition to enable KUKA to hedge trees, find the arch and go through it.

1. Too large friction on the ground will cause physical collapse (console warning), where wheels are all locked dead or move in an uncontrolled pattern. Too small friction causes KUKA to slide on the ramp and unable to climb. According to results, the friction between exterior material of wheels and the ground is set as 20, while on the bridge this number is set as 30.

2. KUKA has a very low chassis, so it can hardly outcome convex terrains with a small radius of curvature. Besides, it has a protruding headstock, prone to collide with the ramp and stop moving. However, KUKA can pass concave terrains stably.

Figure. 45: Chassis locked dead before redesign

Therefore, I designed an arc-shaped bridge floor with a radius of curvature of 2m and added an auxiliary ramp at the start of the bridge, thus preventing collision. With these features KUKA can cross the bridge unhindered.

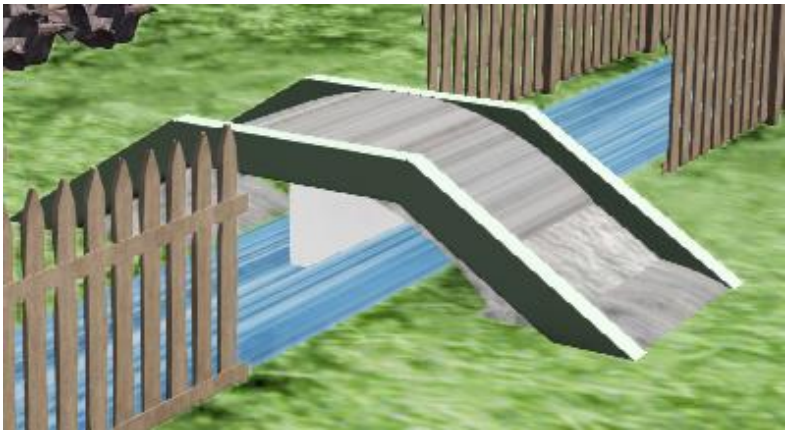


Figure. 46: Refined structure of the bridge

3. VRML97 is the only valid format for importing 3d models to webots. It is a format commonly used in dynamic simulation software like Solidworks, but can also be opened in 3DS MAX, a 3d modeling software majoring in static object crafting. Considering that we only need to build a bridge and an arch, that is enough. The bridge is directly crafted in 3DS MAX. The prototype of the arch is downloaded from[13] that is licensed for non-commercial use, and further adjusted in 3DS MAX to change its scale. After that, models are exported as VRML from 3DS MAX, then imported in webots. By doing so models are stored in world file as a set of vertices, so it requires no change in the root folder.

Figure. 47: Imported arch model

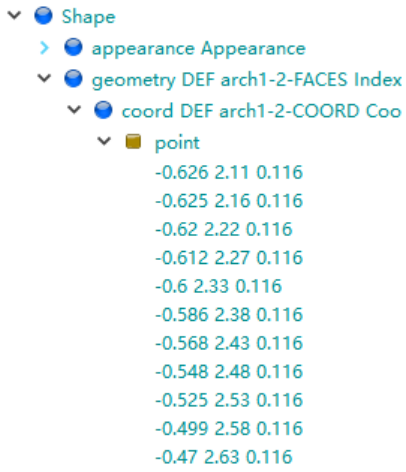


Figure. 48: The model is stored as a set of vertices

4. Webots can export VRML file as well. However, such action will lead to a lower precision in the exported file. For example, when we build a cylinder with a subdivision of 64 in webots and export it, the resulting model will only have about 18 subdivisions, as observed in 3DS MAX.

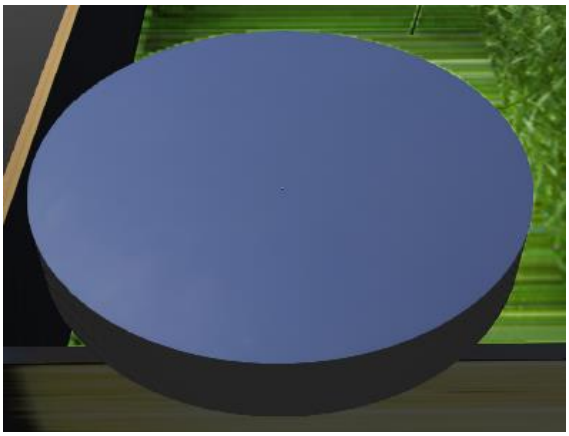


Figure. 49: A cylinder with a subdivision of 64

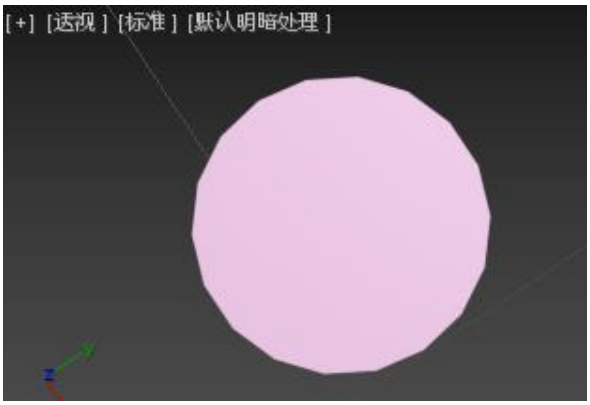


Figure. 50: The cylinder is exported to 3DS MAX

My initial idea was to export the prototype of crafted object from webots, and then modify it in 3DS MAX. However, due to this problem, I changed the procedure so that all objects based on 3d modeling are directly created in 3DS MAX. After that we only need to slightly adjust the scale of imported models in webots.

5. Our method of camera recognition is based on ID of recognized objects. Once there are multiple objects detected, KUKA should neglect those objects irrelevant with the current task. After task3, the bridge should be ignored. As the first version, KUKA judges its place by motion time after swerving to hedge trees, then finds the arch and modifies its direction with recognition. However, when its speed changes, timing parameters must be modified. This is substituted by later versions with better performance.

2.1.5.5 Night Mode (Ze Sheng)

As one of our most impressive features, the night mode is presented in a separate section. As outlined in the synopsis, realistic TDPS will encounter a variety of weather problems, so we need to take this into account in the simulation. Here we chose night as the extreme weather simulation. In the case of night, all street lamps and building lights that are not turned on during the day are turned on now. The following picture shows the environment of our night mode:

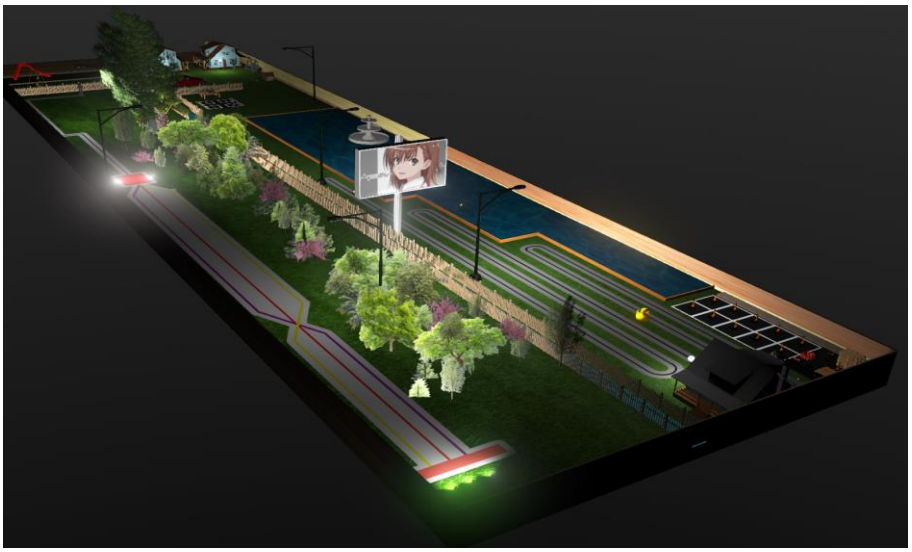


Figure. 51: General map of Night Mode

2.1.5.5.1 The Setting of Night Node & the Change of Environment (Ze Sheng)

By adjusting the exposure property of the viewpoints within Webots, we can darken the world and then disable the background light (setting its brightness to 0) to create a dark environment. The following figure is the parameter setting of the night mode in our project:

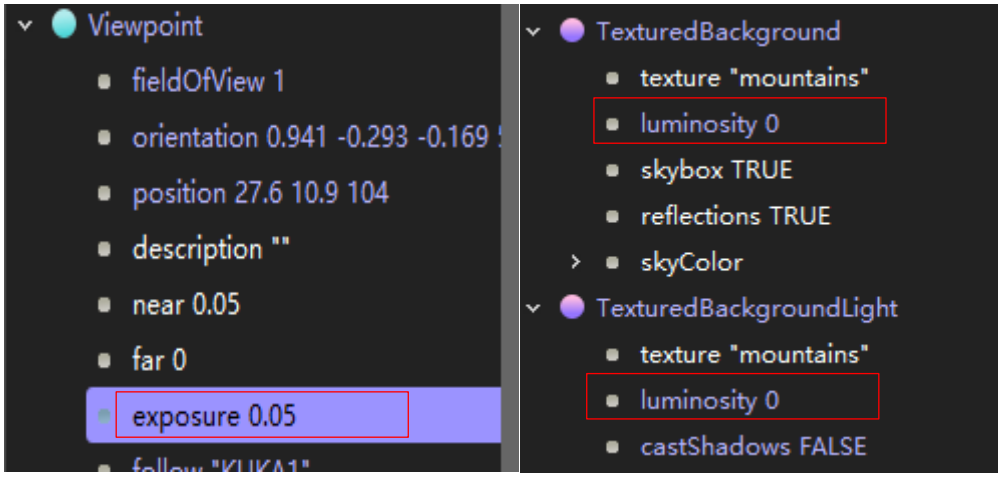


Figure. 52: Settings of Night Mode

Now the whole world is dark. If there were no lights, nothing could be seen. Therefore, we need to turn on all the lights. Taking a streetlight as an example, its brightness can reach the standard of illuminating line through the parameter setting in the figure below.

Figure. 53: The Settings of a streetlight

Similarly, in order for our Youbot 7 to find its way forward, we also installed a construction light on it. Note that the position of the building lights should not hinder the normal use of the manipulator. In order to cope with the change of color recognition threshold caused by the dark state, we also set up light sources at several key locations. The first important location is the Orange box, whether this location is correctly identified affects the normal operation of Task2. Color box and Finish Point in Task5 are the next two, because color recognition in the dark is very difficult, so we use a light source to illuminate it to help our robot to recognize the correct color. The following figure shows the rendering of our point light arrangement:

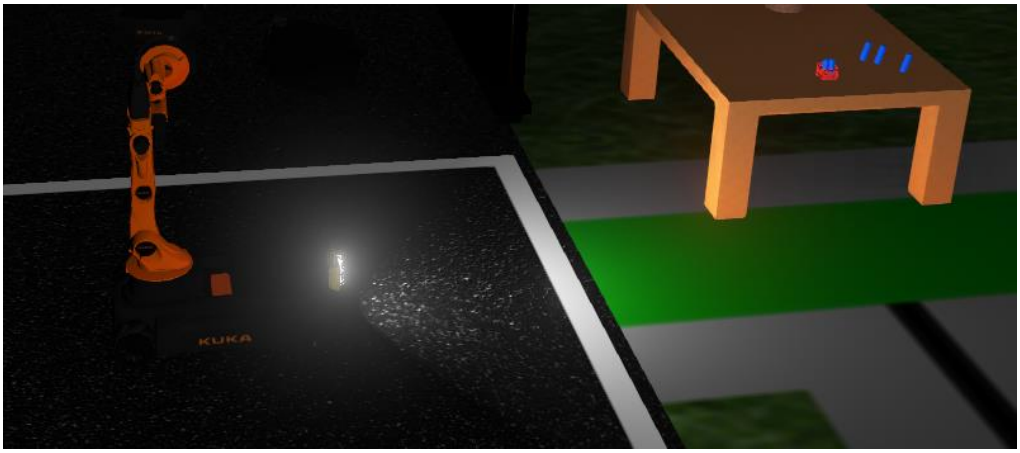


Figure. 54: The construction light on youbot 7

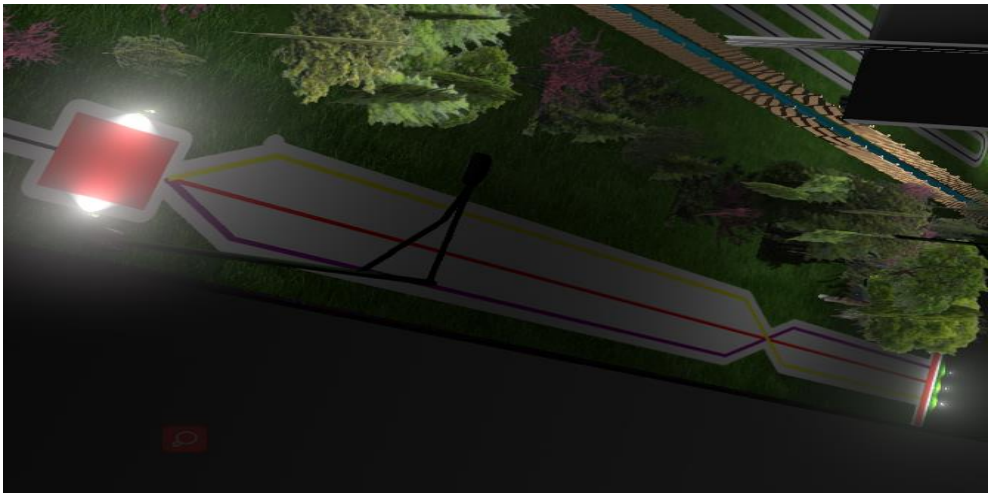


Figure. 55: Color box in task 5

The next part is very important, we don't want to switch controllers when we switch between day and night, we want the two modes to share a Controller. Therefore, how to let our robot recognize the current weather conditions is very necessary. Fortunately, Webots provides a user parameter input location for the robot. This means that we can change this parameter in the World File to tell the robot the current weather, and it can be set once and never change in the future, which is very convenient. Here's how this parameter is set for day and night:

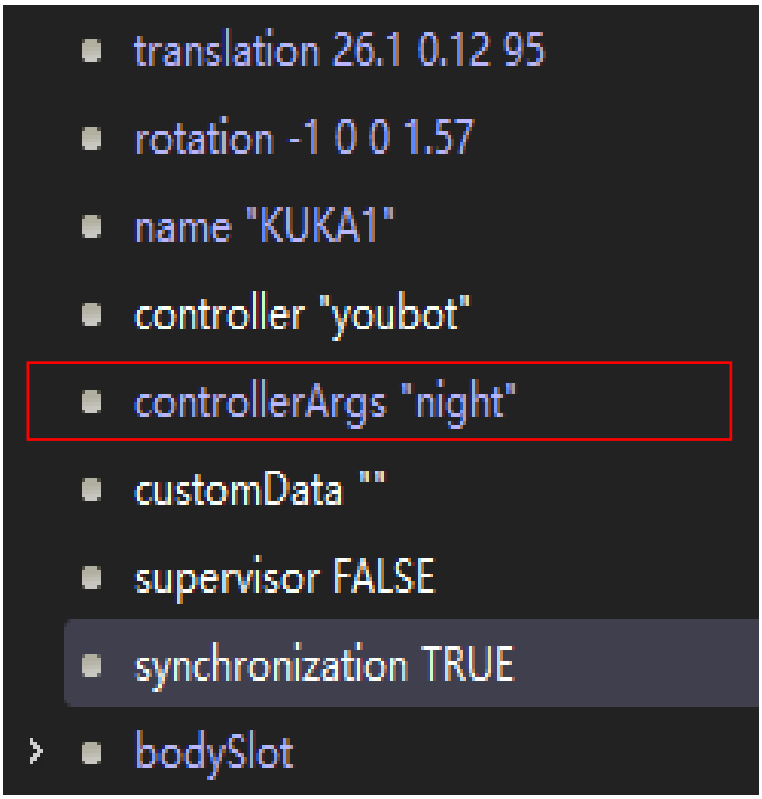


Figure. 56: The key parameter for Night Mode

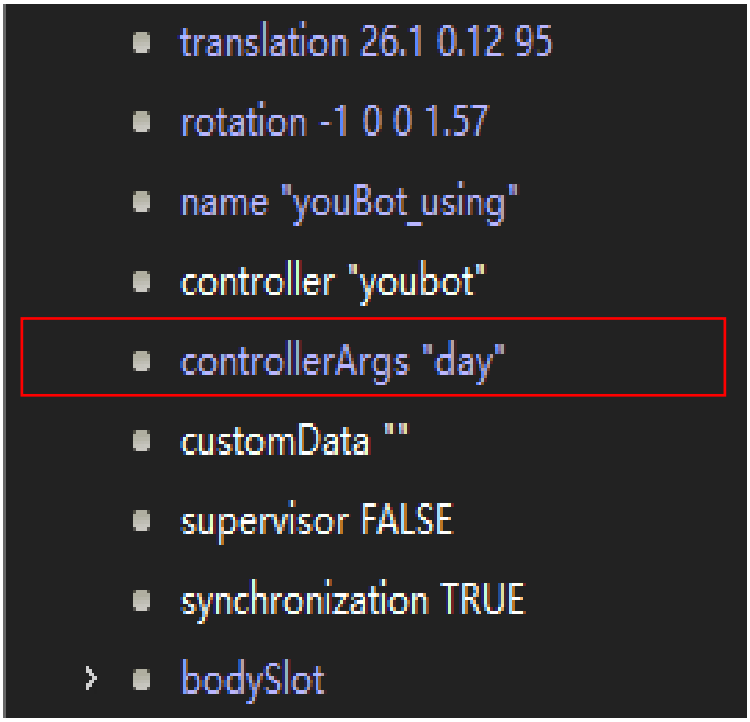


Figure. 57: The key parameter for Day Mode

This parameter is passed to our controller by Webots, so in the code it is an input variable to the main function. Through the if function, we can distinguish between day and night programs. In the part of environment construction, we try not to involve code analysis, so we discuss the specific analysis of the code part in the specific task.

At this point, the part of setting up the environment of night mode has been completed. Because it is in the state of night, some parts must be illuminated by lights, which is a great test to the stability of the algorithm. The two core algorithms in this project, PID line patrol and color recognition, are changed in night mode. Since it involves the principle and analysis of the algorithm, these changes will be explained in the specific task.

Overall, the night mode was very challenging, but we were happy to take the challenge. It turns out that our algorithm is stable enough to deal with the challenges of night, and we have made it! Not only is the night mode a great visual improvement, it also proves that the stability of our whole system is excellent.

2.1.5.5.2 Simulation Performance Analysis of Night Mode (Ze Sheng)

Night mode has a lot of light and shadow compared to day mode, which means computers need to compute more. Therefore, the computer configuration required for night mode should not be lower than our recommended configuration, otherwise it may

be jammed or even crash. Here, we use the influence ratio to analyze the performance ratio of night mode versus day environment-free mode to quantitatively analyze the specific proportion of performance degradation caused by a large number of light and shadow calculations.

Appendix B is the table of simulation multiples that we run under the recommended configuration, where the data has been de-noised by mean filtering, so there is no need to worry about the errors caused by noise. The following figure records the simulation multiplier in the night mode:

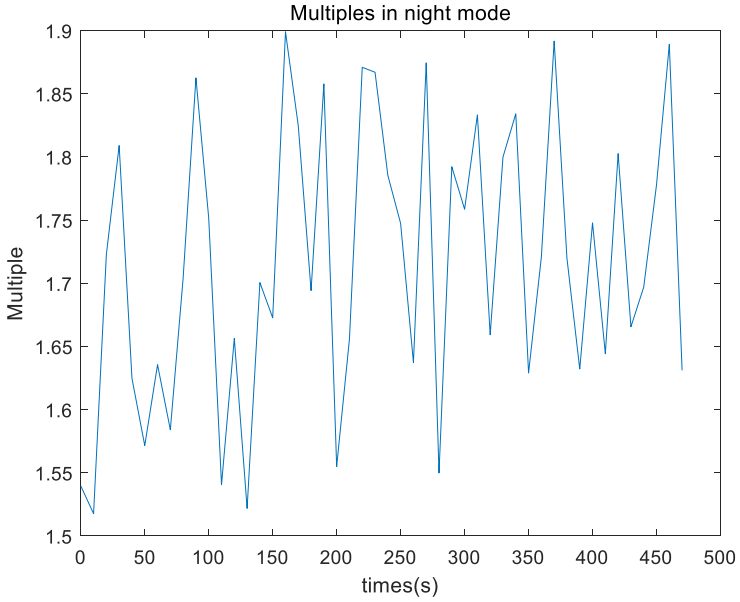


Figure. 58: Multiples in Night Mode without noise

Then, the following formula is used to calculate the average simulation multiple:

$$M_{\text{Original}} = \frac{\sum_{i=1}^{48} M_{1i}}{48} \approx 2.66$$

$$M_{\text{Night}} = \frac{\sum_{i=1}^{48} M_{2i}}{48} \approx 1.72$$

Finally, use the previous formula to calculate the night impact ratio

$$\varepsilon_{\text{Night}} = \frac{M_{\text{Night}}}{M_{\text{Original}}} \times 100\% = \frac{1.72}{2.66} \times 100\% = 64.66\%$$

It can be seen from the above formula that compared with the daytime mode, the performance of the night mode is about 65% of the original performance. Compared to the impact ratio of the optimized system, the night mode consumes approximately 10-15% of the computing resources in light and shadow processing. Next we discuss

whether the impact ratio is improved under advanced configuration. After a lot of tests, we finally determined that the real simulation multiples of night mode in advanced configuration were about 3.2 and 4.9 in no environment. Therefore, we can calculate calculate the influence ratio of night mode under advanced configuration:

$$\varepsilon_{\text{Night}}=\frac{M_{\text{Night}}}{M_{\text{Original}}}\times 100\%=\frac{3.24}{4.93}\times 100\%=65.72\%$$

As expected, night-time mode also had a nearly 35% performance drop compared to no-environment emulation. To sum up, we can conclude that the night mode will operate with an efficiency of about 65-66%. This value is more than 50%, indicating that the light and shadow in the night mode is normal and will not cause excessive burden to the whole system. The following pie chart shows the performance cost of the night mode:

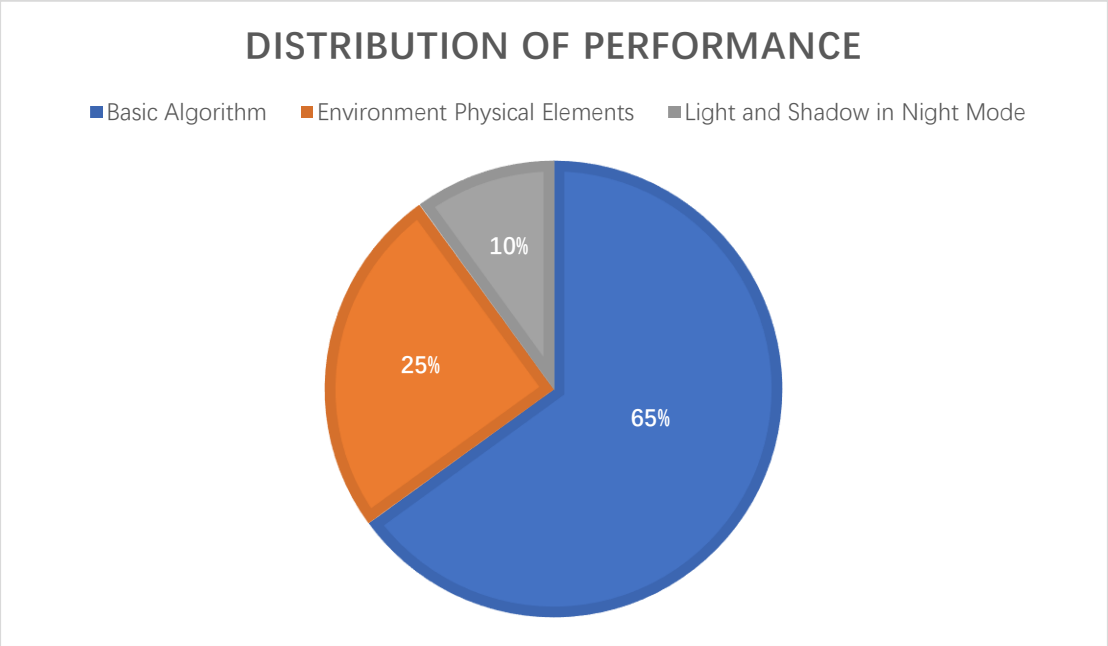


Figure. 59: The distribution of performance

The above is the performance analysis of night mode. Through quantitative data analysis, we can accurately grasp whether the current simulation efficiency is within the expectation, so as to achieve the purpose of controlling the number of environmental variables.

2.2 Selection of Robot and Programming Language Selection (Yukai Song)

2.2.1 Robot Selection: Youbot (Yukai Song)

Technical Content

1. After we finished learning the tutorial, we met the problem of deciding which robot to select for our projects for the future time. I tested several robots from views of size, speed and functions and chose Youbot from KUKA company as our robot.

2. Find the solution of how to add sensors and cameras needed for the KUKA's youbot by adding them in the bodySlot of youbot. This can enhance the function of KUKA's youbot and help it tackle the remaining tasks. The detail of how we install those sensors and cameras in the bodySlot of youbot is shown in the figure below.

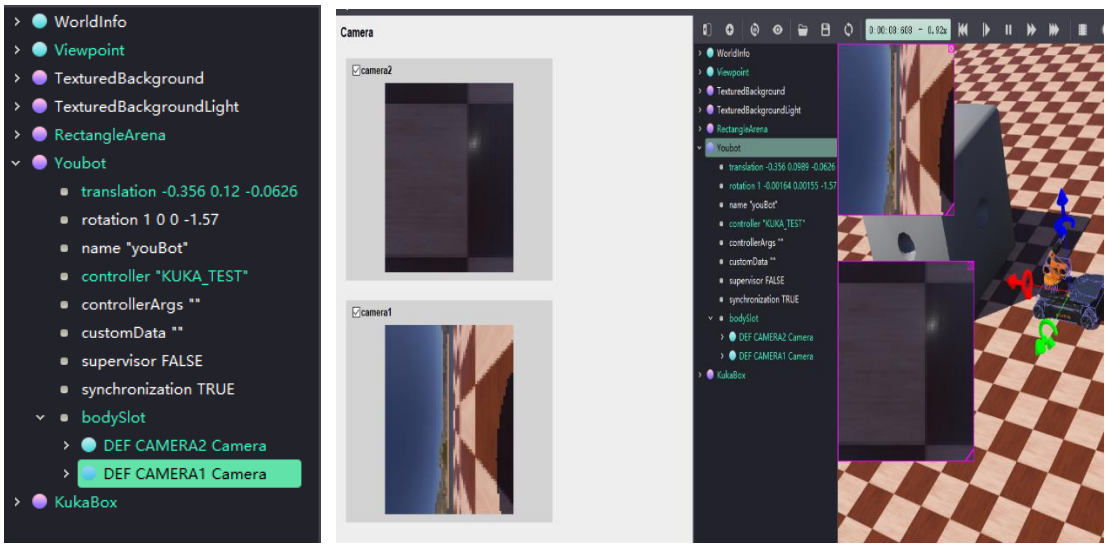


Figure. 60: Cameras in bodyslot

Analysis and Discussion

1. During the process of selecting which robot to use, two robots become potential robots for our project: e-puck and KUKA's youbot, which has different advantages as well as disadvantages. The following figures show those two robots look like.

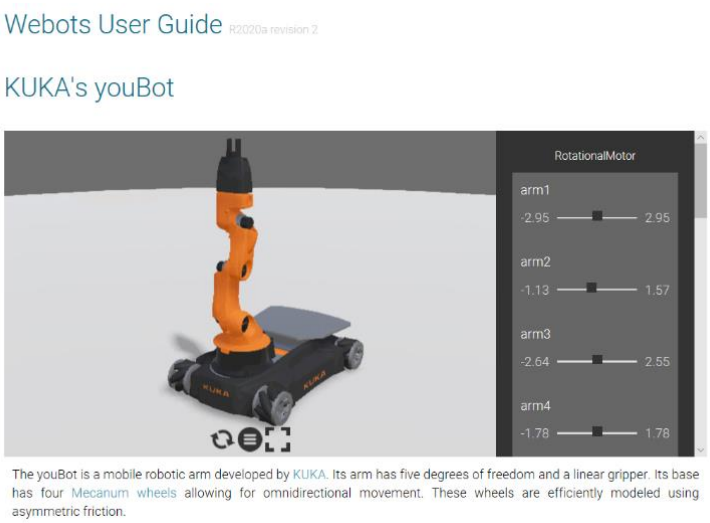


Figure. 61: KUKA's youBot

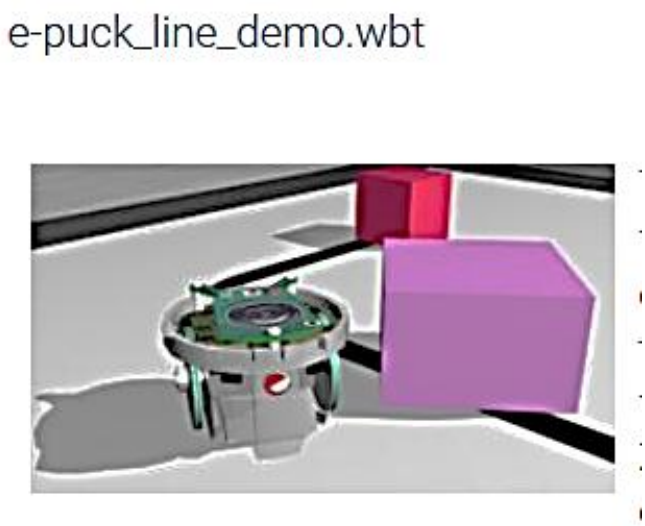


Figure. 62: e-puck

The robot e-puck has six distance sensors which can be used for line tracing directly while the KUKA's youbot owns a robot arm that can be used for grabbing and releasing staff for task 2. When it comes to disadvantages, the e-puck is too small and slow in this project and if we use e-puck for our project, it will cost us more than 40 minutes.

Figure. 63: Compare size of e-puck and KUKA's youbot

In comparison, although KUKA's youbot does not have any sensors, we can add sensors or cameras that we want to use in its bodyslot. At the same time, the youbot is a four-wheeled robot, theoretically faster than our hockey robot and size of it is also satisfactory. It is also important to note that youbot utlizes the Mecanum wheel (a word that can be used in reports to provide a degree of professionalism), which is a universal wheel that can be turned smoothly at will.

2. One of the biggest problems existed in the youbot is that it does not install any sensors in the first hand. Thus, it is impossible for it to complete the tasks of line following, color recognition and object detection without adding sensors. After searching the Webot tutorial, we found out the way to install sensors in Webots by adding sensors in Webots and the details about how we install are shown in the technical content above.

2.2.2 Programming Language Selection (Yukai Song)

Technical Content

1. Apart from the robot, language of the controller is another important part of preparation of the project. We tested several programming languages from views of efficiency and stability in our group member computers and C was selected as our programming language for the controller.

Analysis and Discussion

1. During the choice of language, python, Matlab and C are three popular

languages among five languages provided in Webots. However, when we tested them, we found out that C can work well in computers of all member while python and Matlab get many issues related to the version or environment. Considering the fact that we are far away from each other, it could be hard for us to teach others how to successfully use Matlab or python in other computers. Besides, C is a fundamental course that we learned in year 1 and everyone has done a small project related to it in the microelectronic course. We made our decision to use C as our programming language.

2.3 Line Tracing (Include Task 1 and Part of Task 4&5)

2.3.1 Ground Sensor Algorithm and Testing (Yukai Song)

Technical Content

1. Work with Ze Sheng on groud sensor algorithm for line following and successfully enabled e-puck to follow the lines in the environment that we build. The basic principle of groud sensor algorithm is utilized the idea of proportional control: subtract the distance got from the right ground senor to the distance got from the left ground sensor as the controlling error and use this as feedback to control the speed of left and right motors. To be more specific, if the difference of the right ground senor distance and the left ground sensor distance sensor is zero, the robot is following at the center of the line while if it is not zero, the robot is not tracing the direction of the line and its direction needs to be corrected.

2. Test the mid-value camera algorithm with PID control which is developed by Fangze Xu and find out the problem of mid-value algorithm that it can no longer be useful when the robot meets a right angle turn as in task 4 route. This is mainly due to the reason that the right angle is discrete in terms of angle while the turn in the task 1 is continuous with respect to angle and I report this issue to Fangze Xu and Ze Sheng.

Analysis and Discussion

1. The ground sensor proportional control algorithm for e-puck works perfectly well in the small size map but in this project, the size is too large for e-puck as illustrated before.

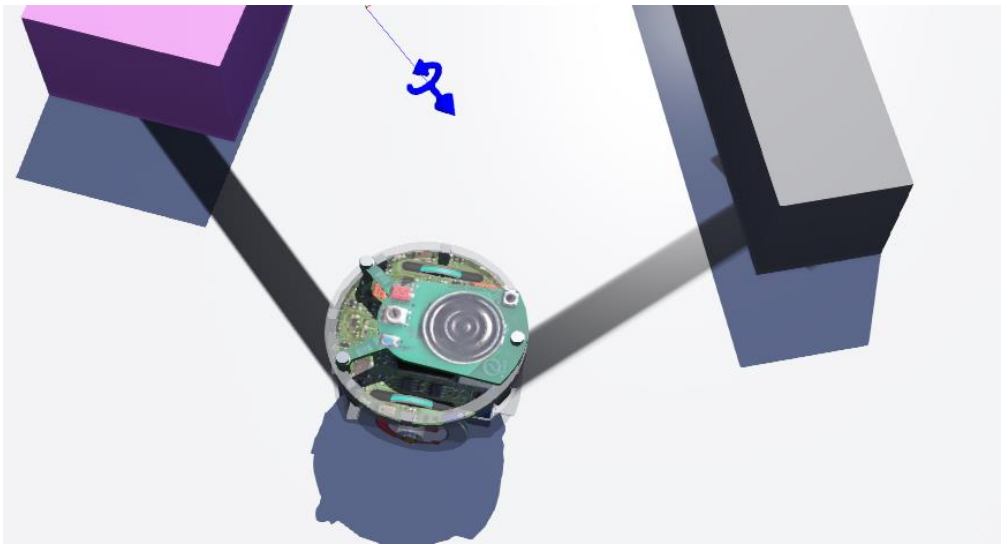


Figure. 64: E-puck works well in the small map

Besides, the position of ground sensors should be perfectly symmetric, which can be hard to achieve. Considered about those disadvantages, we finally decided to choose the youbot camera line following algorithm.

2. The mid-value algorithm for line tracing is no longer useful for the right angle turn in the task 4 is found by me when I put the youbot in the task 4 lines for testing. I discussed why this would happen with Fangze Xu and found out that the discrete in the change of angle might be the problem. They worked out this problem latter in their least square tracing method.

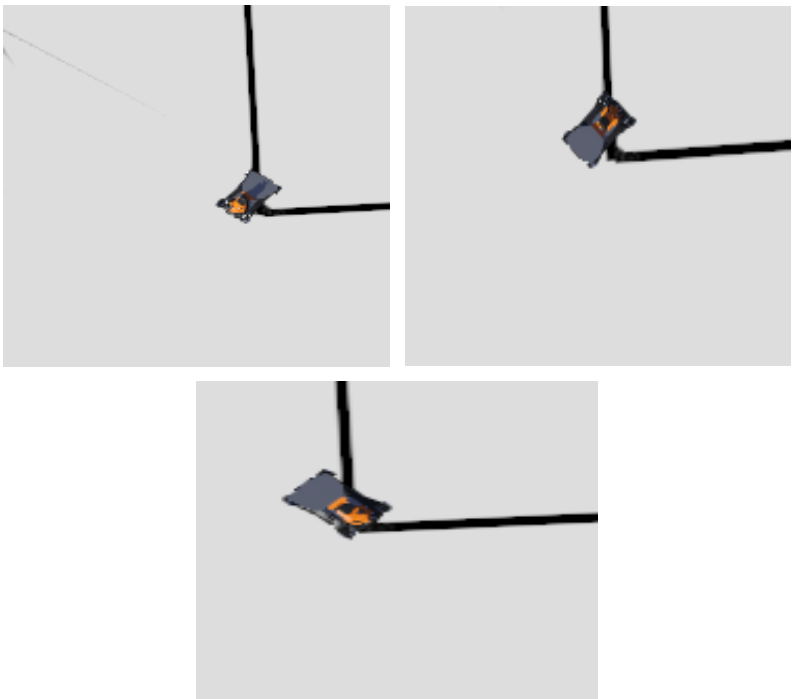


Figure. 65: Mid-value algorithm fails at a turn with a right angle

2.3.2 PID Controlled Line Tracing in Task1,4,5 (Fangze Xu)

Requirements

Programing language: C

The requirements in Task1:

Follow the black line:

Figure. 66: Start point of task 1

Requirements of Task4:

Go through three right turnings

Figure. 67: Right angles in task 4

Requirements of Task5:

Recognize the color box and trace the corresponding line.

Figure. 68: General looking of task 5

The tool I use: Camera1 and next part is our algorithms.

2.3.2.1 Mid-value Method (Fangze Xu)

Technical Content

Use the camera API “wb_camera_get_image” to get the image about the line:

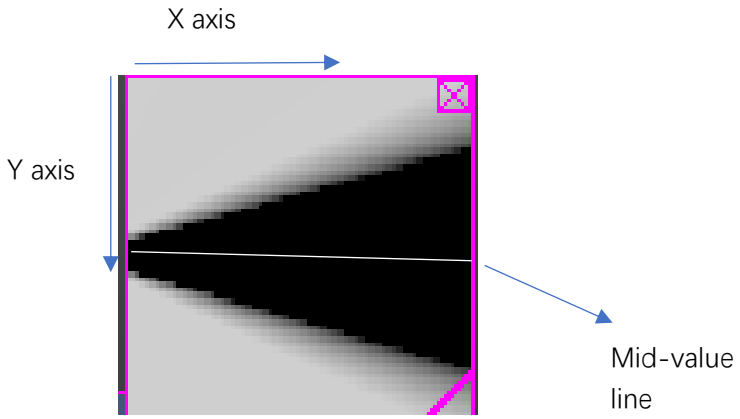


Figure. 69: An Image of the line

Before the further operation, this image needs to be transformed into the black color image by using the camera API "wb_camera_image_get_gray". The width and height of the camera is 64 pixels. We can calculate the mid-value of y-position of the black line to locate the trace. We can use this formula to calculate the mid-value position: the mid-value of the whole picture can be simplified as the uniform value of the summation of the all the mid-value at x-position. Set the mid-value position of the whole image is $T(trace)$, in one x position is T_x, the pixel value(0~254) in one point is $P_{x,y}(the\ pixel\ value\ of\ black\ is\ 0)$, then

$$T_x = \sum_{y=0}^{63} \frac{(254-P_{x,y})\times y}{\sum_{y=0}^{63}(254-P_{x,y})}(1),$$

$$T = \sum_{x=0}^{63} \frac{T_x}{64} = \sum_{x=0}^{63}\sum_{y=0}^{63} \frac{(254-P_{x,y})\times y}{64\times\sum_{y=0}^{63}(254-P_{x,y})}(2).$$

$$\text{We set } \sum_{y=0}^{63}(254 - P_{x,y}) = N_x(3)$$

However, in the practical situation, we find that if there is no line in some x-positions such like the figure shown below, the x-length(camera width) of 64 is inaccurate in the denominator because it makes mid-value smaller than the expected value if the line does not always exist along x-axis. We should only count the number of x-position that has line. We set it as N.

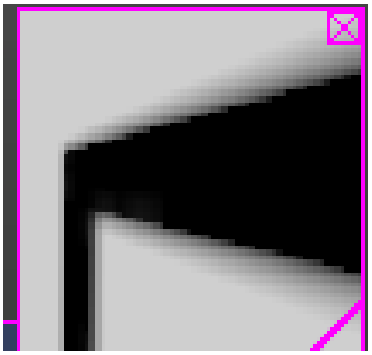


Figure. 70: A right angle in task 4

If we set a threshold value that if $P_{x,y} \leq 50, then\ P_{x,y} = 255, else\ P_{x,y} = 0$, and we set processed y value as$y = y_{x,N} = \begin{cases} y, P_{x,y} \leq 50 \\ 0, P_{x,y} \geq 200 \end{cases}(4), P'_{x,y} = \begin{cases} 1, P_{x,y} \leq 50 \\ 0, P_{x,y} \geq 200 \end{cases}(5)$

The formulas become:

$$T_x = \sum_{y=0}^{63} \frac{P'_{x,y} \times y}{\sum_{y=0}^{63} P'_{x,y}}(6),$$

$$T = \sum_{x=0}^{63} \frac{T_x}{64} = \sum_{x=0}^{63} \sum_{y=0}^{63} \frac{P'_{x,y} \times y}{N \times \sum_{y=0}^{63} P'_{x,y}}(7),$$

$$\sum_{y=0}^{63} P'_{x,y} = N_x(8),$$

then the formula of the mid-value position T can be simplified as $\sum_{x=0}^{63} \sum_{y=0}^{63} \frac{y_{x,N}}{N \times N_x}(9)$. And we can use this value to complete the PID-controlled line tracing.

This is the core code of calculating the mid-value:

```
if(*mode==0){
    for(int x=0;x<image_width_1;x++){//black line
        for(int y=0;y<image_height_1;y++)
            if(wb_camera_image_get_gray(image,image_width_1,x,y)<=50)
pixcount++;//calculating  N_y
        if(pixcount!=0) length++;//calculating the  N
        for(int y=0;y<image_height_1;y++){
         if((wb_camera_image_get_gray(image,image_width_1,x,y)<=50)  &&
pixcount>0)
            singletrace+=((double)y)/pixcount;// ∑_{x=0}^{63} ∑_{y=0}^{63} y_{x,N}/N_x
        }
        pixcount=0;
    }
```

}

singletrace=singletrace/length;// $\sum_{x=0}^{63}\sum_{y=0}^{63}\frac{y_{x,N}}{N\times N_x}$

The algorithms of line tracing in Task5 is the same as task2. Only the camera API function “wb_camera_image_get_gray” cannot be used anymore because the robot needs to recognize different color (yellow, purple, and red). Use “wb_camera_image_get_red/blue/green” instead. The RGB pixel value of yellow is (255,255,0), purple is (160,0,160) and red is (255,0,0), which means only the judging preference $P_{x,y}$ in equation (4)(5) has changed. The final equation (7) is still the same form.

Analysis and Discussion

We find that the mid-value method is very suitable to solve the straight line and continuous turning shown below, which means the arrange of the value of P, I and D could be very wide.

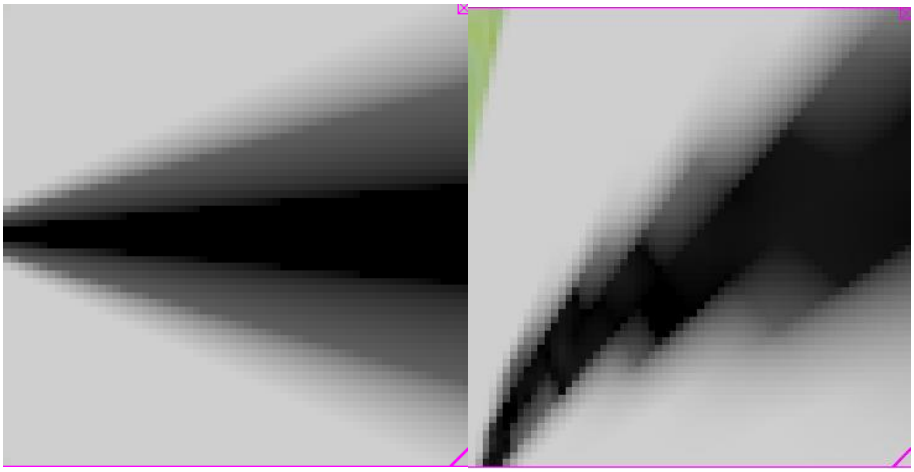


Figure. 71: Straight line and curve in Task1

Compared with simple edge-detection method, which is widely used in line trace, mid-value method has lots of advantages in our enviroment, it can avoid large amount of noise and the its complexity is $O(n^2)$, which is the same as the simple edge-detection method without using recursion. And this method is particularly suitable in our environment because our map design based on the photoshop will be stretched compared with the original jpg format. The environment is so large that it is almost impossible to draw the map on the common computer without being stretched. And this action will blur the edge and make line become the figures shown above.

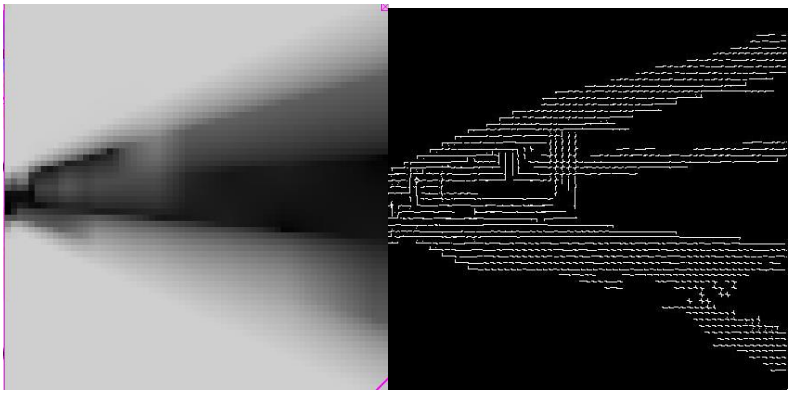


Figure. 72: the edge captured by Canny

And mid-value method is also suitable to handle the line with irregular shape such like the right figure shown above. It is clear that the edge-detect method such like canny cannot handle this blurred irregular line, if we want to improve edge-detect method, we have to increase the size of the camera and the size of this map, which will increase the computer processing time.

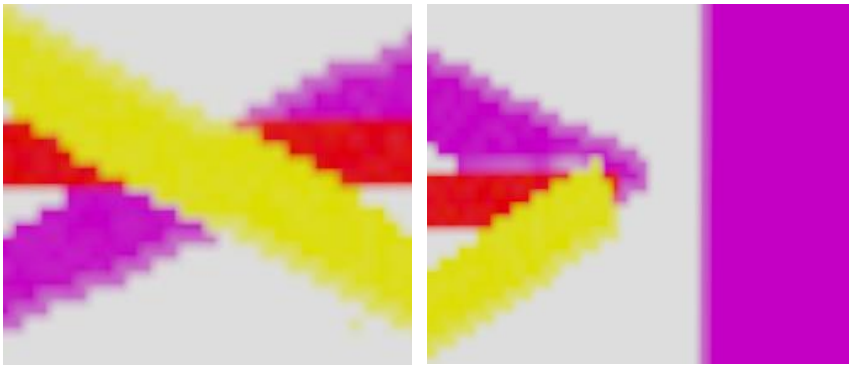


Figure. 73: Overlays and blank in Taks5

The mid-value method can also handle three overlaid lines or bank between two lines. In edge-detect, we need to write additional functions to detect such discontinuity and make corresponding decision. Our method does not have this problem, we can use one function from almost the beginning to the end (only the right turning in task4 need implementation).

In conclusion, because of the mathematical fact that mid-value is the expectation of the y position in one picture, the mid-value represents the general trend of this line. And any small or discontinuous changing of traced line will be attenuated by this method. This will cause a problem while the robot is passing through the right turning and we will introduce an implementation in Results and Analysis. But in general, mid-value requires little on map design, a photoshop freshmen can make the map easily.

2.3.2.2 PID Control (Double PID system) (Fangze Xu)

Technical Content

According to the equation (9), we can get the information that if the line is full of black, then the mid-value is 31.5, we call it the central line. The difference between the central line and the mid-value minus the slop (according to the xy-plane) is the error. And we can build a close loop control based on the error:

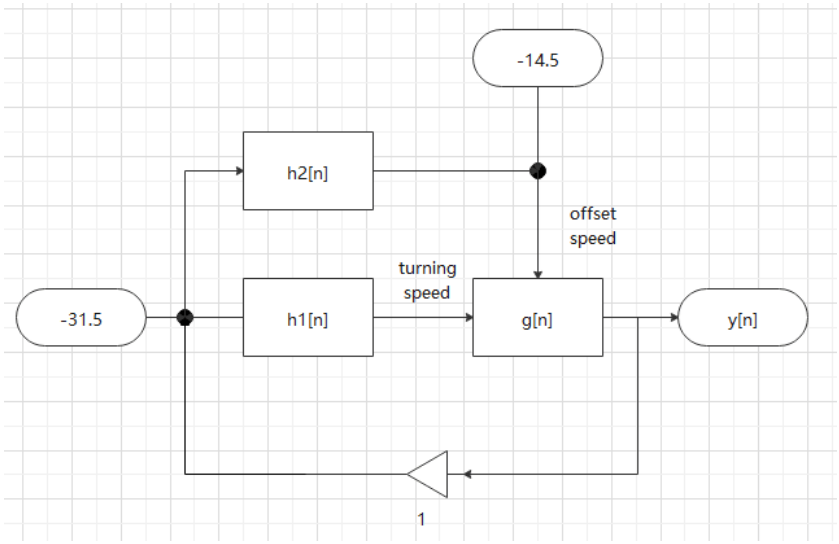


Figure. 74: PID close loop control

Because in webots, the signal is discreate. The integration part becomes summation, and the overall equation of h[n] is:

$H(z) = P + I \times \sum_{i=0}^{n} z^{-i} + D \times (1 - z^{-1})$, g[n] is the plant function and we do not know exactly its specific function because mid-value algorithm is not a linear function which is included in g[n], we cannot get the Z-transform or Laplace-transform function, but we know the input of g[n] is the speed of 4 wheels. And the main purpose of this double PID system is keeping the robot's position closed to the mid-value and move faster.

After that, the algorithms of PID-controlled line tracing part are completed, we can program it based on the algorithms above and adjust P, I and D to find the best suitable value of them. And I will show the codes in the appendix because the algorithms of the line tracing in task5 are the same and I can combine them in one function. Based on the double PID system,

This is the core code of PID-controlled line tracing:

```
double    offset=(3*fabs(error1))+(1*fabs(difference))+(0.5*fabs(integration1))-
```

```
14.5;//offset PID
        if(offset>-9&&(current_task==1||current_task==5))offset=-9;//offset limit
control
        if(offset>-1&&current_task==4)offset=-1;
        double speed_left=(P*error1)+I*integration1+D*difference+offset;
        if (speed_left>14.80) speed_left=14.80;//speed limit control
        else if(speed_left<-14.80) speed_left=-14.80;
        double speed_right=(-1*P*error1)-I*integration1-
D*difference+offset;//turning speed PID
        if(speed_right>14.80) speed_right=14.80;
        else if(speed_right<-14.80) speed_right=-14.80;
        double speeds[4] = {speed_left,speed_right,speed_left,speed_right};
        base_set_wheel_speeds_helper(speeds);
```

Analysis and Discussion

The four figures shown below are the data of error collected when robot is passing through the first turning with different PID gain value in Task1.

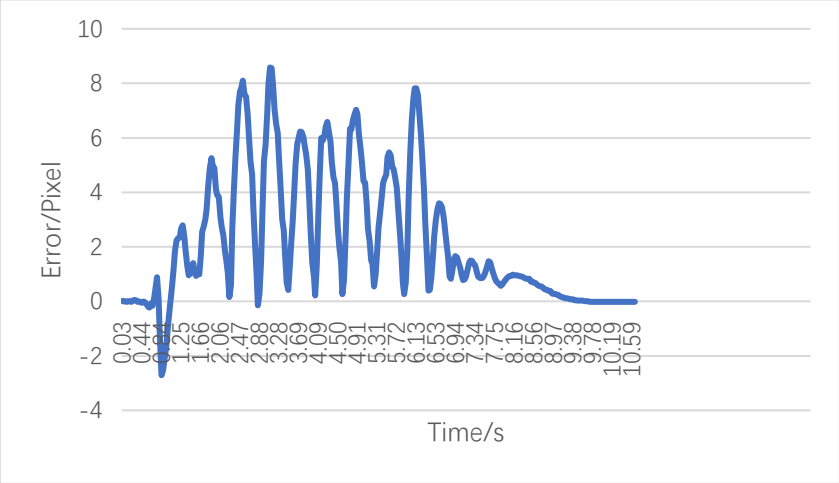


Figure. 75: PID is 3.0,0.9,0.1

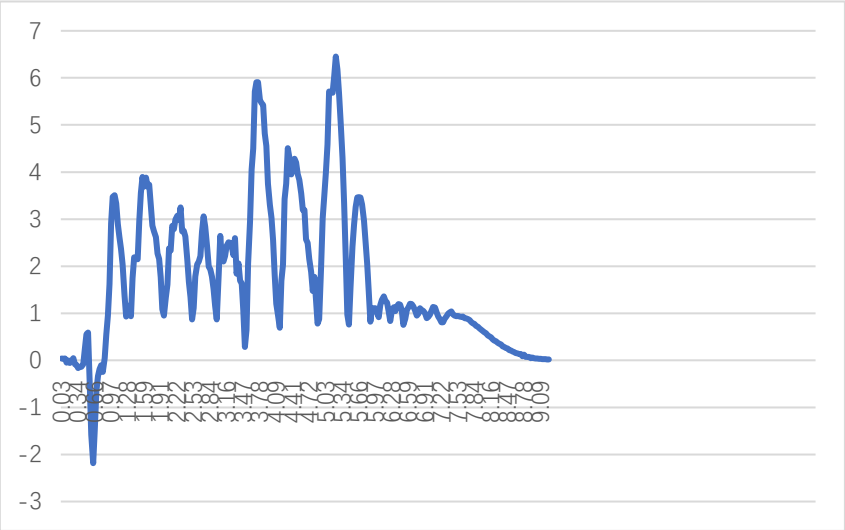


Figure. 76: PID is 0.5,0.9,0.1

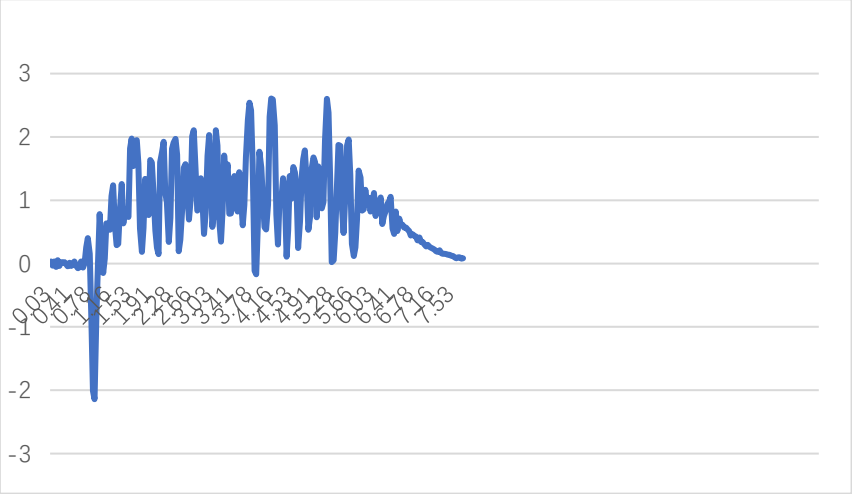


Figure. 77: PID is 3.0,0.9,1.0

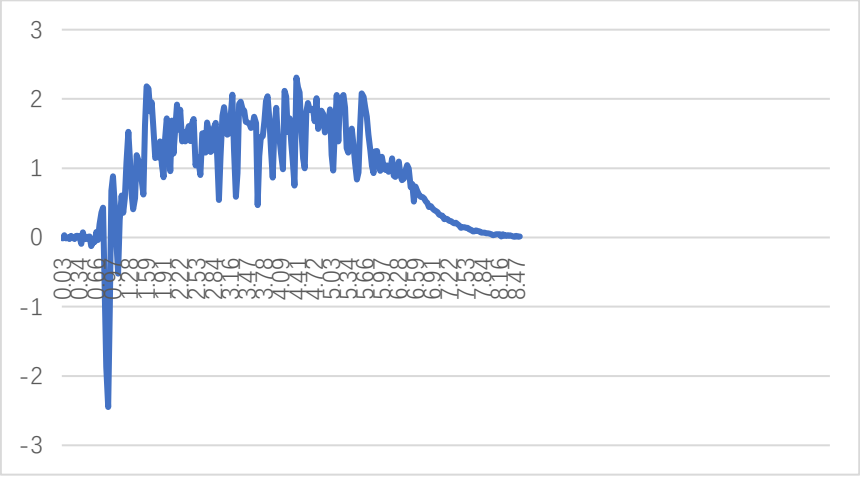


Figure. 78: PID is 3.0,0.1,0.1

From these four figures we can find that if we change P (figure1 vs figure2), the magnitude of the response will be changed, if we change D(figure1 vs figure3), the time between each peak value is also changed which means the frequency of the

response is changed, and if we change I, (figure1 vs figure4) the average error is changed, or we can say the response is closer or further away to 0 axis. And this result fit the function of PID control well. Based on characteristic we find about our PID close loop system, we can set a suitable PID gain in keep the average error close to zero and reduce the amplitude of the oscillation.

However, we find that when the robot is passing the right turning, mid-value method does not work, the system response to the right turning is too slow.

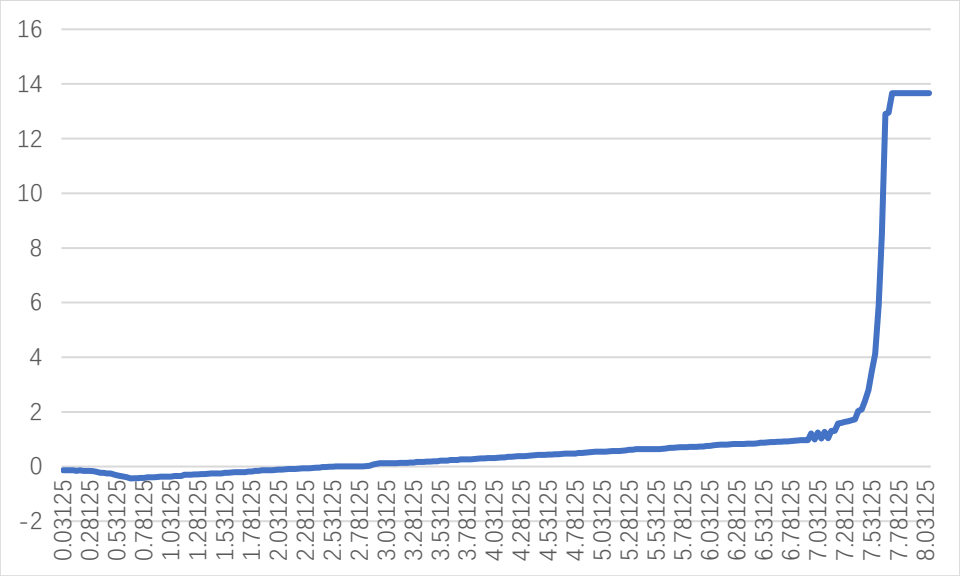


Figure. 79: Error of the right turning

From figure5, we can see that its error takes less than 0.2 second from less than 2 to nearly 14, so it is almost impossible to only adjust PID to increase its response speed. We need another way in solve right turning.

2.3.2.3 Colorful Box and Colorful Line Recognition (Day Time in Task 5) (Fangze Xu)

While the robot is recognizing color box, I set the threshold value (very large) of the number of pixels that have specific color (yellow, purple, and red). And the threshold can avoid the noise color because noise color usually cannot have so many pixels.

And I also use the same algorithm while the robot is approaching to the stop position. The threshold value can determine where the robot should stop.

This is the code (although this code only represents recognizing red part, the codes of other two colors are almost the same except the RGB value):

```
for(int x=0;x<image_width_1;x++)
```

```
        for(int y=0;y<image_height_1;y++){
            int r=wb_camera_image_get_red(image,image_width_1,x,y);
            int g=wb_camera_image_get_green(image,image_width_1,x,y);
            int b=wb_camera_image_get_blue(image,image_width_1,x,y);
            if(r>=200 && g<=50 && b<=50) //rgb value
       pixcountR++;// if it is red
        }
      if(pixcountR>1000){//threshold value
        *mode=1;// corresponding mode to the PIDtracing(black line-tracing, blue line-tracing, purple line-tracing or green line-tracing)
      }
```

2.3.2.4 Least Square Method (in Task 4) (Fangze Xu)

There are still some critical situations that mid-value method cannot handle them. Thanks to my team leader, Yu Kaisong's mention, I find that the robot cannot go through the right turning in Task4. The mid-value calculation for vertical turning can only produce a small error because the changes of the mid-value at each x-position is reduced by the average operation. It is not useful to handle the trace with large discontinuity like vertical turning or cross road. In one interesting case, if the line nearly parallel to y-axis like the picture shown below, the mid-value of the picture is very closed to 31.5, but we know that the robot is far away from the traced line.

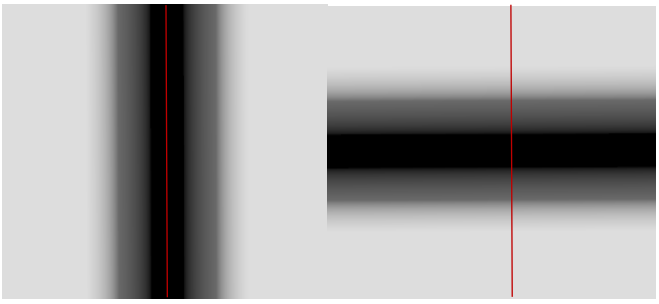


Figure. 80: The mid-value if these two figures are the same

We need new information about the robot: the angle between the robot and traced line. We can use least square method the find the linear fitting curve of the picture. And the its slop can represent the angle of the fitting line.

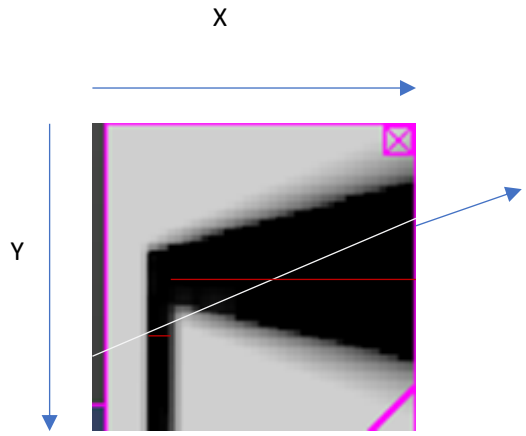
The formula of least square method is

$k = \frac{\sum(x_i-\bar{x})\times(y_i-\bar{y})}{\sum(x_i-\bar{x})^2}$,k is the slop if the fitting curve, in this situation we do not need to

consider the intersection in y-axis.

x_i is the position that has a mid-value, and y_i is the mid-value.

And according to the formula above, we can draw a fitting line in the figure below.



Least squared line of all the mid-value at specific x-position

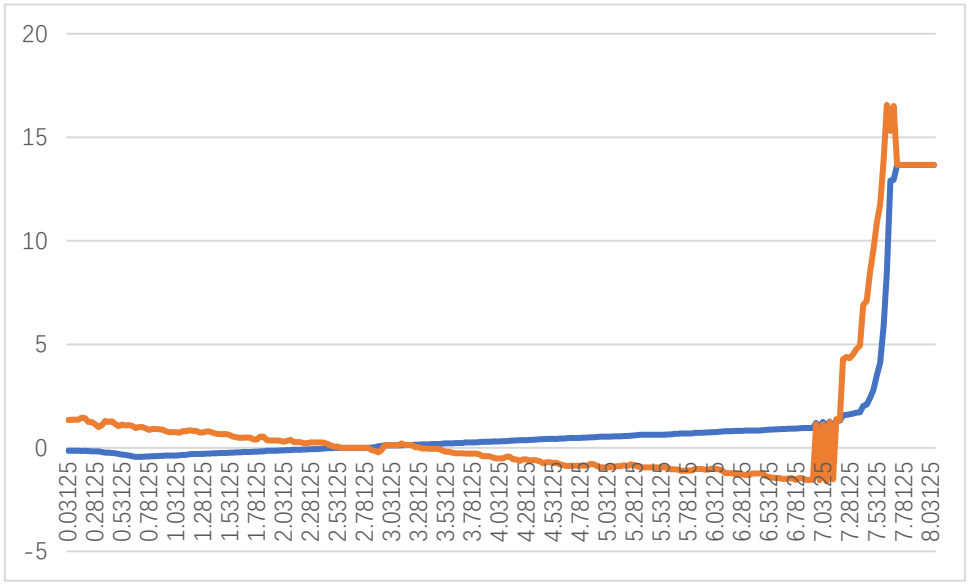


Figure. 81: The error of system with least-square method(red) and original(blue)

From this graph, we can see that it takes about 0.8s to get from 2 to maximum error. And it gives robot three times than the original time to respond.

2.4 Movement and Dynamic Design of Robot Arms（Ze Sheng）

2.4.1 Abstract of Robot Arm (Ze Sheng)

In this chapter, I evaluated the various robots in Webots and selected the best one -- youbot, an old instructional robot made by Kuka. In this chapter I will explain how we can use youbot's arm and how we can power it. Specifically, we will analyze and explain the whole set of fish food taking actions before Task1.

2.4.1.1 Introduction to the Usage of the Robot Arm (Ze Sheng)

The robot arm built by Youbot has five degrees of freedom, which can rotate within a certain range. The following is a small program of the orientation and rotation Angle of the robot arm:

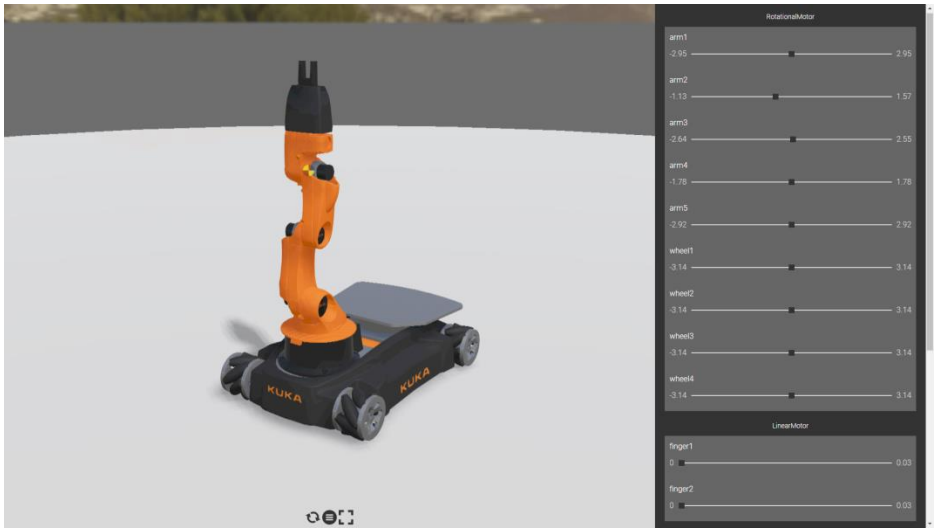


Figure. 82: The control panel of youbot

Note that this program is very important and you can find it on the Webots website. This program is the basis for all the changes we make to the arm. In this project, the positions and poses of the robotic arms provided by WEBOTS did not meet our requirements, so we defined all the positions and poses of the robotic arms by ourselves. As shown in the figure above, the mechanical arm is controlled by five motors respectively, so the Angle control of these five motors is the foundation and key to realize the control of the mechanical arm. In the code, we obtain the control by transferring the handles of the five mechanical arms, as shown in the figure below:

```
568   arm_elements[ARM1] = wb_robot_get_device("arm1");
569   arm_elements[ARM2] = wb_robot_get_device("arm2");
570   arm_elements[ARM3] = wb_robot_get_device("arm3");
571   arm_elements[ARM4] = wb_robot_get_device("arm4");
572   arm_elements[ARM5] = wb_robot_get_device("arm5");
```

Figure. 83: 5 elements of a whole arm

The code above is simple enough to understand without much explanation. Now we have control of the various parts of the arm, and we can change its attitude by making adjustments to the various joints. With the small program shown above, we can preset the parameters for each action and then fine-tune them in Webots.

After the manipulator arm is adjusted, the next step is to let the fingers on the

manipulator arm perform the grasping action. In Webots, this is a function that has been packaged and written in gripper's header file, which can be called directly.

2.4.2 Grab and Release the Fish Food (Ze Sheng)

Now that we know how to operate the robotic arm and robot, we can pick up and feed the fish! Here, the idea is that we need to change the parameters of the robot to fit the environment, not change the environment to fit the robot. In short, we do not adjust the height of the table to adapt to the mechanical claws, but we adjust the position of the mechanical claws to accurately grasp the object, because environmental factors are difficult to change in reality, we need to take this seriously.

In the choice of fish food, we used capsule type fish food, and the choice was based on reality. As shown in the figure below, we can also find the capsule type of fish food on the shopping website:

Figure. 84: The capsule we chose

Here, for the sake of authenticity, we put the fish food in a small basket, because in reality, no one would feed the fish directly with their hands. It's worth noting that this adds a lot of difficulty to the algorithm because our machine can't just skim over the top of the head or the lures will spill out, so we need to find a way to solve these problems.

In this part, we have creatively used three methods to achieve accurate robotic arm

retrieval. Position (or pose) set method, middle state method and mechanical claw propulsion method. On the basis of this method, our robotic arm can achieve the absolute accuracy of the object, the success rate is nearly 100%

In the following part, the fish feeding operation before Task1 and feeding operation in Task2 will be analyzed and expounded in detail through the explanation of the three methods. In contrast, fish feeding will not be covered in the technical details of Task2.

2.4.2.1 Position Set Method (Ze Sheng)

Position set method is a method of forming a fixed position by collectively defining five joints of a mechanical arm. Each fixed position is equivalent to a set containing all the position information of each joint motor. This method is the basis of manipulator control in this project, so it is very important.

In the whole project, we set five sets, namely: 1. Grasping position set, 2. Feeding position set, 3. Grasping - placing position set, 4. Placing - throwing position set and 5. Middle position set. The following table shows the specific element parameters corresponding to each set, namely, the rad of each joint.

Table 4: Position of arm

	Grab Positon Set	Throw Positon Set	Grab-Put Positon Set	Put-Throw Positon Set	Middle Positon Set
Arm1 (rad)	0	0	0	-0.24	0
Arm2 (rad)	0	1.1	-0.4	0.66	0.6
Arm3 (rad)	-0.95	0.5	-2.35	1.95	2
Arm4 (rad)	-1.2	0.4	-0.2	-0.65	-0.6
Arm5 (rad)	0	0	0	0	0

These sets in the table above correspond to different actions, and both the part where the lures are taken and placed in Task1 and the feeding part in Task2 play an important role. Of course, just defining these moves is not enough, in order to improve the success rate, you must have more precise design. That's the middle state method

and the mechanical claw propulsion.

2.4.2.2 Middle State Method (Ze Sheng)

The middle state is no less important than the individual pose set, and it usually changes the rotation of only one joint (task2 has an middle state using the attitude set method) to adjust the whole system. The middle state has a lot of functions, specifically the following points:

1. Connects each pose set: In this part, the middle state makes a smoother and more reasonable connection between the attitude sets. In the catch phase, if you jump directly from state 1 to state 3, the robot arm will swing around in the air, which means our bait will fall out. As shown in the figure below:

Figure. 85: Failure of grabbing the fish food

To avoid this, we use a middle state that rotates Arm1 half a turn to the left, as shown in the figure below:

Figure. 86: Success of grabbing fish food

This allows perfect access without spilling the capsule. There are many similar

connections in the project, but this is a typical example.

2. Perfects the physical factors: Collisions between components within youbot are ignored because of the default Settings of Webots. So, there is the possibility of mechanical claws going through the body, which we don't want to have. So we designed an middle state to avoid this. As shown in the figure below, in the absence of an middle state, the mechanical claws will droop after placing the basket, resulting in model errors.

Figure. 87: Physic crash caused by model error

To avoid this, the middle state is our best weapon. After placing the basket, we set an middle position with the Arm4 at the 1.4-radian position (i.e., the mechanical claw moving up). The following is what happens when the middle state is set:

Figure. 88: Solution to physical issue

In this way, this problem can be solved perfectly, but it also brings new problems. In the process of moving the mechanical claw up, it is easy to meet the basket, which will cause the position deviation, which will seriously affect the success rate of picking and throwing on Task2. Therefore, we need to use the mechanical claw advancing method to solve this problem.

3. Avoid unnecessary friction and collision: In the process of grasping the object,

the mechanical claw has a movement of the base to the left, which causes the basket to have a friction on the table. If this friction is too high, it may cause the body to deviate unpredictably, which is not what we would expect. We have drawn a schematic of the friction force as shown in the figure below

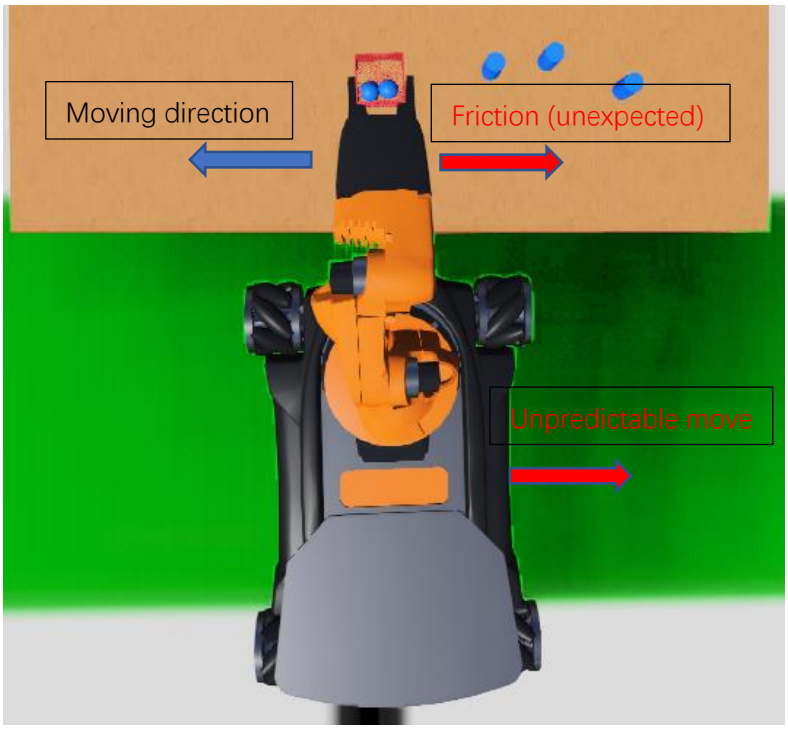


Figure. 89: Friction may cause unexpected move

For this, we designed the middle state: the Arm2 moves to 0.2 radians, which means we lift the object slightly to avoid contact with the table and eliminate friction.

In addition, in the fish-feeding section of Task2, we used Middle State written in the attitude set. In the absence of this middle state, the manipulator will place the basket on the tray after throwing the bait, which will result in collision with the Orange box. In severe cases, the robot gets stuck on the Orange box. Therefore, we designed this middle state, so that after feeding, the robot arm would recover to the middle state before placing the basket, which could perfectly avoid the collision with the box. See specific flowchart in the example part later.

The above is the idea and function of the middle state method. As a harmonic agent of the robot arm, the middle state method plays a great role, which not only makes the movement of the robot smooth, but also solves many thorny problems.

2.4.2.3 Mechanical Claw Propulsion Method (Ze Sheng)

Mechanical claw propulsion is one of the most important methodologies in this

section. It greatly improves the success rate of grasping and is one of the proudest improvements in our group. Because in our project, the basket had to be placed until Task2 to be picked up again, during which time the collision of mechanical claws and relative motion due to robot movement and its own inertia were inevitable. To put it simply, it is logical that the basket should be placed on the tray in an unpredictable way. So this exposes the fatal flaw in the first two approaches, which are too precise. All states are very accurate, while offsets are unpredictable, which directly leads to our success rate in Task2 is only about 70%. In order to be steady, we once wanted to give up the whole picking process and change the mechanical claw to maintain the fetching posture to Task2. As shown in the figure below, before using this method, we often encounter failures:

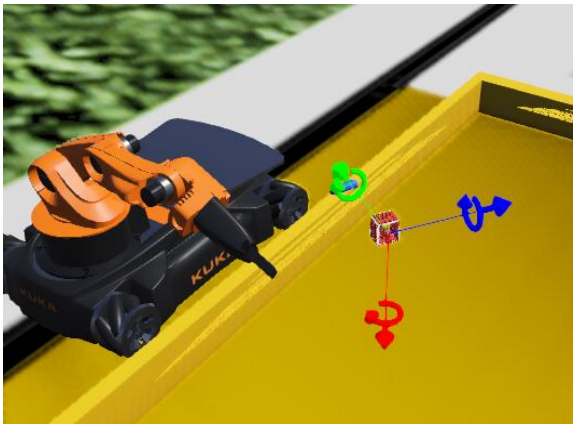


Figure. 90: Fail to hold our box

After summary, there are mainly two types of failure: 1. Unable to pick up the object correctly; 2.The root cause of both failures is that the manipulator arm is too precise and cannot tolerate the deviation of the position of the object and thus cannot hold the object properly. So, we developed mechanical claw propulsion method to solve this problem.

The principle of mechanical claw propulsion method is very simple, which changes the grasping attitude of mechanical claw from static to dynamic. In simple terms, it is to push and take, so that the mechanical claw actually sweeps a distance, and as long as the offset of the object is within this distance, the object is guaranteed to be taken 100.The following is the train of thought for this approach:

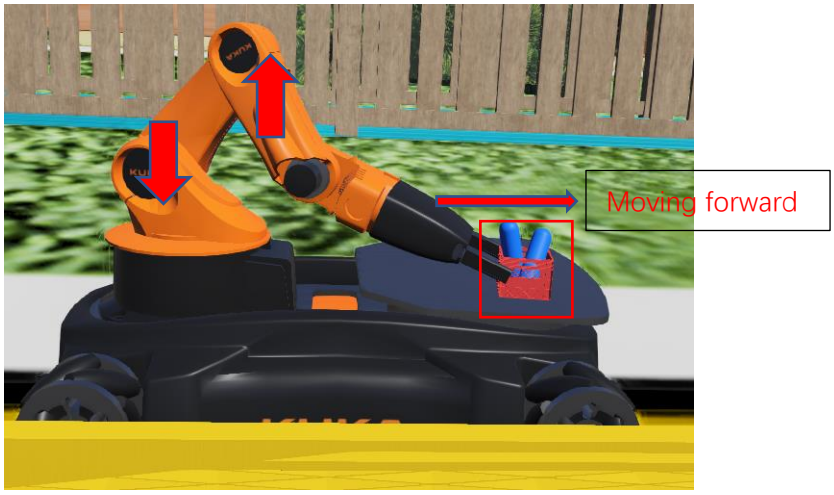


Figure. 91: Gripper only holds the edge of the box

As shown above, the mechanical claw is going to grab the edge of the object, which is very unstable, and we want the mechanical claw to grab more of the object. So we control the joints of the robot arm for dynamic grasping. By reducing the height of Arm2 and increasing the height of Arm3, we realize dynamic grasping of objects. The following is the rendering:

Figure. 92: Improved method can grab a larger part of the box

You can see that the mechanical claw dynamically grabs the object, which means that the deflection of the object will not affect the stability of the grab. Through a lot of simulations, we can compare the mechanical claw propulsion method to improve the success rate is very significant. See the table below for specific data.

Table 5: Comparison between original and mechanical claw propulsion method

	Original Method	Mechanical claw propulsion
Success Rates （30 times）	63.3% （19 times）	100% （30 times）

To sum up, mechanical claw propulsion is a necessary method to realize the perfect

grasping of objects, and the most core method of the whole grasping part. With this method, the success rate of grasping is greatly improved, that is, the stability of the system is improved. The system also has a tolerance for the small deviation of the object, which is also a significant improvement in the anti-interference ability of the system.

2.4.3 Example: Task 1 and Task 2 (Ze Sheng)

By using the core algorithm for the three manipulator operations mentioned above, we can combine the operations before Task1 and Task2.

The following flow chart clearly illustrates this process:

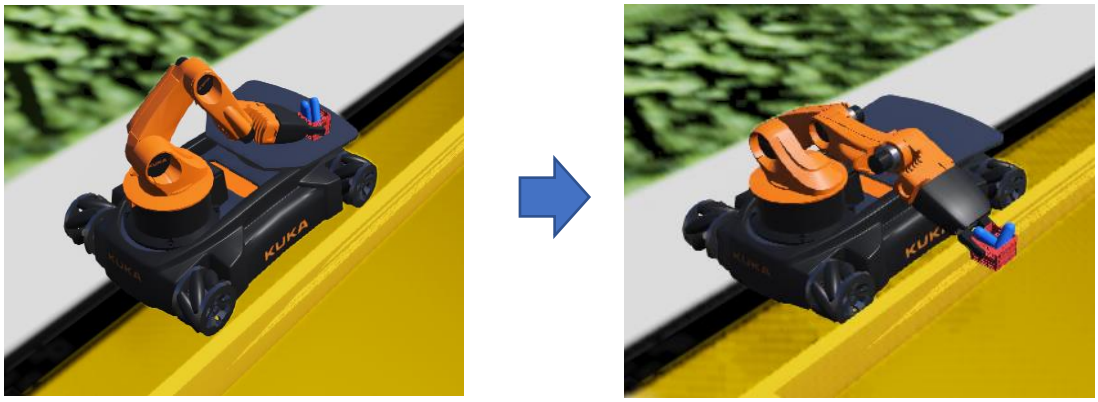
Grabbing（set 1）

Turning（middle state）

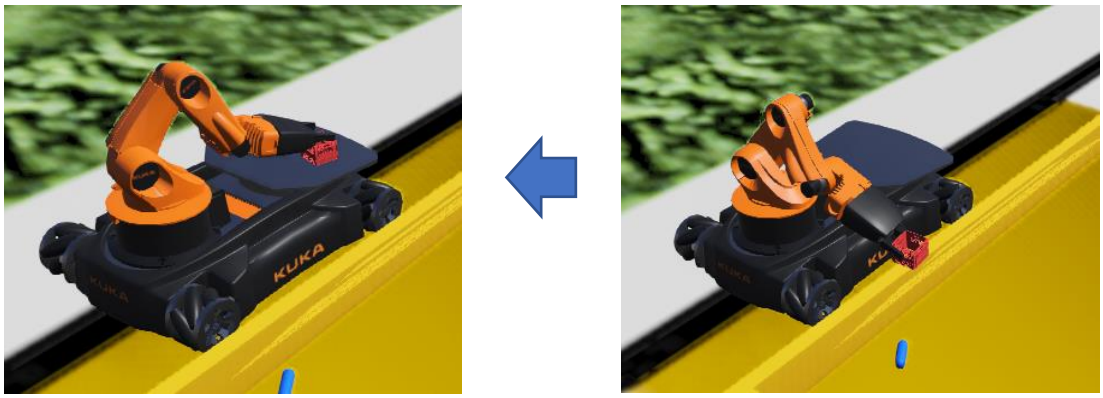
Placing（set 3）

Figure. 93: Grabbing part before task 1

And The figure below shows task2 completion：



Put-Throw position (set 4) + claw propulsion　　Throw（set 2）



Grab-put position（set 2）　　Arm back（middle state）

Figure. 94: Movements in task 2

As shown in the figure above, the three methods cooperate with each other to give full play to their advantages and jointly complete the complex and high-precision grasping and feeding actions. At this point, the operation of the robot arm is over. The operation mode of the robot arm and the detailed explanation of our three original methods are introduced to verify the accuracy, stability and anti-interference of the whole system. This is a very challenging part to consider.

2.4.4 Supplement of Arm Usage (Codes Example)(Dingte Yang)

The code that control the arm:

```
gripper_release();
arm_set_movement(4);
passive_wait(4);
wb_motor_set_position(arm_elements[ARM4], -0.63);
passive_wait(0.3);
wb_motor_set_position(arm_elements[ARM2], 0.76);
```

```
wb_motor_set_position(arm_elements[ARM3], 1.81);
passive_wait(0.3);
gripper_grip();
passive_wait(0.5);
arm_set_orientation(ARM_LEFT);
passive_wait(1.5);
arm_set_movement(2);//Throw the fish food
```

The strategy is also used in the beginning to grab the food before task1.

Analysis and Discussion

I did the job to control the arm with Ze Sheng, Jiajun Huang and Weihao Xiong. Other team members decided to use the KUKA because it have preset codes to control the arm to move and grab, and it will be convenient in food releasing in task2. However we find that sometimes the arm will stuck on the table and push the robot, which will arise error in task3. So, we made a discussion and decided to write the arm movement code by ourselves. The arm will arise high so that it will not touch the orange box, then we controlled the arm and let it hold the food, drop the third joint proportionally until the food touch the table. This step need accurate distance control and we tried many different distances to make sure the arm will not push the robot.

2.5 Color Recognition Algorithm (Ze Sheng)

In this chapter, color recognition is our main algorithm. Our group creatively proposed a threshold color recognition method based on image segmentation and inverse traversal, which is an algorithm with good anti-interference and strong performance. We will elaborate on the details of this algorithm. In addition, the feeding part of fish food has been described in detail in the posture part of the robot, so it is not necessary to elaborate here.

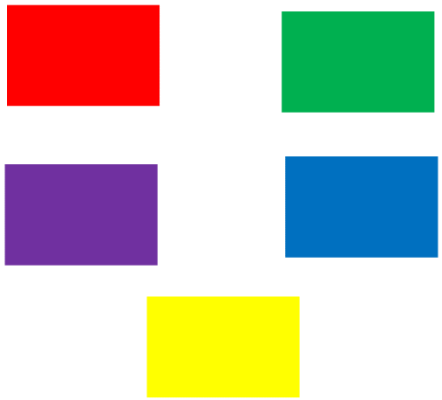
2.5.1 Color Recognition Algorithm Based on Double Threshold Comparison and Image Segmentation-inverse Traversal (Ze Sheng)

As one of the core algorithms of this project, color recognition plays an important role in both Task2 and Task5. In this chapter, I will introduce in detail the color recognition algorithm based on threshold comparison and image segment-inverse traversal independently developed by our group. In order to enhance the coherence, we

first introduce the basic dual-threshold comparison color recognition algorithm, then introduce the image segmentation and inverse traversal algorithm, and finally through performance analysis to verify the anti-interference and intelligence of the algorithm.

2.5.1.1 Double Threshold Comparison Method (Ze Sheng)

Double threshold comparison method is the basic part of color recognition. Here we introduce color threshold and quantity threshold to design the algorithm. In computers, any color is made up of R,G, and B. This means that any image can be divided into R, G, and B layers corresponding to the depth of its red, green, and blue. These three layers are combined to create the final color image. Using this principle, we predefined thresholds for all the colors we need to recognize in this project. As shown in the figure below, through the definition of the range of three variables R,G and B, we can get the following corresponding threshold of color, which is called the color threshold:



White: R=0, G=0, B=0

Red: R>200, G<20, B<20

Green: R<20, G>200, B<20

Yellow: R>150, G>150, B<50

Blue: R<20, G<20, B>200

Purple: R>150, G<50, B>150

Figure. 95: Different colors and their color thresholds

Note that the RGB value here is based on UINT8 color threshold notation, where all colors are represented from light to dark from 0 to 255.

The above is the definition of the color threshold. In order to enhance the anti-interference and intelligence of the algorithm, we introduce the quantity threshold to realize the "double insurance" of the algorithm.

Considering that there will be a lot of color interference in reality, which will lead to robot misrecognition and misoperation, we must guarantee the stability and anti-interference of the algorithm. If you only use the color threshold, that is, recognize a yellow pixel point to do task2 immediately, this is clearly unscientific. If there is color

or noise interference in the distance, the whole algorithm will fail, so we need the quantity threshold to cooperate with the color threshold to achieve intelligent operation.

Quantity threshold is the minimum number of recognized pixels that the robot reaches the specified position when it is off the patrol. The following flow chart illustrates the relationship between color threshold and quantity threshold:

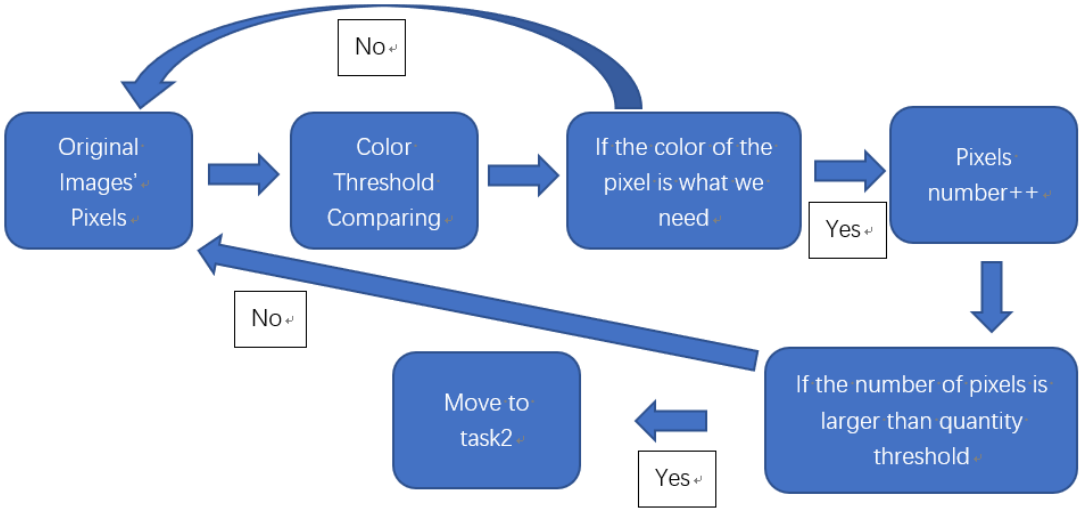


Figure. 96: The design of color recognition

By comparing the number of pixels with the threshold value, we can enhance the stability of the system. When the required pixel points in the graph (yellow in Task2) are greater than the quantity threshold, task2 is performed; otherwise, image pixel traversal is always performed. In Task2, we have a quantitative threshold of 90, which will be explained in the section of quantitative analysis and Optimal location. The following is the schematic diagram of our color recognition and bait feeding:

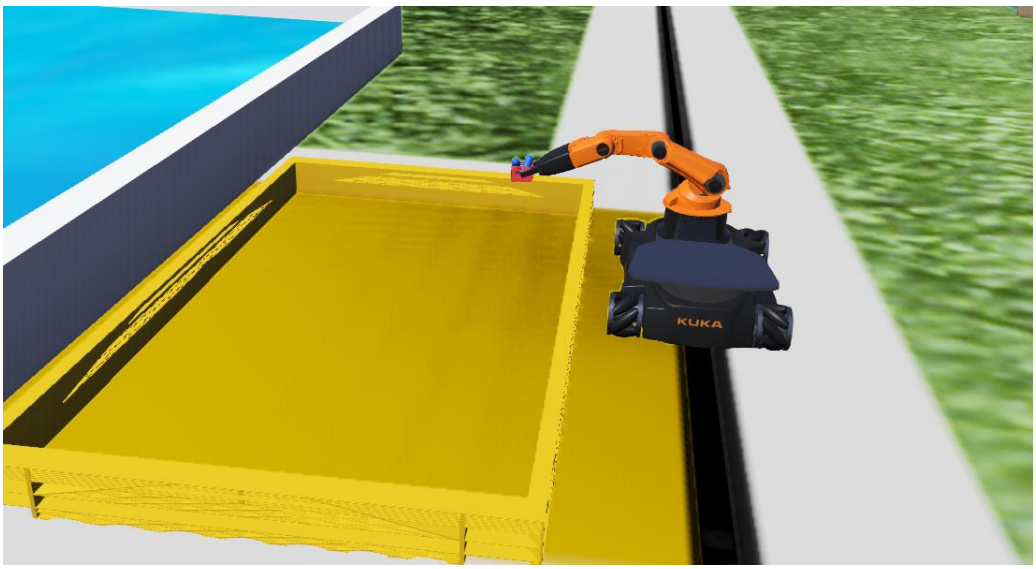


Figure. 97: An example image of color recognition

The above is the color recognition method of double threshold comparison, which ensures that our robot can correctly find orange box and perform task2 actions through two kinds of insurance: color threshold and quantity threshold. This greatly increases the stability and anti-interference of the system, making the robot more intelligent.

2.5.1.2 Image Segmentation-inverse Traversal Method (Ze Sheng)

Image segmentation - inverse traversal method is a very important algorithm for color recognition. In a nutshell, it is to segment the image to speed up the simulation efficiency and speed, and to do so also increases the stability of the algorithm.

Because the simulation software needs to do a lot of calculations, especially in the case of the current compiler processing image (matrix) slowly, we need to make some changes to make it more efficient!

Below is a picture taken by this chapter's camera (both 30 pixels long and wide, for a total of 900 pixels):

Figure. 98: The view of camera2

From the above chapter, we know that to calculate the proportion of the yellow part, we need to traverse the R, G and B components of the 900 points in the whole picture and compare them with the threshold value. However, our camera has a sampling period of 32ms, which means that the simulation will have at least 506,250 operations every second. The analytical formula of n is as follows:

$$n = \frac{1}{0.032}(\text{images}) \times 900(\text{pixels}) \times 3(\text{RGB}) \times 3(\text{yellow\_threshold}) \times 2(\text{logic}) = 506250$$

All of this is too much for the later judgment statements and the compiler, which is not good at dealing with matrices in the first place. And so it is, in this case our

simulation is very, very slow (0.3 to 0.4 times real time).So we need to figure out how to speed it up.

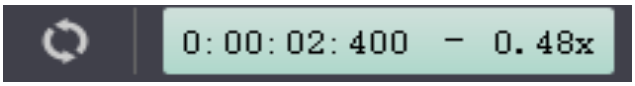


Figure. 99: Low speed caused by heavy calculation

Our idea is to segment the image, that is, to process the image at a high level, and then to improve the simulation efficiency without changing the sampling frequency. From the figure above, we can see that the yellow part only exists in about 40% of the image, so the top part can be completely removed. Based on this idea, we improve the algorithm.

The implementation on the code is shown in the following figure:

Image cutting

Color threshold

Quantity threshold

```
int image_width2 = wb_camera_get_width(ca2);
int image_height2 = ceil(0.45 * wb_camera_get_height(ca2));
int image_height3 = wb_camera_get_height(ca2);

for (int x = 0; x < image_width2; x++)
  for (int y = image_height3; y > image_height2 ; y--) {
    int r = wb_camera_image_get_red(image, image_width2, x, y);
    int g = wb_camera_image_get_green(image, image_width2, x, y);
    int b = wb_camera_image_get_blue(image, image_width2, x, y);

    if (r > 150 && g > 150 && b < 50)
                {
                   base_reset();
                     yellow_pixels++;
                   printf("%d",yellow_pixels);
                }
                else{
                base_backwards();
                }
  }
if (yellow_pixels>220){
base_reset();
automatic_behavior();
}
}
wb_robot_cleanup();

  return 0;
}
```

Figure. 100: General modules of our algorithm

By defining an image height of 0.45 times of height2, the region below the image is obtained by traversing it in reverse order of height. This greatly reduces the area of the excess parts of the image, leaving only the desired effective area, that is, the yellow part, which can not only reduce the interference of the ambient color, but also speed up the calculation efficiency. The simulation efficiency is improved on the basis of ensuring the sampling rate of camera.

Figure. 101: Region of interests

This algorithm combined with the closed-loop operation in the previous optimization can greatly accelerate the simulation speed. Now our simulation speed can reach about 1 times of the real speed under the ordinary speed, indicating that the algorithm design is very successful. The practical meaning is to speed up our whole simulation

Figure. 102: Improved algorithm makes the speed faster

The above are the two core modules of the whole color recognition algorithm, the double threshold comparison method and the detailed description of image segment-inverse traversal calendar. Through these two methods, we successfully realized the anti-interference and perfect operation of Task2. The next part is the quantitative analysis of the optimal region and the anti-interference capability of the algorithm.

2.5.2 Location Planning (Jingyi Ran)

Even if the recognition algorithm is successfully applied and the robot can correctly recognize yellow, we need to make decision where the robot should stop. We choose to compare the number of yellow pixels of different position, and get the graph below.

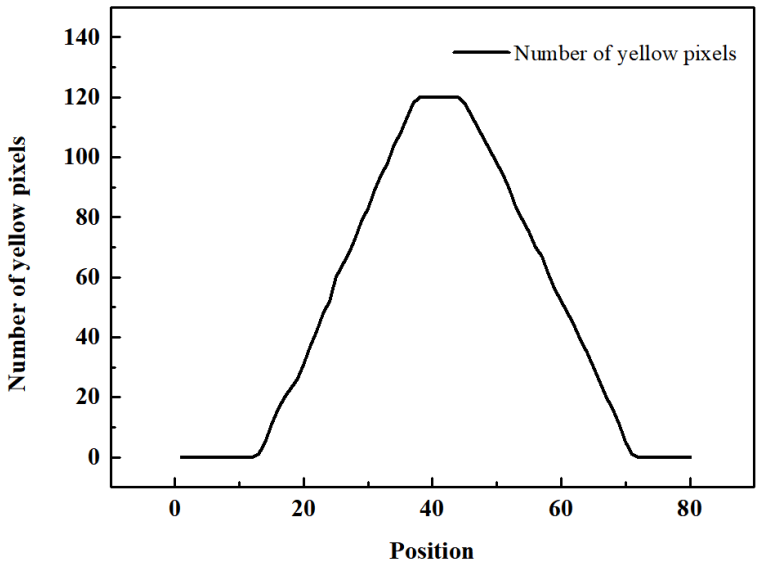


Figure. 103: The number of pixels we detected

We decide to make the robot stop and take action when there is the most number of yellow pixels. From the graph we can see that the place is around the center of the image. Finally, we choose the threshold value to be 90, which is within ± 0.75 of the max value.

From now on, task2 has fundamentally completed. But there are some detects, which means, once the camera capture one yellow pixel, robot will take actions. It is easy to be disturbed. Also, it is inefficient and cause an increase in open loop control since no feedback is applied.

Figure. 104: The position is near the center

2.5.3 Anti-interference Analysis of Color Recognition (Ze Sheng)

Based on the above regional optimization analysis, we propose a method to test the anti-interference performance of the algorithm using the yellow duck as the interference source. We added a yellow duck somewhere in Task1, which not only beautified the environment, but also tested our anti-interference ability of color recognition.

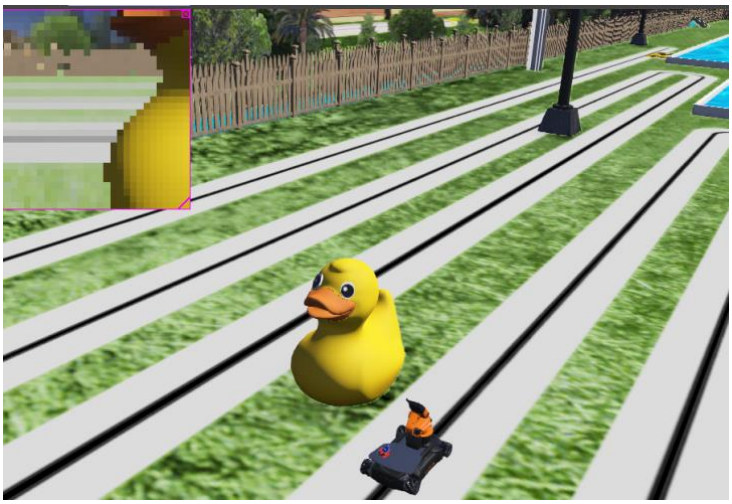


Figure. 105: Yellow duck as a kind of interference

As expected, the robot did not enter Task2 here, but walked by as normal. We can see the reasons for this through the following figure:

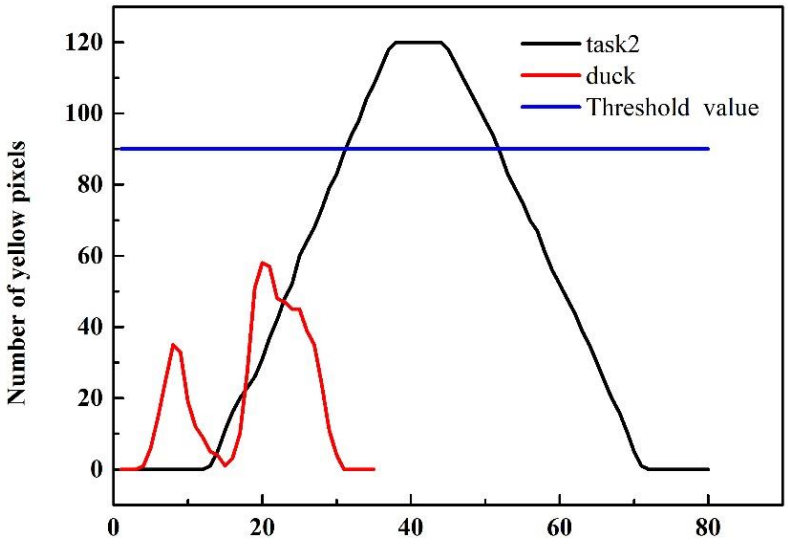


Figure. 106: The pixels of duck and our threshold

The red line represents the number of yellow pixels identified on the duck, and the blue line represents the threshold required to enter Task2.Here we clearly see that

according to the double threshold comparison method and image segmentation method, the number of yellow pixels on the duck body does not exceed the preset threshold, so it cannot enter Task2, and the anti-interference of the system has been verified.

The above conditions show that our color recognition algorithm has certain anti-interference ability, which comes from the double threshold comparison method and image segmentation method. Therefore, the use of these two methods together can greatly improve the stability of the system, reduce misoperation and misidentification.

2.5.4 Color Recognition in Night Mode (Jingyi Ran)

In night mode, we don't use the method of adjusting threshold value, instead, we adjust streetlight brightness and camera exposure to compensate image brightness.

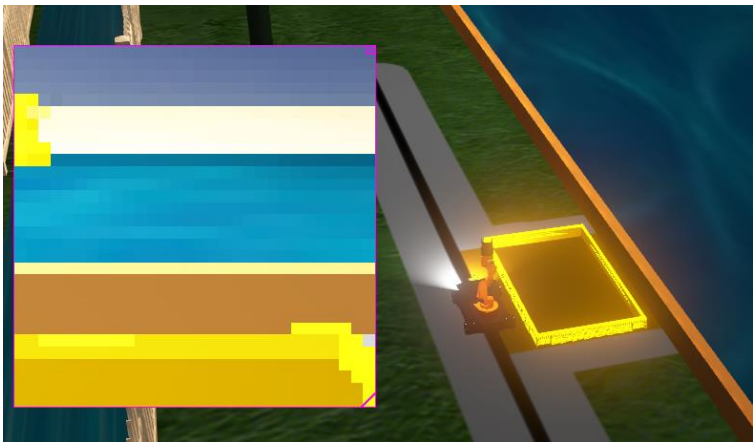


Figure. 107: Light compensation by adjusting exposure

Here, we adjust the exposure from 1 to 3 to compensate for light. We can see that by adjusting hardware we can make robot correctly recognize color in task2.

This project aims to verify the stability of the system by changing equipment and algorithm. In this part, we tend to change equipment to verify the stability and the change of algorithm will be explained in task5.

2.5.5 K-means Clustering (Jiajun Huang)

Technical Content

1.In this part I am responsible for the development of color recognition algorithm. At first, I decided to use the K-means clustering in OpenCV. By the Color Quantization process, we could reduce the number of colors in an image. Thus, the required memory would be reduced. By applying the following codes:

```
import numpy as np
import cv2

img = cv2.imread('home.jpg')
Z = img.reshape((-1,3))

# convert to np.float32
Z = np.float32(Z)

# define criteria, number of clusters(K) and apply kmeans()
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
K = 8
ret,label,center=cv2.kmeans(Z,K,None,criteria,10,cv2.KMEANS_RANDOM_CENTERS)

# Now convert back into uint8, and make original image
center = np.uint8(center)
res = center[label.flatten()]
res2 = res.reshape((img.shape))

cv2.imshow('res2',res2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Figure. 108: K-means clustering in OpenCV

The specified color would be reduced and here is the result when K=8:

Figure. 109: the result when K=8

However, when the task finnaly exposed, I find that the color to recognize is too simple to use this technique. Thus, this method has not been used in our final solution.

2.At first, the me and my partner Jingyi Ran decided to use python for writing codes for color recognition, because there are a huge amount of functions and method already available in the Python environment. By using open-cv, I finally successfullly develop the code:

```
for i in range(3):
        for j in range(3):
                position=position+1
                img1=board[((i*x)/3)):((((i+1)*x)/3)),(((j*y)/3)):((((j+1)*y)/3)),:]
                gray=cv2.cvtColor(img1,cv2.COLOR_BGR2GRAY)
                ret,thresh = cv2.threshold(gray,127,255,1)
                contours,h = cv2.findContours(thresh,1,2)
                cnt=contours[0]
                (p,q,s)=img1.shape
                (B,G,R)=img1[p/2,q/2]
                if (B<10 and G<10 and R>200):
                        colour="red"
                if (B<10 and R<10 and G>200):
                        colour="green"
                if (G<10 and R<10 and B>200):
                        colour="blue"
                if (B<10 and R>200 and G>200):
                        colour="yellow"

                board_objects=  board_objects + [(position,colour,board_shape[(position-1)])]
                board_area = board_area + [(cv2.contourArea(cnt),cv2.arcLength(cnt,True),colour)]
position=0
#print board_objects


##container objects
(x,y,z)=container.shape
for i in range(4):
        for j in range(4):
                position=position+1
                img1=container[((i*x)/4)):((((i+1)*x)/4)),(((j*y)/4)):((((j+1)*y)/4)),:]
                gray=cv2.cvtColor(img1,cv2.COLOR_BGR2GRAY)
                ret,thresh = cv2.threshold(gray,127,255,1)
                contours,h = cv2.findContours(thresh,1,2)
                cnt=contours[0]
                (p,q,s)=img1.shape
                (B,G,R)=img1[p/2,q/2]
                if (B<10 and G<10 and R>200):
                        colour='red'
                if (B<10 and R<10 and G>200):
                        colour="green"
                if (G<10 and R<10 and B>200):
                        colour="blue"
                if (B<10 and R>200 and G>200):
                        colour="yellow"
                if (B<10 and R<10 and G<10):
                        colour="none"

                container_objects=  container_objects + [(position,colour,shape)]
                container_area = container_area + [(cv2.contourArea(cnt),cv2.arcLength(cnt,True),colour)]

#print container_objects
#print container_area
#print board_area
position=0
```

Figure. 110: part of the codes for recognition

By my program, the colors red, green, blue as well as yellow could be detected with a high accuracy. However, then we find that the youbot has its own recognition method. Therefore, this code has not been used in the final project, but the idea in this code has helped us successfully to develop the final recognition code which is to compare the pixels in ROI with the threshold value:

```
for (int x = 0; x < image_width_2; x++)
    for (int y = oringinal_height; y > image_height_2; y--) {
        int r = wb_camera_image_get_red(image, image_width_2, x, y);
        int g = wb_camera_image_get_green(image, image_width_2, x, y);
        int b = wb_camera_image_get_blue(image, image_width_2, x, y);
        if (r > 100 && g > 100 && b < 10) yellow_pixels++;
    }
```

Figure. 111: the compare process

2.6 Distance Sensor and Its Application (Task 3)

2.6.1 Integration and Usage of Distance Sensor (Ze Sheng)

In this project, we do not use the existing distance sensor (the distance is not ideal enough), we use the Distance sensor Node and write the lookup table by ourselves. Note that the distance sensor should also be installed on the Bodyslot node or it will not be enabled. We used the most basic laser sensor with a range of 0.1-2m, and its Look up Table is shown in the following figure:

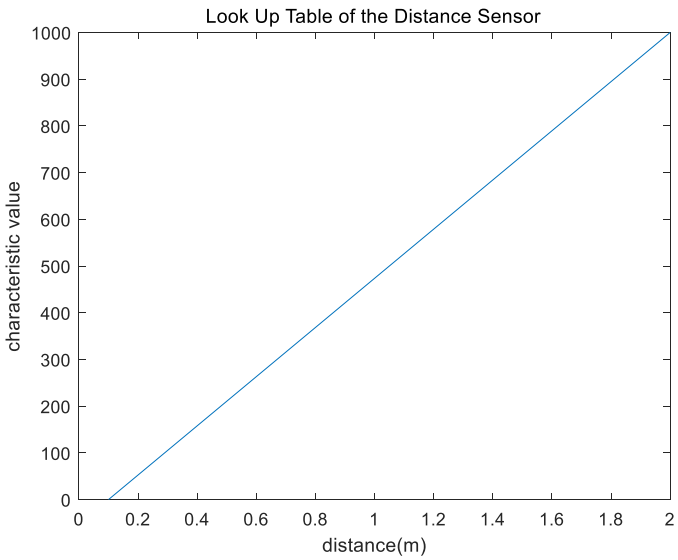


Figure. 112: The lookup table of distance sensor

This is a linear, noiseless lookup table, and you can see that as the distance increases, the eigenvalues returned by the distance sensor increase. This is not the case

in reality (the larger the distance, the smaller the eigenvalue), but here we use a positive correlation linear lookup table for the convenience of understanding the code.

After the lookup table is set, we still need to use the algorithm to convert the eigenvalue and distance. Since there is a linear relationship here, no fitting or denoising means are needed. We can obtain the relationship between the eigenvalue and distance as follows:

$$CV=526.32d-52.63$$

According to the above equation, we can calculate the eigenvalue according to the required distance, and also calculate the distance between the current object and the robot according to the returned eigenvalue. This feature will be applied in the obstacle avoidance section of Task3.

2.6.2 Evading the Forest (Ze Sheng)

After successfully crossing the bridge, we need to avoid the woods, here we use the distance sensor module. The installation and use of sensors has been described in detail in the previous section. This chapter mainly describes the principle and algorithm of obstacle avoidance.

Because the distance between the robot and the ground may be too close when it gets off the bridge, in order to avoid mis-operation, we choose to turn on the distance sensor 0.5 seconds after getting off the bridge. The maximum recognition distance from the sensor here is 2m, but 2m is a little too far for the obstacle, so we decide to avoid obstacles when the object is 1.05m away from the robot. According to the distance sensor lookup table, we can calculate that when the distance is 1.05m, the eigenvalue returned by the sensor is:

$$CV=526.32d-52.63=526.32\times1.05-52.63=500.006$$

Here, we can directly judge whether the eigenvalue or the actual distance calculated by the eigenvalue meets the requirements. In this project, we used specific distance and if loop. The diagram below is the system flow chart:

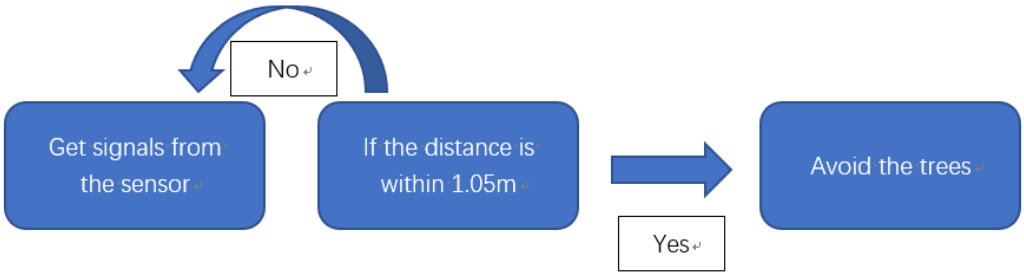


Figure. 113: Closed-loop and open-loop control

As for the obstacle avoidance action, we adopt the open-loop action design because it is the obstacle avoidance in the specified direction. Just like in real life, our robot moves back a certain distance before turning to avoid collisions with trees.

To sum up, we adopted the closed-loop and open-loop joint control method to avoid the forest. In the closed-loop part, we used the distance sensor based on the self-designed lookup table. The open loop is a self-designed, realistic obstacle avoidance operation that prevents the robot from colliding with trees as it turns.

2.7 Object Recognition and Route Control Algorithm (Task 3&4)

2.7.1 Camera Installing and Enablement (Weihao Xiong)

A generic camera node can be used for most of the application and many of the characteristics of the camera can be adjusted like focus and size. the resolution of all the internal rendering devices is determined by the cameraWidth and cameraHeight fields. The enablePhysics field specifies whether the sensor should be affected by physics (mass = 1.5 [kg]) or not.

As for enabling the camera, we can just use the wb_camera_enable function . While using wb_camera_disable function can turn the camera off.

In the whole project, we need five cameras in total. Two are used to find the bridge and the arch individually, the rest three cameras are used for following lanes, color recognition and showing the front view respectively. All cameras are needed to be installed in bodyslot of youbot, otherwise it cannot be invoked. In order to be pleasing to the eyes, we do not set the appearance of the camera, which means these cameras are invisible. The details are shown below.

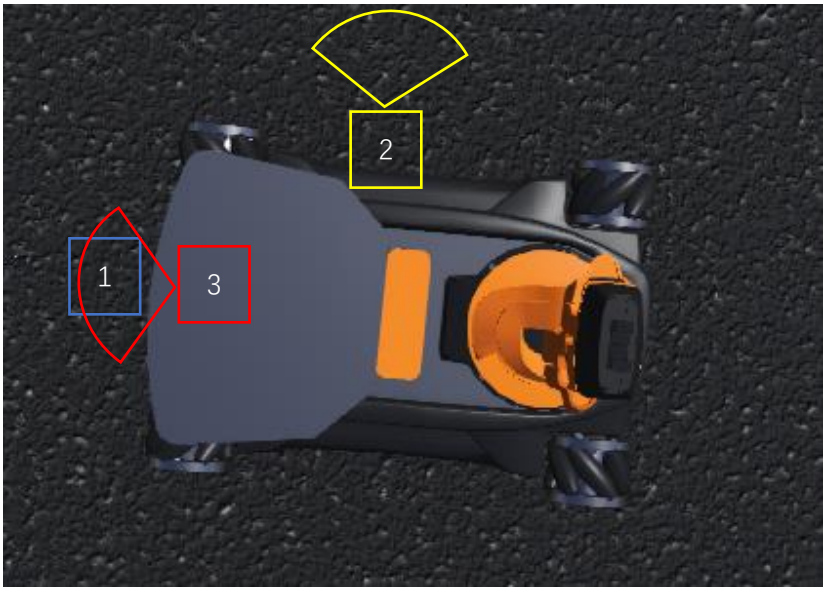


Figure. 114: Locations and view areas of the cameras

It needs to be noticed that all the detail part of the calculation, including how to use these cameras will be introduced in the individual task.

As for the two world cameras, they are supposed to install in the specific locations, due to the interference of the environment. The image below indicates the details.

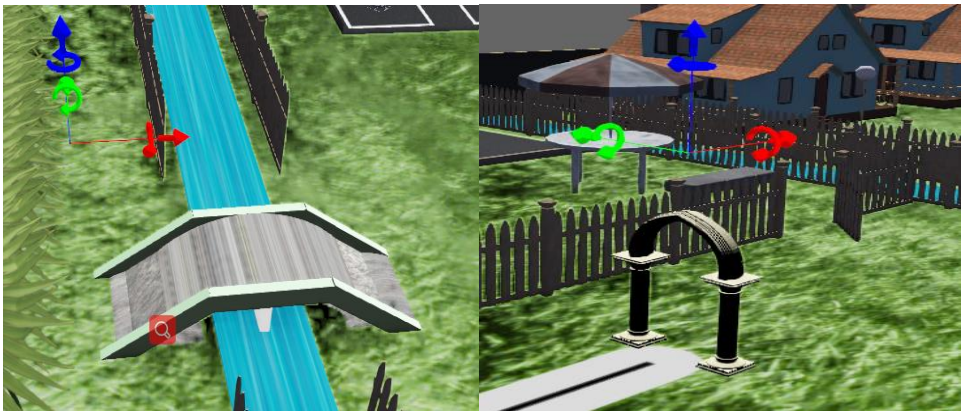


Figure. 115: Locations of the world cameras

In this way, world cameras can find and recognize the id of the bridge and arch. Besides, the world cameras are only opened in the beginning part of the simulation, in case too much consumptions of the calculation.

2.7.2 Camera Recognition (Weihao Xiong)

The camera node is used to model a robot's on-board camera to compute a OpenGL rendered image which is 64×64 . Using wb_camera_get_image can obtain the pixel information, similarly, The red,

green and blue channels (RGB) can be extracted from the resulting image by

wb_camera_image_get_*-like functions.

The recognition node can provide the capability to recognize the solid nodes with recognitionColors. The object size is estimated using the boundingObject . In case the solid nodes and its children do not have any bounding object, the dimension is estimated using the shape .

When the https://cyberbotics.com/doc/reference/camera - wb_camera_has_recognitionrecognition mode is enabled using wb_camera_recognition_enable , rectangles surround the recognized objects. If the mouse cursor is over one of these rectangles and the simulator is paused, then a complete description of the recognized object is displayed in red . A camera recognition object is defined by the following structure:

```
typedef struct {
  int          id;
  double       position[3];
  double       orientation[4];
  double       size[2];
  int          position_on_image[2];
  int          size_on_image[2];
  int          number_of_colors;
  double    *colors;
  char        *model;
}
```

So every solid object has its own special id , so the recognition node can identify the objects respectively. position_on_image and size_on_image can be used to determine the bounding box of the object in the camera image with pixels units . Since each color is represented by 3 doubles (R, G and B) , the number_of_colors and colors returns respectively the number of colors of the objects and pointer to the colors array . The final effect will be shown below.

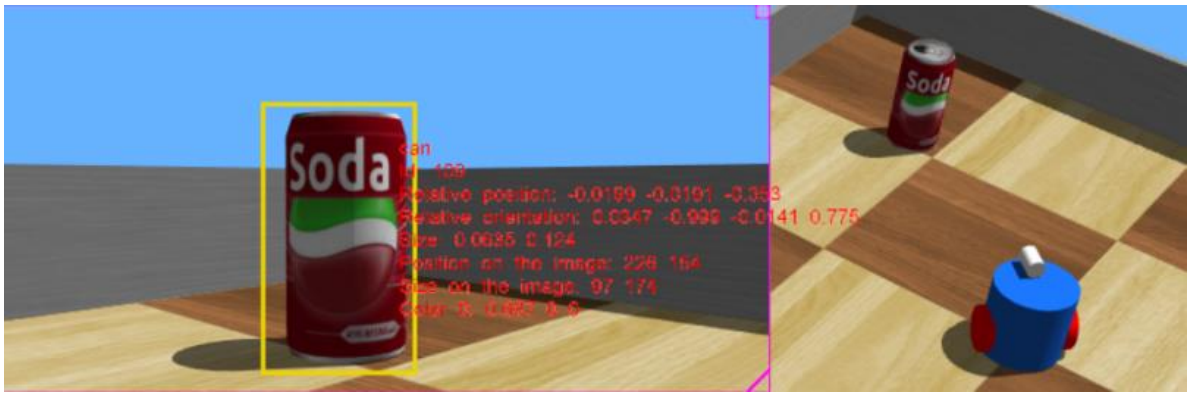
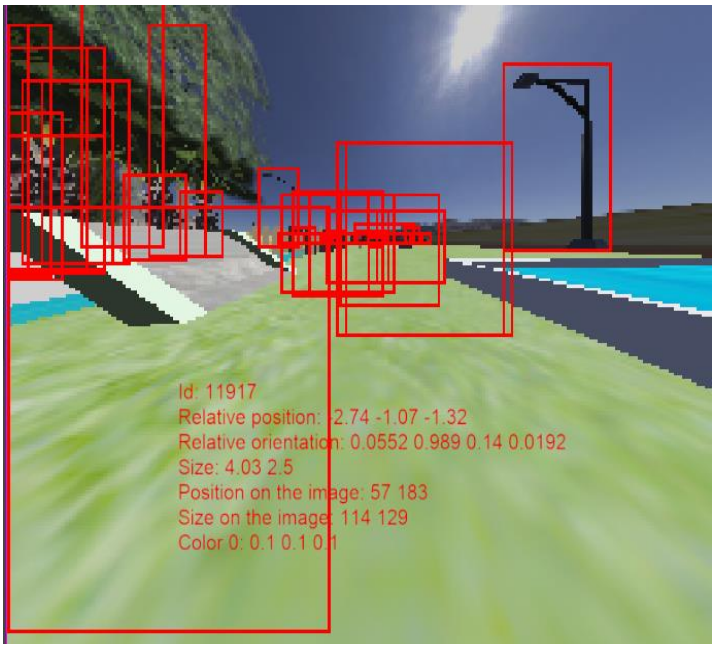


Figure. 116: Camera recognition effect

2.7.3 Extraction of Object Id By World Camera (Jingyi Ran)

By starting using recognition function of camera, we can let it recognize one object and give this object an id, and it can be retrieved by calling the object.id structure function. In this project, there were too many environmental factors. In order to avoid unnecessary occlusion and false identification, we used two world cameras to identify the id of arch bridge and arch. See the sensor installation section for the specific installation steps of the camera.

Every object has its own id. The id of arch bridge and arch is 11917,12220 respectively, since every time open the world the id will change, we choose to use the world camera, not by default. This operation is to increase the stability and intelligence of the system.



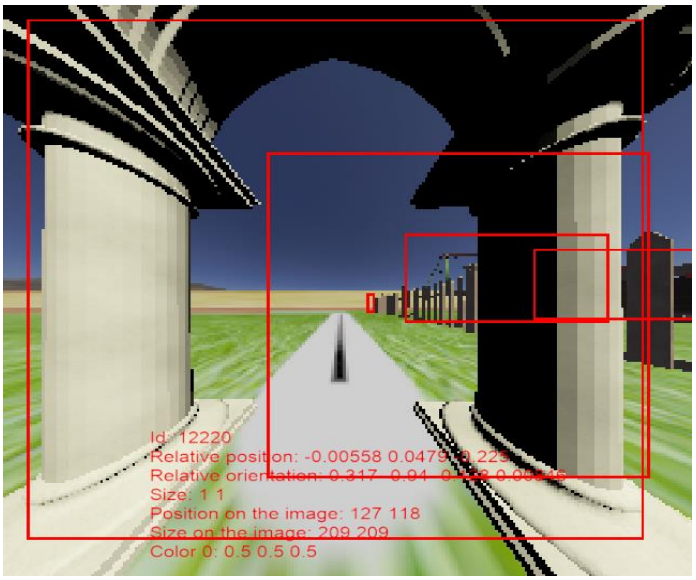


Figure. 117: The id of the bridge and the arch

2.7.4 Route Control Algorithm Based on Object Recognition (Zihao Huang)

As the task 3 and 4 involve the identification and distance measurement, Camera Recognition Object function in camera API is suitable. For a solid object, if recognition colours are set on the surface of it, data of relative distance, orientation and size can be captured by a camera, which can be useful to adjust route.

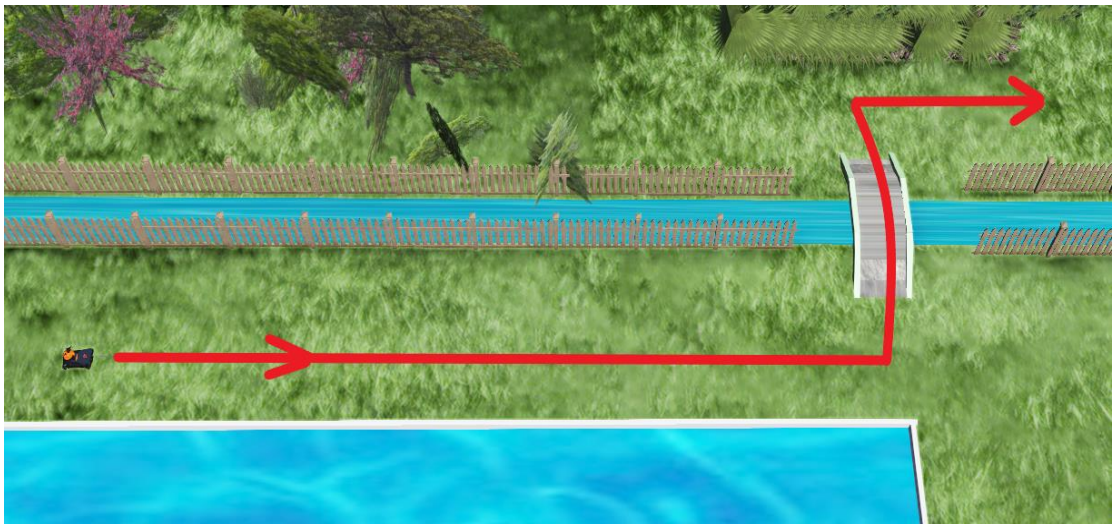


Figure. 118: Requirement route of task 3

Taking the task 3 as an example, the car is required to get close and cross the bridge.

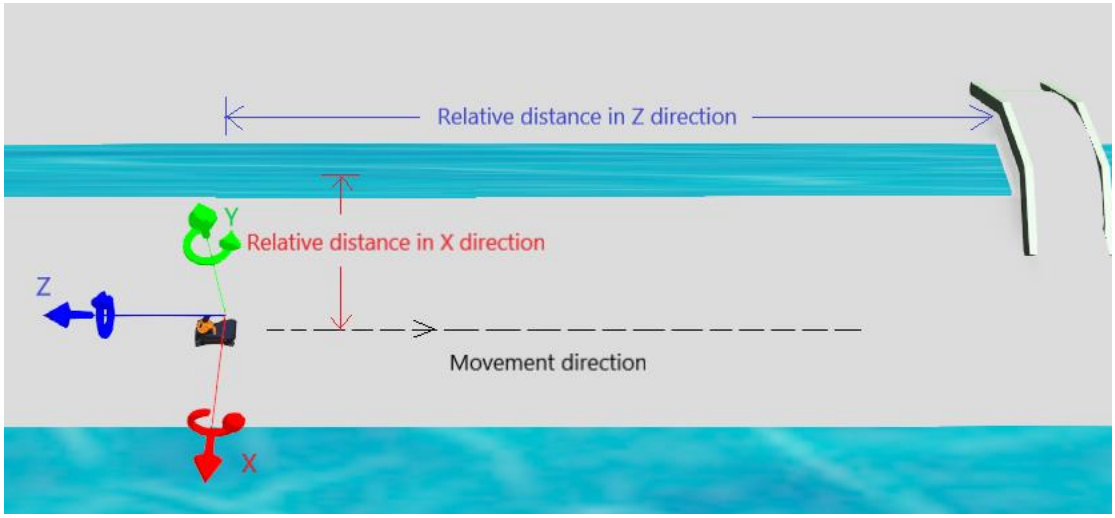


Figure. 119: Route planning based on object recognition

As the ID of the bridge has been obtained, the car can recognize the bridge. At the beginning, it moves in parallel with the river to the right with a default speed. Meanwhile, the relative distance of the bridge in X direction obtained by the camera can be utilized, and the proportional feedback control can be introduced to keep the relative distance in X direction between the car and the bridge unchanged while moving forward. That is, when the direction of the car starts to shift abnormally, the speed of the left and right tires will be automatically adjusted to return to the normal direction. The car also needs to monitor the distance of the bridge in Z direction throughout the whole process. When the Z distance is too close, it will start to turn left and cross the bridge.

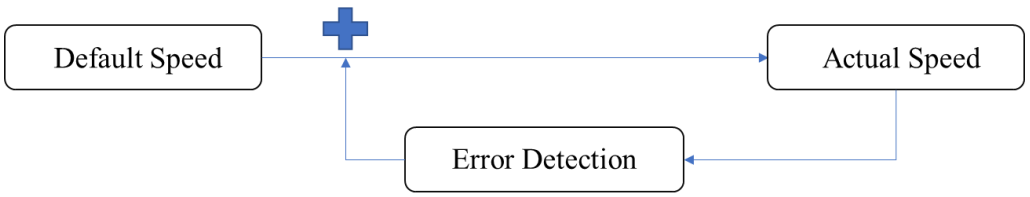


Figure. 120: The proportional feedback system

```
for (int i = 0; i < number_of_objects; ++i) {
  if(objects[i].id == Bridge_ID && !*flag_3){ //Identity the bridge
    error_x=(double)(objects[i].position[0])+2.9;//Error in X direction
    speed_left=-error_x-14;
    speed_right=error_x-14;
    if(speed_left<-14.80) speed_left=-14.8;
    if(speed_right<-14.80) speed_right=-14.8;
    double speeds[4]=
{speed_left,speed_right,speed_left,speed_right};//Adjust speed of tires
    base_set_wheel_speeds_helper(speeds);
    if(objects[i].position[2]>-1.2){
      //Turn and pass the bridge if Z distance is close
      passive_wait(0.3);
      base_backwards();
      passive_wait(3);
      base_turn_right();
      passive_wait(1.8);
      base_set_wheel_speeds_helper(constant_speeds);
      passive_wait(8);
      *flag_3=1;
    }
  }
}
```

Figure. 121: Codes of route control

In terms of codes, the algorithm is run when the task flag is not 0, which means the car has not passed the bridge. The expected distance in X direction between the car and the bridge is 2.9m, so the X error is calculated to adjust velocities of tires. In the logic, if the distance in Z direction is too small, which means the car gets close to the bridge, the car will turn left and cross the bridge and the flag turns 1, so in later process this algorithm will not be executed anymore.

Analysis and Discussion

1. For route control algorithm based on object recognition, it plays an important role in task 3&4. With it, the car can move close and cross the bridge and the arch as required. For this algorithm, I did not introduce a PID feedback system like what Fangze Xu did for tracing line function, because I think the process of straight moving does not involve too much interfere. The simpler proportional feedback system can not only reduce calculation to increase simulation speed, but also meet the precision requirement.

2. Anti-interference test

Setting interference in X direction to 5m, which is much larger than the interference the car might encounter in the project.

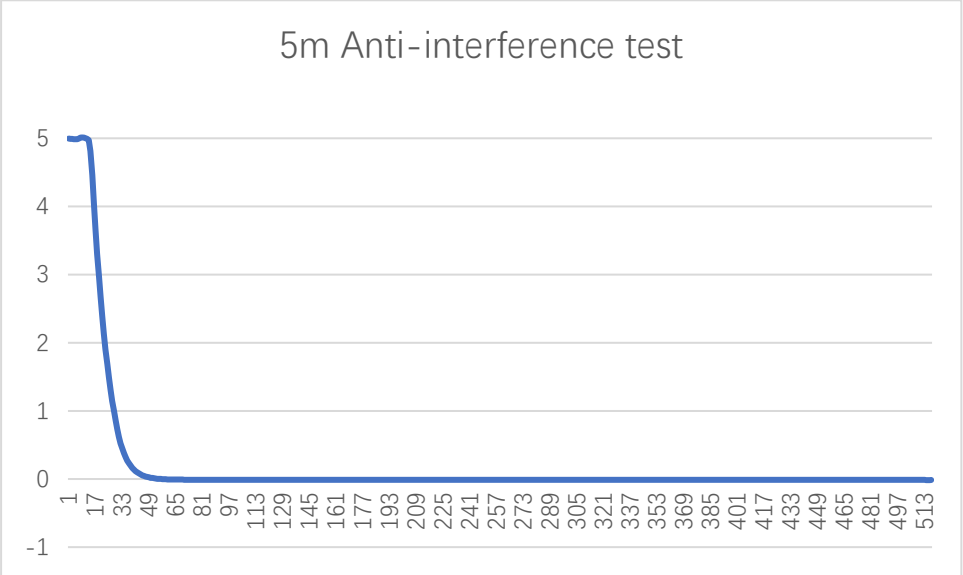


Figure. 122: 5 meters anti-interference test

It can be seen in the diagram that the car quickly returns to the normal distance to the bridge in 33 timesteps. (around 1056ms)

Similarly, setting interference orientation to 45 degrees, which is also larger than it may meet.

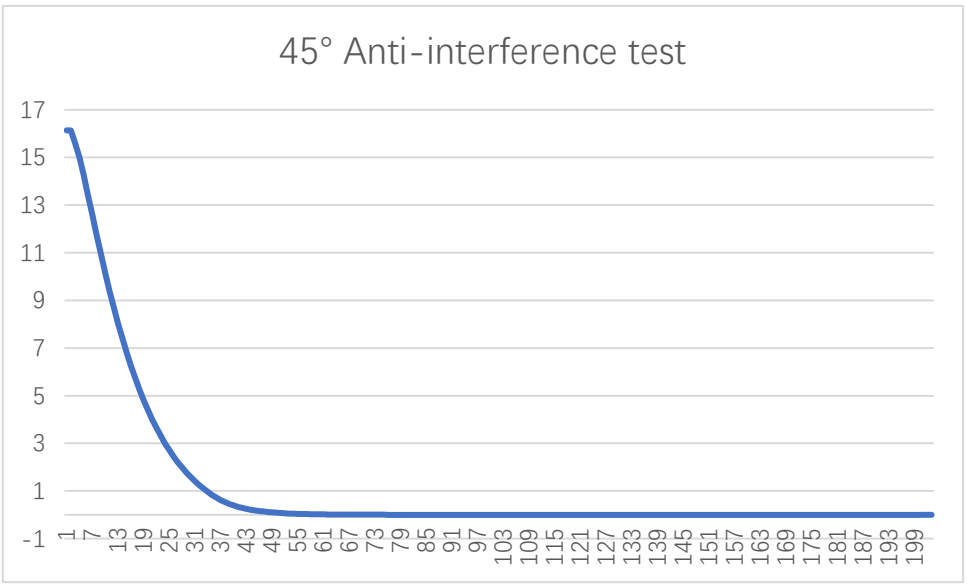


Figure. 123: 45° Anti-interference test

The results are similar with the former test as the X distance error can drop to almost 0 in 49 timesteps. (around 1568ms)

3. In task 3, due to the special property of trees, which cannot be set recognition colour on the surface, my original solution was to utilize a beacon put besides the forest to locate trees, so the car can detect them with the beacon and then avoid crashing them. But Ze Sheng thought a better solution, and he used a distance sensor set on the car to find the trees directly.

2.8 Night Mode Threshold and Basic Logic of Task 5 (Task 5)

2.8.1 The Changes of Color Recognition Threshold in the Night Mode (Ze Sheng)

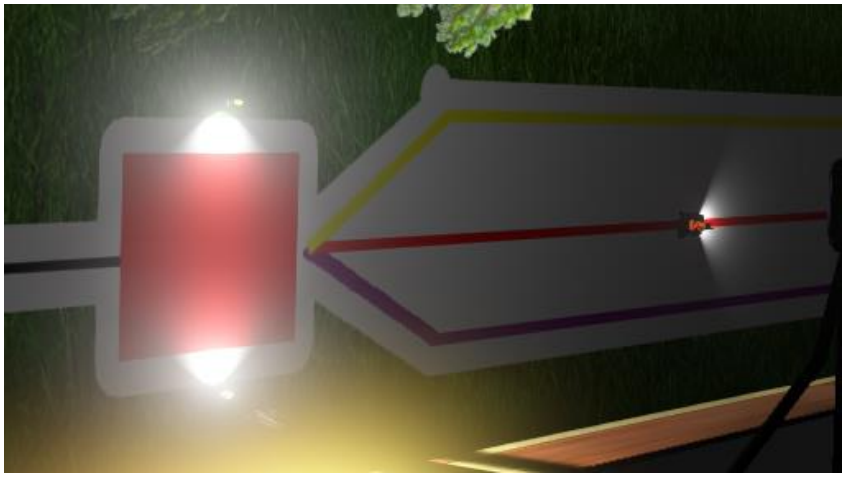


Figure. 124: General looking of task 5 in Night Mode

In night mode, many parameters change, and this chapter discusses the effect of night mode on task5 color recognition in detail.

On the hardware side, there must be enough lighting at the key color blocks to allow the robot to see specific colors. It should be noted that the light here should not be too bright, otherwise it will cause the yellow block cannot be recognized (strong light will cover the yellow and cause the recognition failure).

In terms of algorithm, after adjusting the light intensity, we should set the quantity threshold of color recognition. Since fewer color pixels can be recognized in the dark, the number of pixels that can be recognized will be reduced for the endpoint. However, the red line itself will be taken into account, so the selected threshold should be between the number of red pixels recognized on the red line and the maximum number of red pixels recognized at the end point. A large number of experiments show that the threshold is 90, a sharp decrease in the number of recognizable pixels compared to 2200 during the day.

The above is the change of task5 threshold in night mode. The color threshold can be kept unchanged (or deepened appropriately) by light compensation, while the quantity threshold must be changed, because the number of recognizable pixels will be greatly reduced.

2.8.2 Basic Logic in PID Tracing and Color Recognition in Task 5 (Fangze Xu)

According to different color of line the robot needs to trace, we need to use pointer to convey the value between different (like *mode), we use *mode to represent different color of line we need to trace.

Table 6: Relationship between mode and color

*mode	color
0	black
1	Red
2	Yellow
3	purple

Based on this table, we can change *mode in the color recognition in task5 like:

if(pixcountR>1000){//threshold value

*mode=1;// corresponding mode to the PID tracing(black line-tracing, blue line-tracing, purple line-tracing or green line-tracing)

2.9 Main Loop Structure (Zihao Huang)

Technical Content

1. Responsible for integrating subsystems and debugging for integration.
2. Built a new main function frame to assign functions into 5 parts corresponding each task to improve readability.
3. Released five new versions:

   i. Deleted unnecessary variables in PID tracing function to optimize structure. (2020.4.30)

   ii. Integrated the first four tasks, and modified codes to reduce lag and improve fluency. (2020.5.16)

   iii. Optimized the structure in the main loop to increase the readability so that each case corresponds to task 1 to 5. (2020.5.17)

   iv. Simplified the codes in PID tracing function to improve speed and avoid repeated operation. (2020.5.23)

   v. Modified the codes to adapt new environment and optimized the codes of the route control algorithm based on object recognition in task 3&4 to improve movement speed. (2020.6.4)

2.9.1 Main Loop Explanation

```
while (wb_robot_step(TIME_STEP) != -1) {
  passive_wait(0.01);
  switch(current_task){
    case 1:
      if(Task_2_Detect(ca[1]) && current_task==1) current_task=2;
      else PIDtrace(ca[0],a,b,0.9,0.1,0.7,mode);
      break;
    case 2:
      current_task=Task_2_Behavior();
      if(current_task==3) wb_camera_recognition_enable(ca[2], TIME_STEP);
      break;
    case 3:
      current_task=Task_3(ca[2],flag_3,Bridge_ID);
      break;
    case 4:
      if(!task_4_flag) current_task=Task_4_Arch(ca[2],flag_4,Arch_ID);
      else{
        PIDtrace(ca[0],a,b,0.9,0.1,0.7,mode);
        current_task=Task_5_Detect_Color(ca[0],mode);
        if(current_task==5) wb_camera_recognition_disable(ca[2]);
      }
      break;
    case 5:
      PIDtrace(ca[0],a,b,3.0/8.0,0.1,0.2,mode);
      current_task=Task_5_Detect_Stop(ca[0],current_task);
      break;
    default:
      passive_wait(0.8);
      base_reset();
      break;
  }
}
```

Figure. 125: Main loop codes

To increase the readability of the codes, the variable current task is defined. It is refreshed each time step, which determines what the car is going to do. Its value exactly corresponds to five tasks, and the structure is shown in above figure.

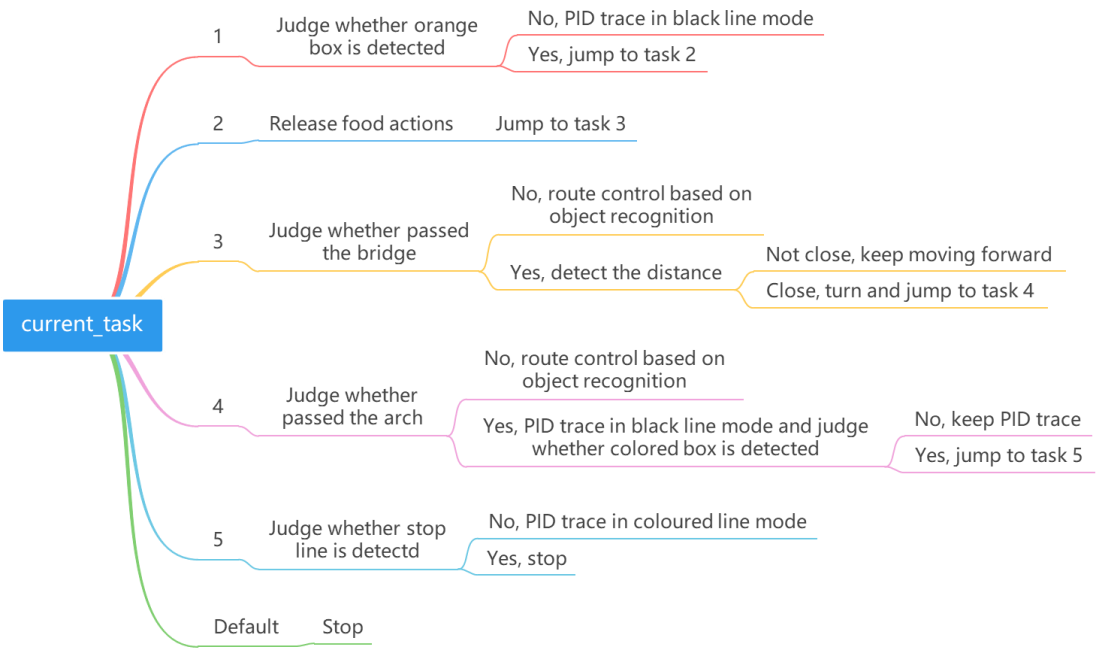


Figure. 126: Main loop structure

At first, the initial value of current task is 1, so the car follows the black line. Meanwhile, it judges whether the orange box is detected, if so, the car will do task 2 and release fish food. After that, current task value turns to 3, and the car starts to move along the route based on what it has recognized in camera before passing the bridge. Then it will keep moving and turn right when the distance sensor warns the trees are close. In task 4, the first step is like in task 3, and next PID tracing function is used again to follow the line. If the color box is detected, the car will start task 5 and still follow the line until reaching the destination. With this logical structure, the car can finish all the tasks smoothly.

Analysis and Discussion

For the logical structure in the main loop, my scheme replaced the earlier version, which contains 10 cases. The new codes have better readability and simpler structure, which helps improve simulation speed.

2.10 Project Management

2.10.1 “What Is Our Project” (Yukai Song)

Technical Content

I illustrate “what is our project” to our team members and find out that it composes of building the patio and task 1 to 5, each with different requirements. We found out

that building the patio, following the lines, color recognition, object detection and control are main parts of our project. After understanding about our projects clearly, the distribution of work and time management become clearer.

Analysis and Discussion

1. The following diagram shows how I divide this project and find out what we will meet in the future:

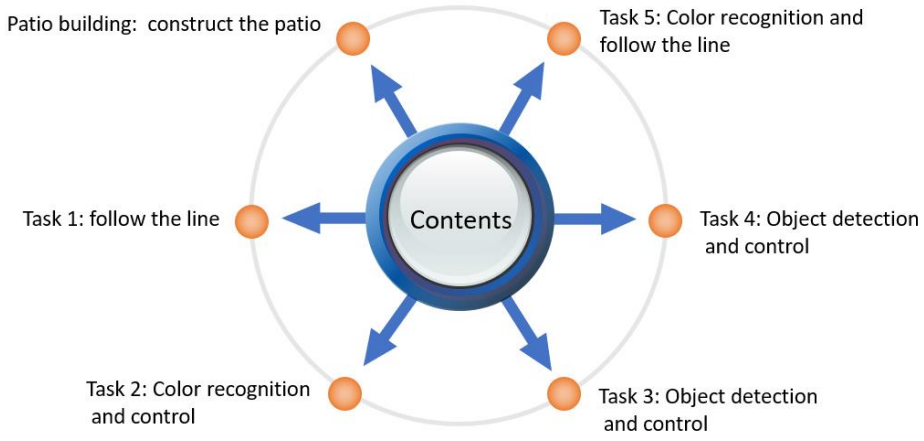


Figure. 127: Contents of the project

The result shows that building the patio, following the lines, color recognition, object detection and control are main components of our project. This result helps me guide our team in a clearer way.

2.10.2 "How To Do The Project" (Yukai Song)

Technical Content

The most important part is how I organize the whole team to do the project. Firstly, we followed the sequence of project planning, initial design, patio building and task 1 to 5 and modification as well as further work in this project to tackle the difficulties of this project in a logical way. Teamwork is one of the most important parts of our project. Good teamwork can ensure the success of this project. We mainly use two techniques to do our teamwork. The first is breaking a huge task into small parts and divide those small parts to our team members. The second way is when facing a difficult problem, we will brainstorm about it to come up with a good solution. As there is no financial pressure in this project, the role of time management becomes very important. By using

the methods of project planning, communication, weekly meeting and feedback, we successfully finish our project in time. In the project planning part, we developed the Workload Distribution, Work Breakdown Structure, Project Network Diagram and Gantt Chart to help us manage the project. Communication with teachers and teammates through teams or other social media cost me most of the time in the time management. From those communications, I know where my teammates have problems and can assign others to help him or her and become clearer about project requirements through communication with teachers. Weekly meeting and feedback are organized every Sunday evening 19:30 pm to get a summary about what we achieve in the last week and what we need to do in the next week. Feedback from teammates enable me to get the condition of their work and check with project plan, I can organize the next week tasks in a better way than I previously planned.

Analysis and Discussion

The following flow diagram shows how I organize the whole project:

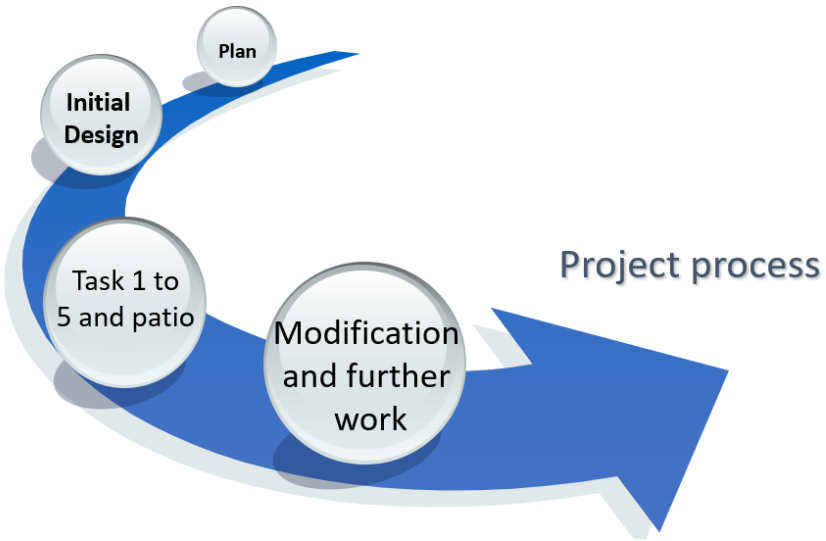


Figure. 128: Process of the project

This process is given through the discussion with whole team and we follow this sequence and successfully finish our project. Two team management skills can be demonstrated through the following picture:

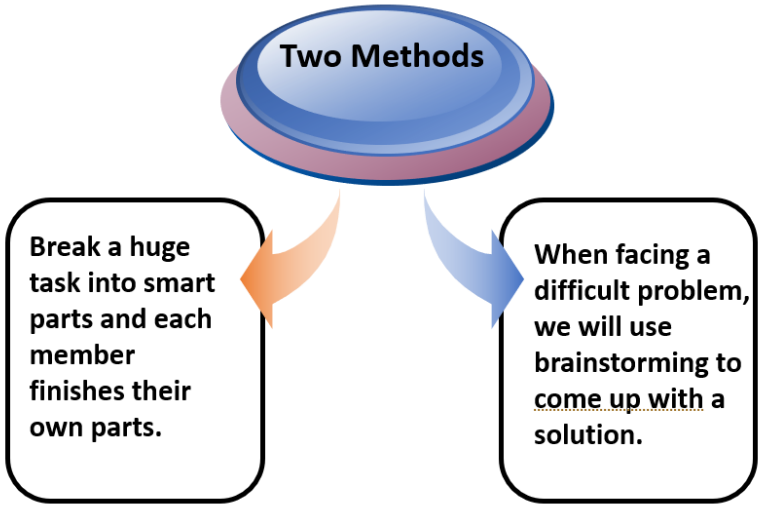


Figure. 129: Two team management methods

We develop those methods through our work and gradually improve it through our practice. Finally, it is time management, the combination of four time management steps help us check our progress and finish the project in time.

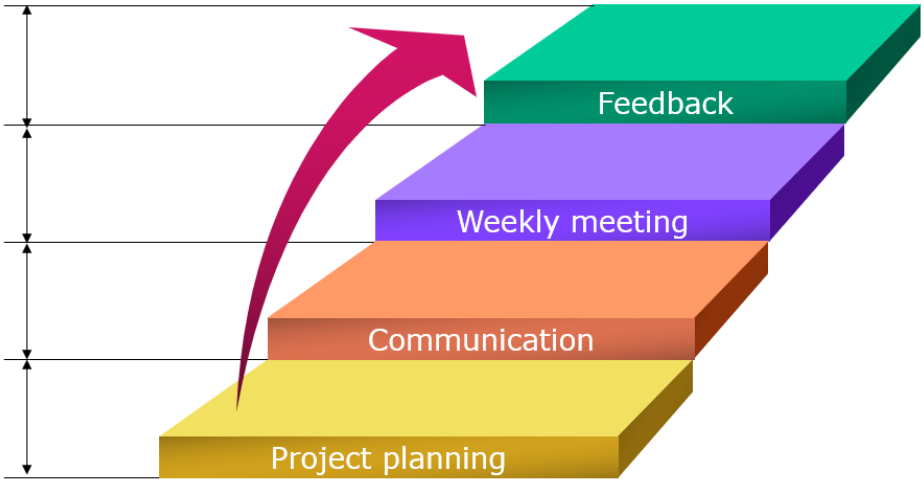


Figure. 130: Steps of time management

2.10.3 Project Planning (Jingyi Ran, Dingte Yang, Zhiheng Lin)

Technical Content (Jingyi Ran)

In this project, we apply what we have learned in Engineering Project Management and Finance and carried out project management plan including state of work, work breakdown structure, project network diagram and Gantt chart. I am in charge of this part.

I mainly do some work in project network diagram as well as Gantt chart.

	Task	Start	Duration
1	Task allocation&pre-thinking discussion	2/26/2020 02:20 PM	18d2h40m
2	Reassign tasks&learning webots	3/15/2020 11:40 PM	8d2h20m
3	Environment building	3/24/2020 02:00 AM	10d21h45m
4	Task1: robot's path following	4/1/2020 04:20 AM	14d18h40m
5	Task2: robot's color recognition & food releasing	4/9/2020 12:00 AM	17d0h35m
6	Task3:robot's bridge crossing & object detection	4/20/2020 11:55 PM	16d0h5m
7	Task4: robot's arch detection & line following	5/3/2020 11:20 PM	19d0h45m
8	Task5:robot's color box detection & correct path following	5/9/2020 10:40 PM	14d20h20m
9	Testing	5/24/2020 11:50 PM	2d1h5m
10	Code optimazation	5/27/2020 03:40 AM	6d10h55m
11	Environment optimazation	5/27/2020 08:40 AM	6d15h5m
12	summary & report & video	6/3/2020 12:55 AM	11d21h30m

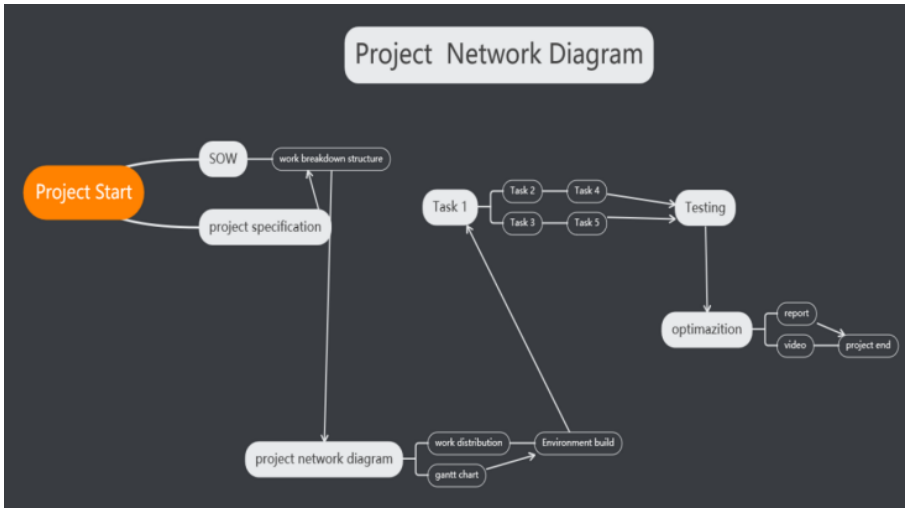


Figure. 131: Project management plan

Team Notebook summary & collection

I am in charge of coordinating the notebook and doing some recording to make the notebook tidy and easy to read.

Technical Content (Dingte Yang)

1. I made the project work distribution diagram which illustrates what will every team member do in the project. The project network diagram helps us to figure out every steps clearly which gives definite goals to achieve during the project. It contributes figuring out what was done and what need to be done next, making the project easy to follow.

2. I made the statement of works which contains introduction, purpose, scope of work, schedule, milestone as well as deliverables. This document gives a general

direction of the project and helped in clarify management methods.

3. I helped Yukai Song in weekly meeting every Sunday evening 19:30 pm. Sometimes I listened team members' feedback and ideas and helped noting them down as weekly meeting summary.

Technical Content (Zhiheng Lin)

1. Work with the team to do the project planning and work out the workload distribution.

2.11 Others

2.11.1 Preparation Work (Yukai Song)

Technical Content

1. Before the simulation software is decided, I use python opencv to fulfill the edge detection and blue color recognition function. The basic idea is to utilize python opencv library's canny edge detecting functions to achieve the edge detection and hsv threshold function to achieve the blue color recognition.

2. Find a useful youtube Webots tutorial video to help teammates learn how to use Webots.

3. Solve the VPN problem of the team by providing the VPN of Glasgow to our team members.

4. Cooperate with Zhiheng Ling to design the script for simulation and organize their training, which shows a great result during the simulation.

Analysis and Discussion

1. During the process of waiting for the news about simulating software, I tested python opencv function and did the experiment with myself and the result shows that it can work. The canny edge detection method provides an idea for our latter line optimization developed by Ze Sheng and the result of edge detection and blue color recognition myself is shown below.

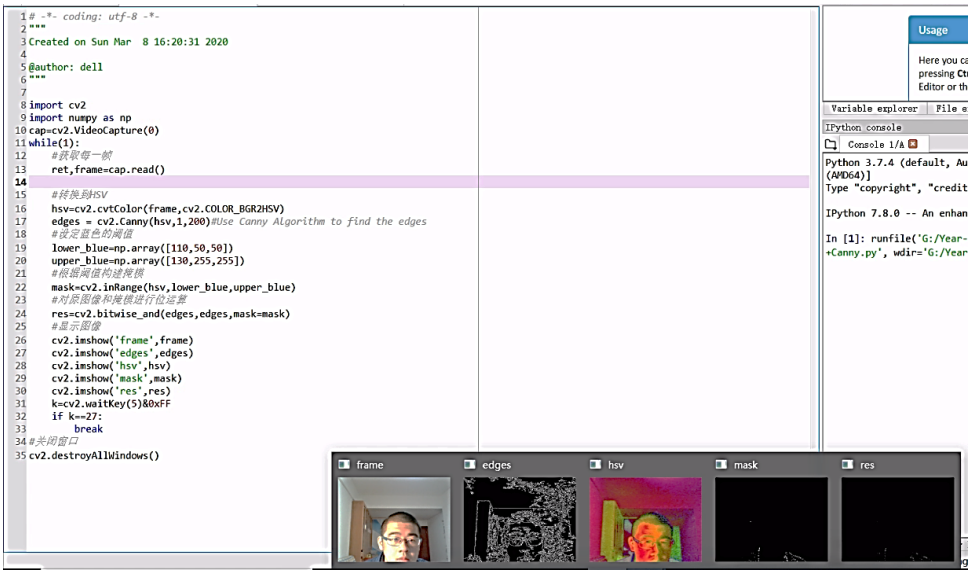


Figure. 132: Edge detection through python opencv

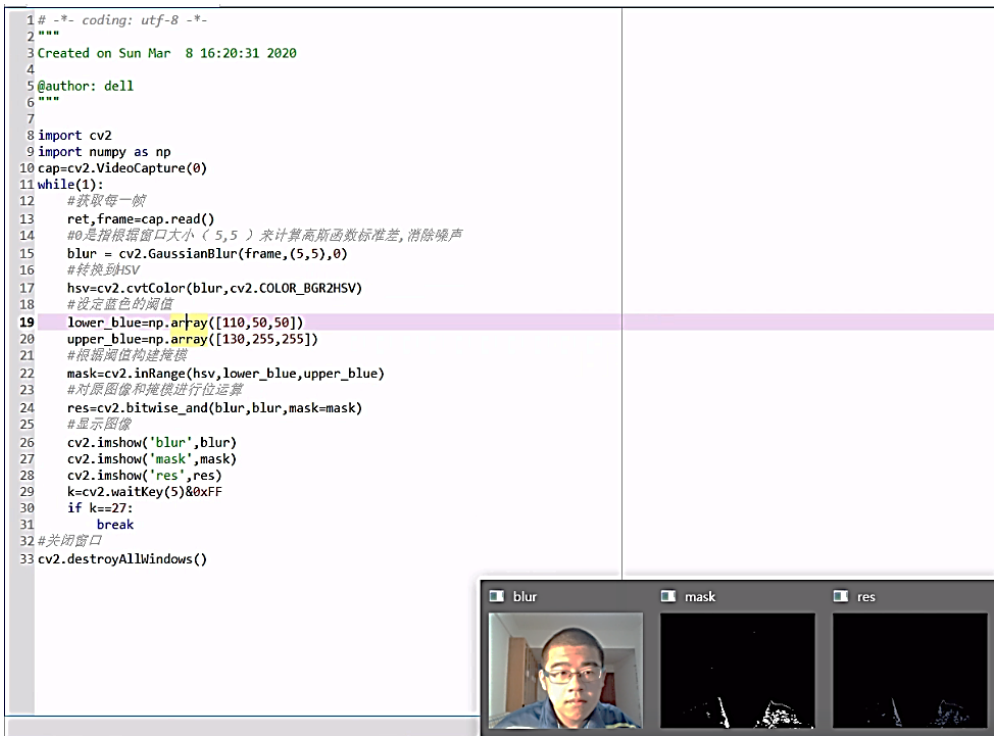


Figure. 133: Blue color recognition through python opencv

2. The youtube video tutorial provides a very great supplement about how to use Webots besides the official tutorial. Many team members gained a better understanding about how to use Webots after watching that tutorial.

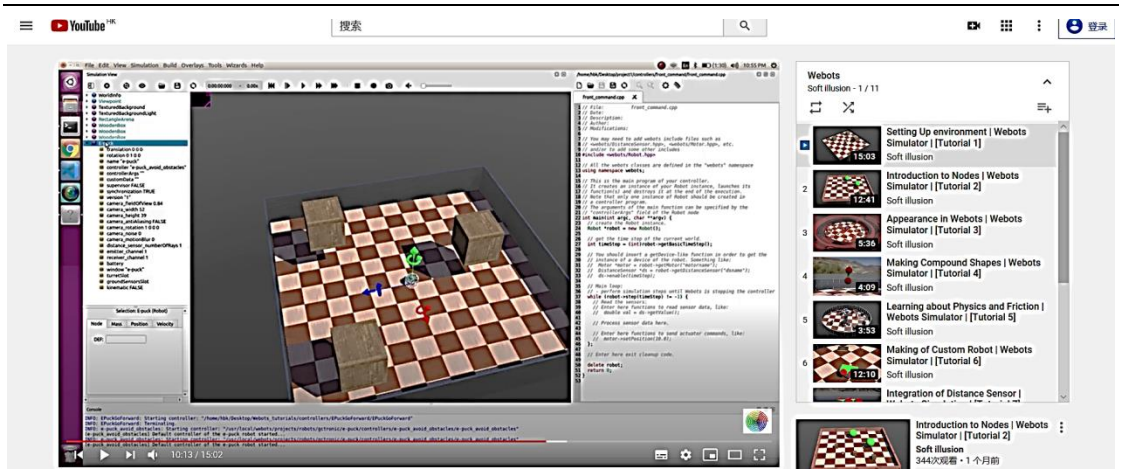


Figure. 134: Useful youtube tutorial

3. After finding the Glasgow VPN, searching the data and other useful material about Webots is no longer a question for us. This VPN also enables us to watch the youtube tutorial I mentioned above easily.

4. The scripts design and training help us behave better in the final presentation and really impress the professor.

2.11.2 Script Witing (Zhiheng Lin)

Technical Content

1.Help Ze Sheng complete the design of the demo guide words with Yukai Song.

The following is our short play. Ze Sheng is the director and photographer, Yukai Song and Zhiheng Lin are the scriptwriters, Ze Sheng and Jingyi Ran are the actors, playing the roles of guide and tourist respectively. This short play was staged in the simulation on June 12 and achieved good results. Here's our script: (B means boy, G means girl)

B: Hello everyone, welcome to Team 30 “A certain UESTC railgun” laboratory, I am your tour guide, today I will lead you guys to enjoy a wonderful high-tech journey with our latest robot youbot 7. Are you ready?

G: Yes, I have heard that your robot is mutifunctional, who can walk by specific line tracing and recognize specific color, is that right?

B: Yes, madam, but you underestimate its function, it can also recognize and avoid obstable, take box and throw it to the pond as well as go through arch brige. Oh, it can also work in the night mode, I bet you haven’t seen that before.

G : Oh, yes, sounds amazing, I cannot wait to start the journey.

B: Ok, let’s start from the Youbots parking lot. The first stop of the journey is line

tracing, robot will use its robot arm to get the super capsules from the table

G : Why does he get the food box

B: No hurry, he will give you a fantastic show in the next stop. now, let's follow line program with PID control and enable the camera. Look, he is moving along the lines.

G: wow, hou can he does this?

B: He always keeps lines in the center of image that he gets from the camera.

G: Be careful, the first turn is coming!

B: Don't worry. He will alwayst keep the lines stay at the center of image through adjusting the error and keep the speed not exceed the maximum range, it is not such a huge problem.

G: Great, Oh, there is a cute orange duck, will that disturb the yellow color recognition in the task 2?

B: Don't worry. We have already installed an intelligent ROI color recognition algoritm. See, Youbot 7 does not enter into the task 2 mode when he sees the orange duck.

B; the remain part of task 1 will be simulated in faster mode. Here you can see the advertising board and our team logo. Really cool isn't it?

G: It looks so nice!

B: Ok, everyone, we have finished the first journey, let's start the second journey, the food box will work. Youbot 7 will detect the orange box and then take the food and release it into the box.

G: Oh, amazing. I think the water, pond as well as the fountain are the best I have ever seen.

B: glad to here that

B: Ok, we will turn to the third stop, Can you guess where the robot will go?

G : I saw a beautiful garden, Will the robot go to the house and have a nap?

B : No, out robot is diligent, he will go across the bridge. This time, no black line will guide him anymore, and he will enable the object detection ability. He can get the relative position which can help him pass the bridge.

B: now he is passing the bridge, and avoid the trees. You can see the reflection area of our distance sensor

B: Now, let's go to the fourth stop, the arch is found by the robot and the position data is being processed like task 3.

G: The turns of task 4 are all right angles and that must cause problems to your mid-value Algorithm.

B: Yes, but in this task, we use another line-tracing algorithm which is developed by our team members with a lot of effort. Look it can tackle the right angle turns successfully!

G: That's amazing.

B: Now, do you see the color box? The last task is approaching.

G: Yes. He will recognize the color in the box and then follow the according line, right?

B: Yes, the color of the box is yellow, look, he follows the yellow line. Thanks for having a journey with me and youbots 7, hope to meet you next time!

G: Amazing. But I have heard that he can work in the night mode, can you show us?

B: Of course, look, now, it's the night mode…

It's worth noting that our lines are not designed for fun. The main purpose of this line is to highlight some of our environmental optimization and algorithm advantages, including the anti-jamming of color recognition and the stability of the line patrol algorithm in the right-angle part. Through this short play, we show the performance of the algorithm and the optimization of the environment in an interesting way.

Analysis and Discussion

In this project, in my participation in these task lists, I learned how to use webot and had a more comprehensive understanding of how to build a complete robot. In cooperation with team members, I gained a new understanding of the use of the C language and increase my interest in the robot production

2.11.3 Safety Issues (Jiajun Huang)

Technical Content

1.As we know, Youbot has its max speed at 0.8 meters per second. However, some groups set the velocity at unsafe values which exceed the max velocity. For our group, me and my partners carefully set the speed at 0.7m/s which is fast enough and safe.

General data youBot omni-directional platform	
kinematics	4 KUKA omniWheels
length	580 mm
width	380 mm
height	140 mm
clearance	15 mm
weight	20 kg
payload	20 kg
structure	steel
maximum velocity	0,8 m/s
communication	EtherCAT
power supply	24 V DC
battery	maintenance-free lead acid rechargable 24 V 5A battery

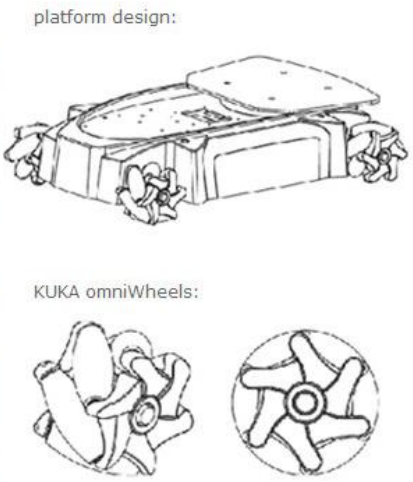


Figure. 135: Specified data for youbot

2.The more closed-loop control a system has, the more stable and safe it is. Based on this principle, me and my partners use as much closed-loop control as possible and change the open-loop control to the closed-loop control:

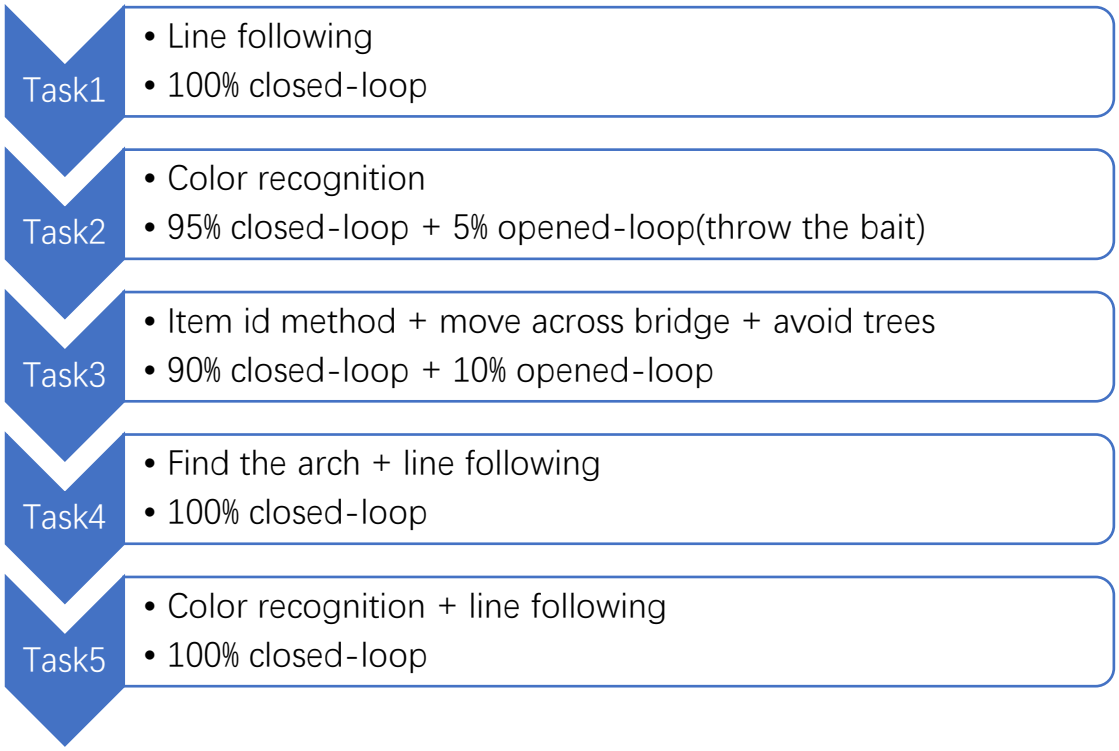


Figure. 136: The final percentages of closed-loop control

2.11.4 Further Work (Jiajun Huang)

Analysis and Discussion

After discussion with our team members, I find that there are still many things that

we could do to make the project more realistic.

1.In task one and five, the path we draw is too straight which is not possible in the real situation. Besides, we consider the whole floor as completely flat, which is not the occasion in reality, too. In the future, we will adjust the original map to a more realistic one and test our path following algorithm in that new environment.

2.In task three and four, the recognition algorithm we used is able to detect the ID number of the target object. The ID number could be printed out and then we could use the ID to find what the target object is. However, in real world, we could never find out the ID of one object because we could not label everything with an ID. Therefore, we could adjust the algorithm in the further work.

3.About other suggestions, the first one is that We could add the natural noise to the environment to better imitate the real world. For the second one, we could use deep learning techniques in the recognition algorithm to increase the accuracy of boundary classification and the pre-trained model could reduce the requirement of cpu and memory which could save money. The last one is that We could develop an automatic algorithm for adjusting parameters in PID which could reduce manpower and increase efficiency.

2.11.5 General Project Consultant (Ze Sheng)

In this project, I'm the general project consultant. Because I've done some relative researches about ROS in Chinese Academy of Sciences two years ago. So I know exactly what is the aim for each part of the project.

In research and development part, I conducted almost 80% part of the whole task. The keys are four core algorithms: line-following algorithm, color recognition, path planning and robot arm control. These four parts constitude over 90% part of our project. What's more, I also took charge in answering some questions about how to realize some specific functions in each part.

In article writing part, I share my thesis writing experience to my teammates to make every thing looks better. What's more, Yukai Song and me took charge in proofread and composing the both slides and final reports. The time is limited so there may still exist some problems, we've tried our best to improve that.

As a member of a professional team, I made the idea of a better environment, dynamic water and night mode to give our client a better experience. I do believe we made it. To increase our safety, adding closed-loop control is an idea I made and with the help of my teammates, our system is safe enough.

3 System Integration, Results and Discussion

3.1 Integration of the system

3.1.1 Hardware Integration Part

3.1.1.1 Why We Need to Integrate Other Hardware

As mentioned in the above, although youbot from KUKA owns a robotic arm that can help it to pick up and release the fish food easily, it does not have any other sensors or camera. It is impossible for the youbot without the assistance of any other sensors to complete task 1 to task 5.

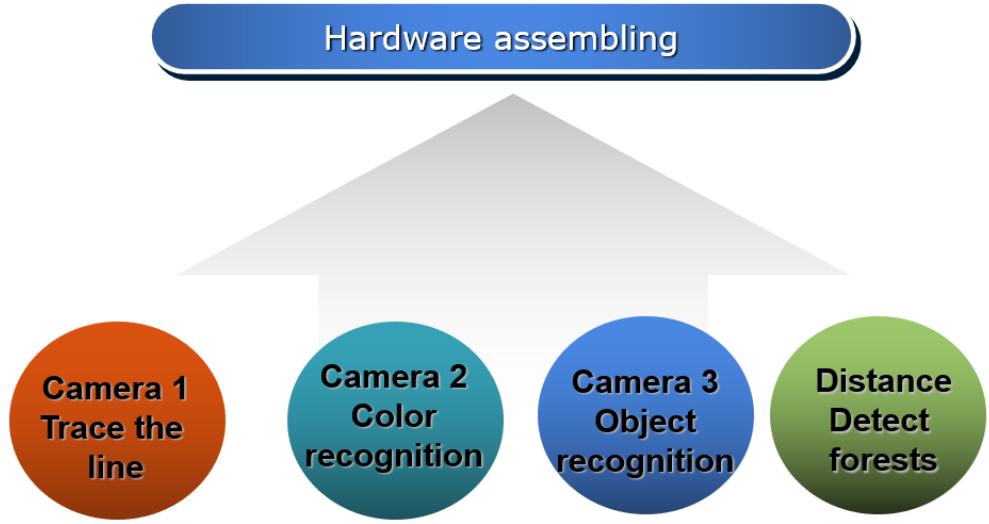


Figure. 137: Hardware assembling

3.1.1.2 Introduction About the Function of Assembling Sensors

In order to help youbot from KUKA to complete line tracing, color recognition and object detection from task 1 to task 5, 3 cameras and 1 distance sensor are assembled on the robot.

From 错误!未找到引用源。 shown above, we can see that the four sensors (Camera1,2,3 and distance sensor) with various functions to finish can enable the robot to finish all tasks and the details of their usage will be illustrated in the next part.

Camera 1: Observe the line to make robot keep close to the traced line and

recognize the color in front (Task5) and principles behind it are 2.3.2 PID Controlled Line Tracing in Task1,4,5 (Fangze Xu)**.**

Camera 2: Recognize the color on the right (**Task2**) that mainly works for the task 2 of the project. The algorithm that controls the behavior of the robot is 2.5 Color Recognition Algorithm (Ze Sheng).

Camera 3: Recognize the object in front (bridge and arch) that works for task 3 and 4. This function is mainly achieved by the effective usage of camera API of Webots, which is an amazingly powerful API. The idea behind of it can be checked at 2.7.4 Route Control Algorithm Based on Object Recognition (Zihao Huang).

Distance sensor: The function of the distance sensor is to detect the distance between the obstacle in front of the robot. This can tell the robot that it is too close to the forest and it is time to turn right to avoid collision.

The following part will demonstrate how those functions are realized, based on the data detected by sensors.

3.1.1.3 How We Use Distance Sensor

In this project, we use the Distance sensor Node and write the lookup table to design our own distance sensor and this distance sensor should also be installed on the Bodyslot node. After the lookup table is set, we can use it to find the distance between the obstacle and the robot. This feature will be applied in the obstacle avoidance section of Task 3 when the robot is crossing the bridge and becoming too close to the forests.

3.1.1.4 How We Use Camera

3.1.1.4.1 Camera Recognition Function

The camera node can be utilized to compute a OpenGL image and the red, green and blue channels (RGB) can be extracted from the resulting image by wb_camera_image_get_∗-like functions. This is very useful for color recognition. Apart from that, the recognition node of the camera can provide much information about the solid nodes with recognitionColors like object ID, relative position and relative distance. To be more specific, every solid object has its own special id , so the recognition node can identify the objects respectively and relative position and orientation of the solid node can help the robot guide to the right position.

3.1.1.4.2 Auto-Focus of Camera

Before realizing auto-focus, it is necessary to add focus node that is not very convenient. We enable the distance sensor through distance_sensor = wb_robot_get_device, which can determine the distance between object and the camera by using const double object_distance = wb_distance_sensor_get_value(distance sensor)/1000, due to the reason that the unit is millimeter . Finally, we can use wb_camera_set_focal_distance to achieve the auto-focus effect.

3.1.2 Software Integration Part

At this stage, controllers for different tasks are integrated into a whole part, which can execute all the required actions in a specific order. Referring to Zihao Huang's main loop logic, the structure of the final controller is vast and complex, involving conflict problems if in simple code structure, such as if-then structure. Thus, a more practical scheme is to introduce Finite State Machine mechanism, also known as case statement.

In the controller, Mealy finite state machine is implemented as the input state also influences the output in multiple logical judgment. For the codes, the critical variable current_task has five states, exactly corresponding to five required tasks and transition of tasks begins only when all the requirements in original task are achieved. The detailed steps are shown in the flow chart.

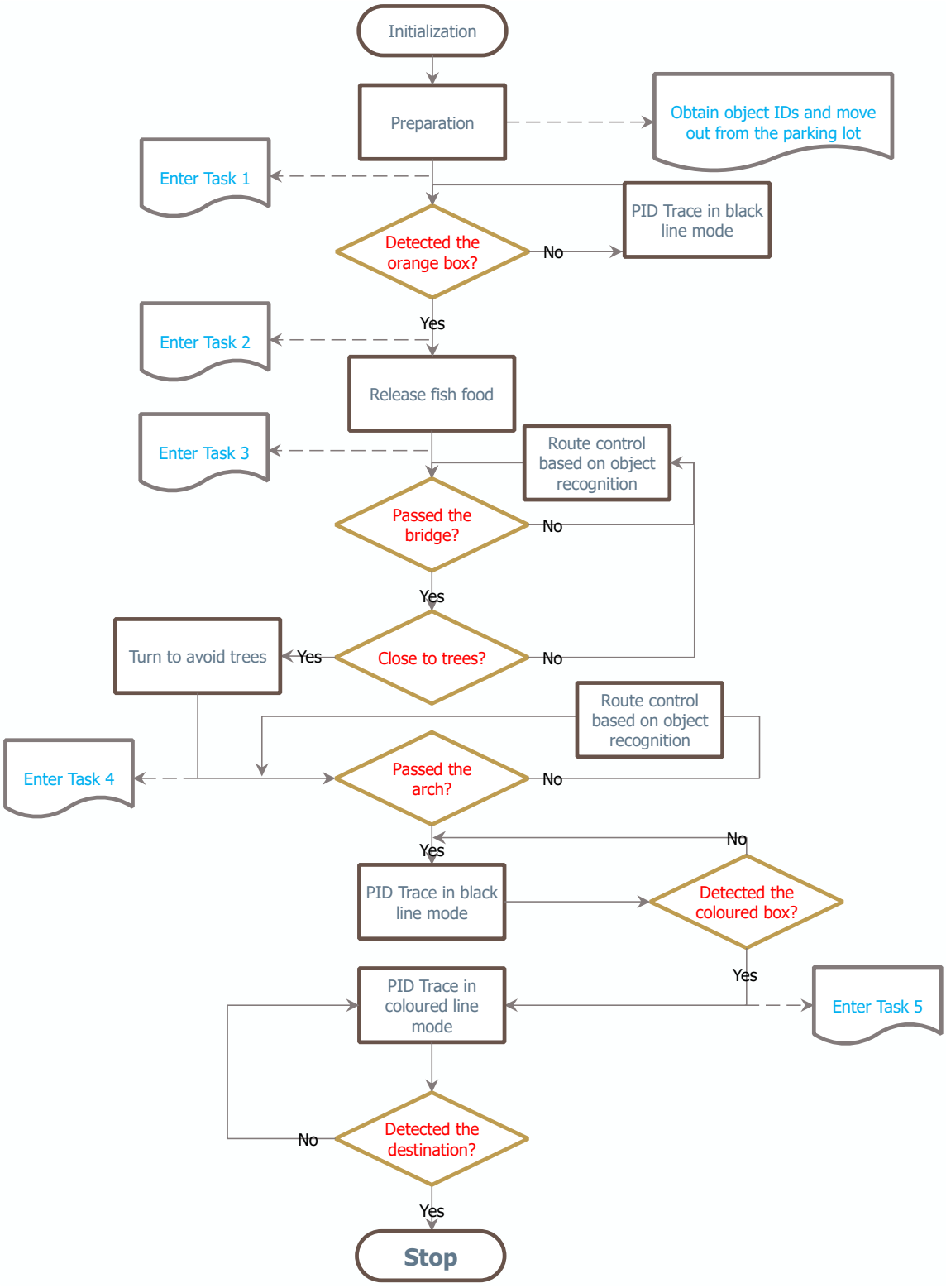


Figure. 138: Flow chart in controller

In the process of integration, although linking problem for different tasks can be solved by finite state machine, another problem is how to combine different functions in one task, such as connecting route control based on object recognition with distance

measurement in task 3 and connecting route control with line tracing in task 4.

As the main loop is run each time step, one flag variable should be defined so that whether the first step is finished can be stored in it. The if-then structure can be used in subfunction to judge what to do in one task according to the value of the flag, and once the first step is done, the flag value can turn to 1 from 0.

Next problem is how to present this in codes. The earliest solution is to define a global variable at the beginning of the controller, but it is replaced by using a pointer, which can be declared in the main function and its value is input to the subfunction, so that the debugging process will be simpler.

3.2 Group Results and Discussion

3.2.1 Line Detection:

At the beginning, we choose three ground sensors to be the line detector. The binary signal of these three ground sensors should provide us enough information about the edge of line. This method is commonly used in practical line trace competition because the price of infrared ray sensor is much cheaper compared with linear CCD camera. However, in webots, our environment is quite different from the practical situation. The line is stretched by the “scale” operation and the edge is blurred heavily. We find that the robot using ground sensor cannot pass the turning even the speed is 4, if we want to solve that problem, we need to make a very large picture in photoshop with millions pixels times millions pixels and use highly skilled method to draw lines with same width, it seems almost impossible for our team members because we are the freshmen of Photoshop. Finally, we choose camera as the sensor to detect the line. And as for image processing method to detect the line, the commonly used way is edge detection, by adding positions of two edges, we can roughly get the position of traced line, in fact, the edge-detection method is also getting the mid-value of the line, and its complexity is , however, based on the poor quality of background design, it cannot support edge-detection method well, we finally choose mid-value method to detect line. For further details about mid-value, please look through the results and analysis in PID trace.

3.2.2 PID Control

At the beginning, we only used one PID system to control the turning speed of robot, however, we found that we can increase the speed when robot is tracing along straight line or the line with vet small angle. The same gain PID can still handle it, so we do not need spend time changing PID. So we build another PID system, which controls the robot's offset speed, where we can increase the while tracing along the straight line, and slow down while turning. Which significantly reduce the overall time cost in 5 tasks.

3.2.3 Code Integration

Because the language wo choose is c, is impossible to load opencv to help us do the image processing. After the beta type of our TDPS project was finished, we found that there are too much functions in one c script, it is crowded, so we decided to use the dynamic library to build new head files(.h) and source files(.c) to store the new function, and change the linking in the Makefile of webots, then we can move all these new functions to source file. However, we failed to build dynamic library, so we have to abandon this method but improve function codes instead.

3.2.4 Comparison Between Two Methods of Color Recognition

Read the pixel, from the left-top of the image, the camera read the pixel and compare them with our RGB value namely in order to determine where and when the robot should take actions. This method is valid, it can be applied to this project.

3.2.4.1 ROI Method:

Cut the image, only study the region we are interested in , and add reverse traversal, for example, if we want to recognize yellow color, because the yellow region is in the bottom of the image, so we count the number of yellow pixels from right-bottom side, if the number of pixel meet expected standards, robot will proceed to the next step.

There are several differences between two methods:

1. Coordinate axis

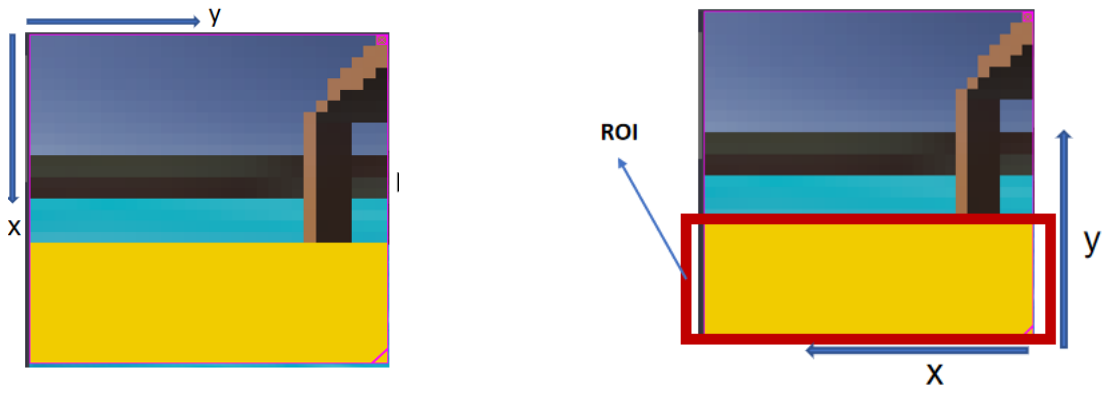


Figure. 139: Comparison between Color recognition of traditional and ROI method

We can see that in traditional method, camera read the pixels from left-top while in ROI, it's from right-bottom side. Actually, there is no difference in essence, only because we choose to add reverse traversal to make the counting of compiler much easier. Since we set the ROI is x*30%y and the region is in the bottom side of the whole image, reverse the traversal so that the camera do not need to read the pixels of whole image.

2. **Efficiency**

ROI method is much more efficient than the traditional method. In traditional method, camera count the pixels of specific color, and need to deal with loads of counting. However, C programming language is not good at dealing with matrix operation as matlab or python. Instead, ROI only focus on the specific region, if the pixels of specific color meet the threshold amount in this region, robot will take action, and it is much more efficient.

3. Stability

In traditional method, once the camera capture one yellow pixel, robot will take action. It is easy to be disturbed: what if one day there appears a sunset, maybe the robot will recognize the yellow pixel in sunset and act a wrong movement. Besides, traditional method causes an increase in open loop control since no feedback is applied.

Figure. 140: Open loop control of traditional method

ROI is different. The counting number of pixels will always give feedback to the system, comparing with the threshold value, thus forming a more close-loop control.

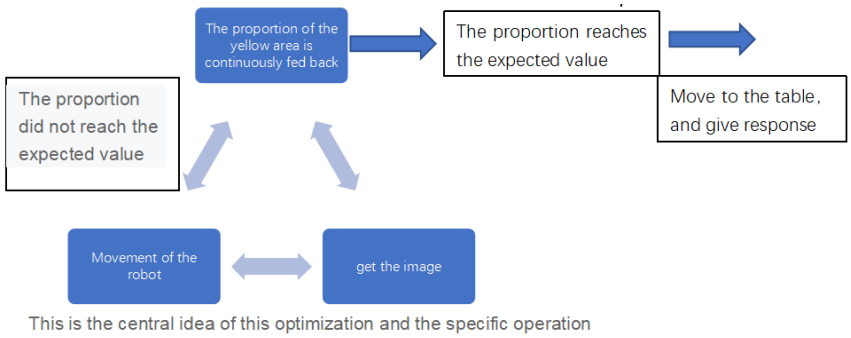


Figure. 141: Close-loop control of ROI method

3.2.5 D* Algorithm

3.2.5.1 Introduction of D* Algorithm

D* algorithm is a mainstream robot path planning algorithm, which has also been considered in this project. The following part of this chapter will illustrate the principle of it in detail.

3.2.5.2 Digital Map

Digital map method is the core part of the whole D* algorithm. First, the map should be digitalized into 0 or 1, where 0 represents free place and 1 represents obstacles. Then the compiler does the calculation to get the best(nearest) path from the robot to its destination. There are two rules for determining the path: 1. The fewer 0 we got, the better the map is, 2: the path cannot include any 1 element (robot may crash) or besides any 1 elements (not save). When the path is determined, compiler only dispatches the nearest 0 point to the robot and then gives a next point when our robots reaches the former one.

That is the basic rules of D* algorithm and digital map method. We have considered

it as the main path planning method while Zihao Huang's route control based on object recognition algorithm is faster and more efficient (since Zihao Huang has illustrated the details of this method clearly in his personal part, the later route control based on object recognition method is our further work). So, we abandoned this method at last. What is more, D* algorithm needs a powerful calculating ability for matrix, so MATLAB could be our ideal Compiler.

3.2.5.3 GPS

GPS is a node that can be used in path planning. The research of GPS method in cancelled because of the development of id finding method is too fast. Here we only talk about some basic operation of developing GPS method. GPS needs two coordinates, one is for our robot, the other one is for the world. These two coordinates can be converted to each other. In this occasion, we should determine the position of the destination in the world coordinate. Then we change it into robot coordinate. At last give commands to the robot and it can go to the destination.

We believe GPS will be used in 60% TDPS groups, so I choose to develop our unique route control based on object recognition. This method is not written in the manual or reference book, so it is really creative. Moreover, we planned to make our robot have a sight, so we want it to recognize the bridge and the arch by itself. It is really convenient for installing our deep learning algorithm in the future, which means a big leap in intelligence of the robot.

3.2.5.4 The Application of Robot Vision in Retrieving Objects

This chapter mainly discusses how to add visual algorithm to the robot and make it take the fish food in any position (if the fish food is within the range of taking).At present, our traditional reference method has achieved a 100% takeup rate, so the development priority of this technology is very low. In short, we only do this technology development when we have a lot of spare time.

The first is the installation of the camera. Our idea is to install a camera above the car, so that we can shoot the object on the table and identify it in the next step. After getting the camera, we need to extract the object. In fact, we only need to extract the part of the red basket, so we need to use color recognition and set the red area to white and the other areas to black to make the image processing easier.

Then, we use the calibration method to transform the photo coordinate system into the world coordinate system. Notice that the photo coordinate system here is only X and Y, which are two-dimensional. Therefore, we need to measure the distance between the camera and the object to obtain a three-dimensional information. Calibration method. We envision the traditional camera calibration method, which is a calibration method requiring reference objects. However, we did not do much research on the algorithm after that. Due to limited time, we terminated the development of this technology.

But there is no doubt that this technology will have a great future, once the algorithm is successful, then the robot will become quite intelligent.

3.2.6 Route Control Based on Object Recognition Method

In fact, route control with only one object of reference in object recognition is not enough to plan a complete path, because data such as the relative distance and orientation is determined by the position and the orientation of the camera, which means if the camera is not put towards the X direction perfectly, error may occur. Although recognizing two objects at the same time can solve this problem, that would bring many difficulties in algorithm design, which also greatly increase the simulation load. Fortunately, the required controlled part is almost a straight line, which does not involve the complex geometric curve, and the recognition function stops when the car gets too close to the bridge of the arch, so only recognizing one object is also acceptable.

If a large displacement interference is applied, the car can also move towards to the target object, but the turning process may not go perfectly as the original orientation of the car is shifted slightly due to interference and only one object of reference while the predetermined turning angle is fixed.

In conclusion, using multiple objects of reference would increase the stability and capacity for more complex route, but it may be much more difficult to design and bring huge amount of calculation. Although in our project the existing method is enough to satisfy the requirements, we hope to provide the direction for further improvement.

3.2.7 Refinement to the Bridge

During the task 3, we found out that crossing the bridge is not as simple as we consider and after the discussion, we realized that this problem is most relevant to physical issues. To complete this task, 3 physical factors are required: proper friction, enough power output, and a bridge structure suitable for the robot.

Initially, the friction factor on the bridge was set as the same as that on the ground-20. However, it was observed that 20 friction factor will lead KUKA to idle on the ramp. On the other hand, for a friction factor as large as 40, a tiny deviation in four wheels' stress condition will result in noticeable deviation in KUKA's overall direction, thus failing to cross the bridge by colliding with the fence. Therefore, the friction is set as 30. Subsequent tests prove this number is reasonable for the same power output as on the ground.

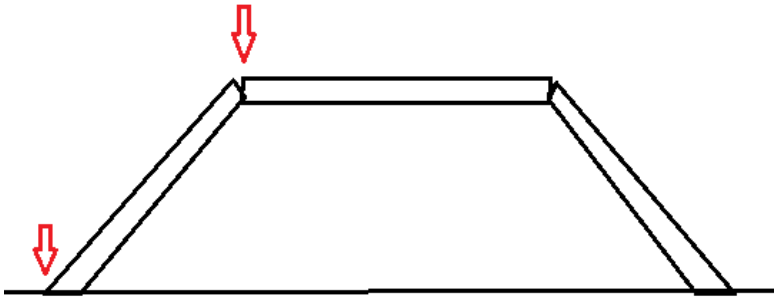


Figure. 142: Original bridge structure

The first design of the bridge is a connection among 3 plates. However, on the two connection points as indicated with red arrows KUKA is very likely to get stuck. That is because the chassis of this robot is very low, unsuitable to go through highly undulating terrains.

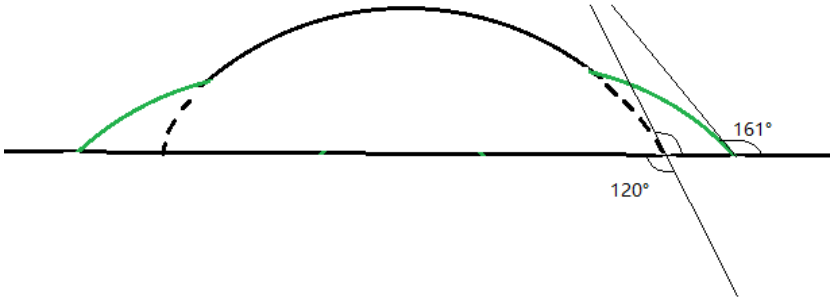


Figure. 143: Refined bridge structure

The refined structure is composed of 3 arcs. The radius of the big arc is 3m, and that of the two small arcs is 2m. Such a curvature can allow youbot from KUKA to pass without being hindered.

The 2 green arcs are auxiliary ramps. Because of them, the angle between ramp and the ground is enlarged, thus preventing collision and stop.

With those refinements, our youbot can finally cross the bridge without any problems.

4 Conclusions

We finally build up the robot that has functions of line tracing, color recognition, bridge crossing, arch detection, color box detection as well as path following that is required from task 1 to task 5. 5 tasks successfully test the powerful functions of the robot. In addition to work on the robot, we also design two modes of environment: day mode and night mode, which can make the environment closer to the real environment.

As for the methods of 4 key algorithms:

1. We found mid-value method of line tracing is great for continuous angle turns and least square method suits turns of the right angle. PID control can wonderfully help the robot to trace the line with little error and adjust the speed of it within the safety margin. The perfect combination of those methods can ensure our robot tackle any kind

of turns easily.

2. Robot arm control is essential part of task 2 and the stability is main concentration of this algorithm. Position set, middle state and mechanical claw propulsion method are implemented to increase the successful probability of picking up and releasing the object. Finally, we arrive a successful probability of nearly 100%.

3. ROI color recognition that only focuses part of the photo works stabler than traditional threshold method which concentrates the whole picture. Apart from that ROI requires fewer computation than the traditional threshold method, which can help our simulation run faster

4. As for path planning part, we found packaging code method of camera ID finding is much better than tradition digital map method. Based on the relative position and orientation given by the camera ID, our robot can be guided to pass the bridge and arch. Similarly, the distance sensor can tell the robot the relative position of forest and helps the robot to avoid colliding.

5 References

[1]. Ding, L. , & Goshtasby, A. . (2001). On the canny edge detector. Pattern Recognition, 34(3), 721-725.

[2]. Canny, J. . (1986). A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(6), 679-698.

[3]. Beeran Kutty, Suhaili, Saaidin, Shuria, Megat Yunus, Puteri Nor Ashikin, & Abu Hassan, Suhana. (2014). Evaluation of canny and sobel operator for logo edge detection. International Symposium on Technology Management & Emerging Technologies. IEEE.

[4]. Nguyen, P. M. L. , Cho, J. H. , & Cho, S. B. . (2014). An architecture for real-time hardware co-simulation of edge detection in image processing using Prewitt edge operator. 2014 International Conference on Electronics, Information and Communications (ICEIC). IEEE.

[5]. Shrivakshan, G. T. , & Chandrasekar, C. . (2012). A comparison of various edge detection techniques used in image processing.

[6]. Jing Huang, Chen Zhao and Mingming Zhou, "Identification of objects for fast color space micro soccer robot based on transformation", Journal of harbin institute of technology, vol. 35, no. 9, pp. 1036-1039, 2003.

[7]. D Martin, C Fowlkes, D Tal and J Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and

measuring ecological statistics", International Conference on Computer Vision, vol. 2, pp. 416-423, 2001.
[8]. D.G. Prasad, P.S. Manoharan and A.P.S. Ramalakshmi, "PID control scheme for twin rotor MIMO system using a real valued genetic algorithm with a predetermined search rang", Power Energy and Control (ICPEC) 2013 International Conference on, no. 448, pp. 443, Feb. 2013.
[9]. P.S. Kannan and P. Sheenu, "PID Control of Twin Rotor MIMO system", IJARBEST, vol. 3, pp. 2456-5717, 2017.
[10]. Sunil Kumar Mishra and Shubhi Purwar, "To design Optimally Tuned FOPID Controller for Twin Rotor MIMO System", Motilal Nehru National Institute of Technology Allahabad IEEE, 2014.
[11]. D. N. Vila-Rosado and J. A. Dominguez-Lopez, "A Matlab toolbox for robotic manipulators", ENC 2005. Sixth Mexican International Conference on In Computer Science, pp. 256-263, 2005, September.
[12]. G. Maletzki, T. Pawletta, S. Pawletta and B. Lampe, "A model-based robot programming approach in the MATLAB/Simulink environment In: Advances in Manufacturing Technology - XX", 4th Int. Conf. on Manufacturing Research (ICMR06), pp. 377-382, September 05–07, 2006.
[13]. www.aigei.com

6 Appendices

Table 7: Simulation multiples in ordinary configuration

	Ordinary Setting		Advanced Setting	
Time	original	optimized	original	optimized
0:00	2.71	2.21	4.81	4.03
0:10	2.58	1.93	5.09	4.13
0:20	2.62	1.13	5.58	4.19
0:30	2.65	1.81	4.89	3.98
0:40	2.76	2.01	5.16	4.13
0:50	2.62	1.96	3.31	4.01
1:00	2.78	1.88	4.67	4.22
1:10	2.58	1.89	5.05	4.22
1:20	1.57	2.01	4.91	3.93
1:30	2.69	1.85	5.25	4.11
1:40	2.65	0.87	4.67	4.09
1:50	2.95	2.04	4.96	4.07
2:00	2.69	2.15	4.94	4.23
2:10	2.55	2.15	4.96	3.93
2:20	2.54	2.12	5.16	3.96
2:30	2.78	1.82	4.87	3.86

2:40	2.55	1.83	5.16	4.27
2:50	2.51	1.96	4.97	4.12
3:00	2.66	2.07	4.69	4.04
3:10	2.75	2.13	4.78	4.12
3:20	2.69	1.65	5.12	4.07
3:30	2.55	2.21	4.97	4.12
3:40	1.98	2.16	4.77	4.07
3:50	2.63	2.28	4.88	4.16
4:00	2.77	2.07	5.05	4.06
4:10	2.54	1.96	4.79	4.89
4:20	2.74	1.85	5.14	3.91
4:30	2.68	2.11	4.82	3.85
4:40	2.61	2.19	5.25	3.85
4:50	2.55	2.01	4.84	3.83
5:00	2.62	1.85	5.15	2.14
5:10	2.63	1.93	4.79	4.02
5:20	2.53	1.88	4.85	3.98
5:30	2.67	1.94	4.73	4.18
5:40	2.56	2.02	5.03	4.11
5:50	2.61	2.06	3.12	4.19
6:00	2.66	2.03	5.01	4.27
6:10	2.57	2.24	4.94	4.29
6:20	2.58	2.06	5.08	2.72
6:30	2.67	2.27	5.08	3.87
6:40	2.57	2.12	2.15	4.15
6:50	2.73	2.28	5.07	3.85
7:00	2.78	1.92	5.27	4.06
7:10	3.01	2.14	3.87	4.07
7:20	2.92	1.94	5.12	4.23
7:30	2.66	2.14	4.82	4.04
7:40	2.53	2.15	4.75	4.21
7:50	2.75	1.83	5.05	4.14

Table 8: Simulation multiples in advanced configuration

	Ordinary Setting	Night Mode
Time	optimized	
0:00	2.21	1.54
0:10	1.93	1.52
0:20	1.13	1.72
0:30	1.81	1.81
0:40	2.01	1.62
0:50	1.96	1.57
1:00	1.88	1.64
1:10	1.89	1.58
1:20	2.01	1.76
1:30	1.85	1.86
1:40	0.87	1.75
1:50	2.04	1.54
2:00	2.15	1.66
2:10	2.15	1.52
2:20	2.12	1.79
2:30	1.82	1.67
2:40	1.83	1.91
2:50	1.96	1.82
3:00	2.07	1.69
3:10	2.13	1.86
3:20	1.65	1.56
3:30	2.21	1.66
3:40	2.16	1.87
3:50	2.28	1.87
4:00	2.07	1.79
4:10	1.96	1.75
4:20	1.85	1.64
4:30	2.11	1.87
4:40	2.19	1.55
4:50	2.01	1.79
5:00	1.85	1.76
5:10	1.93	1.83
5:20	1.88	1.66
5:30	1.94	1.84
5:40	2.02	1.83
5:50	2.06	1.63
6:00	2.03	1.72
6:10	2.24	1.89
6:20	2.06	1.72
6:30	2.27	1.63

6:40	2.12	1.75
6:50	2.28	1.64
7:00	1.92	1.87
7:10	2.14	1.67
7:20	1.94	1.72
7:30	2.14	1.78
7:40	2.15	1.89
7:50	1.83	1.63