Movable Robotic Arm Platform for Laboratory

By

Chenxi Wang (3200115268)

Shihua Zeng (3200112422)

Zhuohao Xu (3200115233)

Zhizhan Li (3200110713)

Final Report for ECE 445, Senior Design, Spring 2024

TA: Yanzhao Gong

May 2024

Project No.31

Abstract

To be able to perform high-precision maneuvers in hazardous environments, we have created a movable robotic arm platform that replicates the precise movements of the human hand. The robotic arm is assembled on a wheeled base and can move omni-directionally, and by combining fine dexterity with powerful maneuvering, it can perform a variety of tasks ranging from complex industrial operations to rapid post-disaster rescue. It is also equipped with a camera for visual recognition tasks. Its design emphasizes flexibility, ease of use and user-friendly interaction, allowing it to adapt to a variety of scenarios without the need for bulky control equipment.

Key words: 4WD-4WS Chassis, Robotic Arm, Somatosensory Control

Contents

1. Introduction

1.1 Problem

In today's rapidly advancing technological landscape, there is a growing need for solutions that can perform tasks in environments that are either hazardous or inaccessible to humans. This encompasses a wide array of scenarios, from executing precise maneuvers in dangerous industrial settings to conducting rescue operations in disaster-struck areas where human responders are at risk of injury or are physically unable to perform the necessary tasks. Traditional methods, such as direct human intervention or the use of cumbersome machinery, often fall short due to safety concerns, physical limitations, or the inability to execute the fine motor skills required in many such situations. Moreover, existing robotic solutions that mimic human dexterity often require the operator to wear specialized equipment, which may not fit all users comfortably or be quickly deployable in emergency situations.

1.2 Solution

Our project proposes a movable multifunctional robotic arm system that is manipulated using a custom controller. The controller is equipped with three three-axis gyroscopes that are used to parse the movements of the hand, thus enabling the robotic arm to replicate the hand movements with high precision. This robotic arm is mounted on a 4WD-4WS omnidirectional chassis base with wheels and can be controller remotely in a variety of terrains and environments. The system is also equipped with a camera for visual recognition. Our innovative approach circumvents the limitations of direct human involvement and the need to wear bulky controls. The system is designed with a focus on flexibility and ease of use, allowing it to be adapted to a variety of operators and scenarios, from complex tasks in unsafe industrial environments to rapid response in post-disaster relief missions. Our solution combines fine and powerful maneuvering, solving the apparent paradox of needing both finesse and power in critical tasks, such as removing heavy debris to rescue earthquake victims. The design of the robotic arm emphasizes user-friendly interaction, and operators can seamlessly switch controls, increasing its utility in a variety of applications.

1.3 Visual Aid and Block Diagram

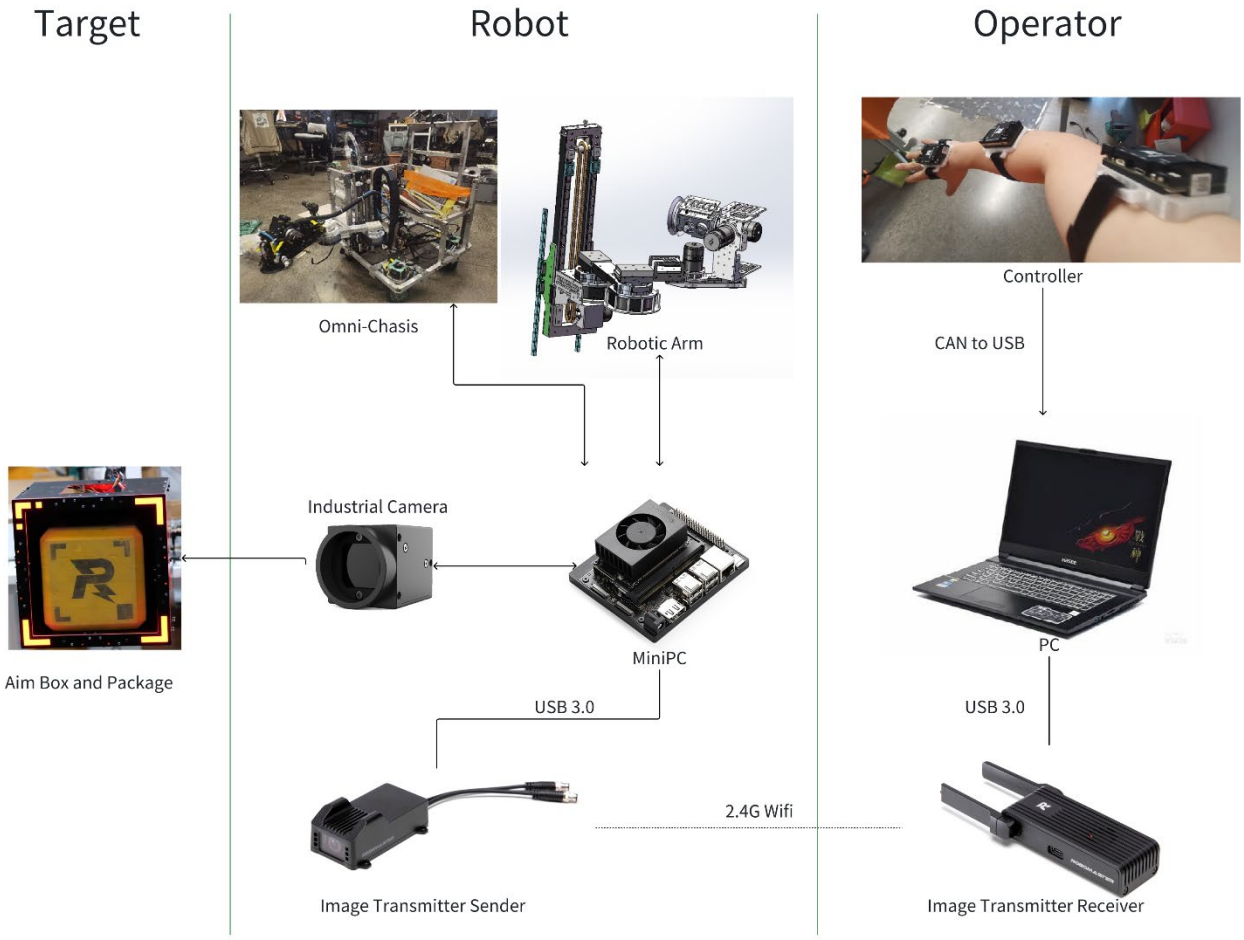

Figure 1.1 Visual Aid

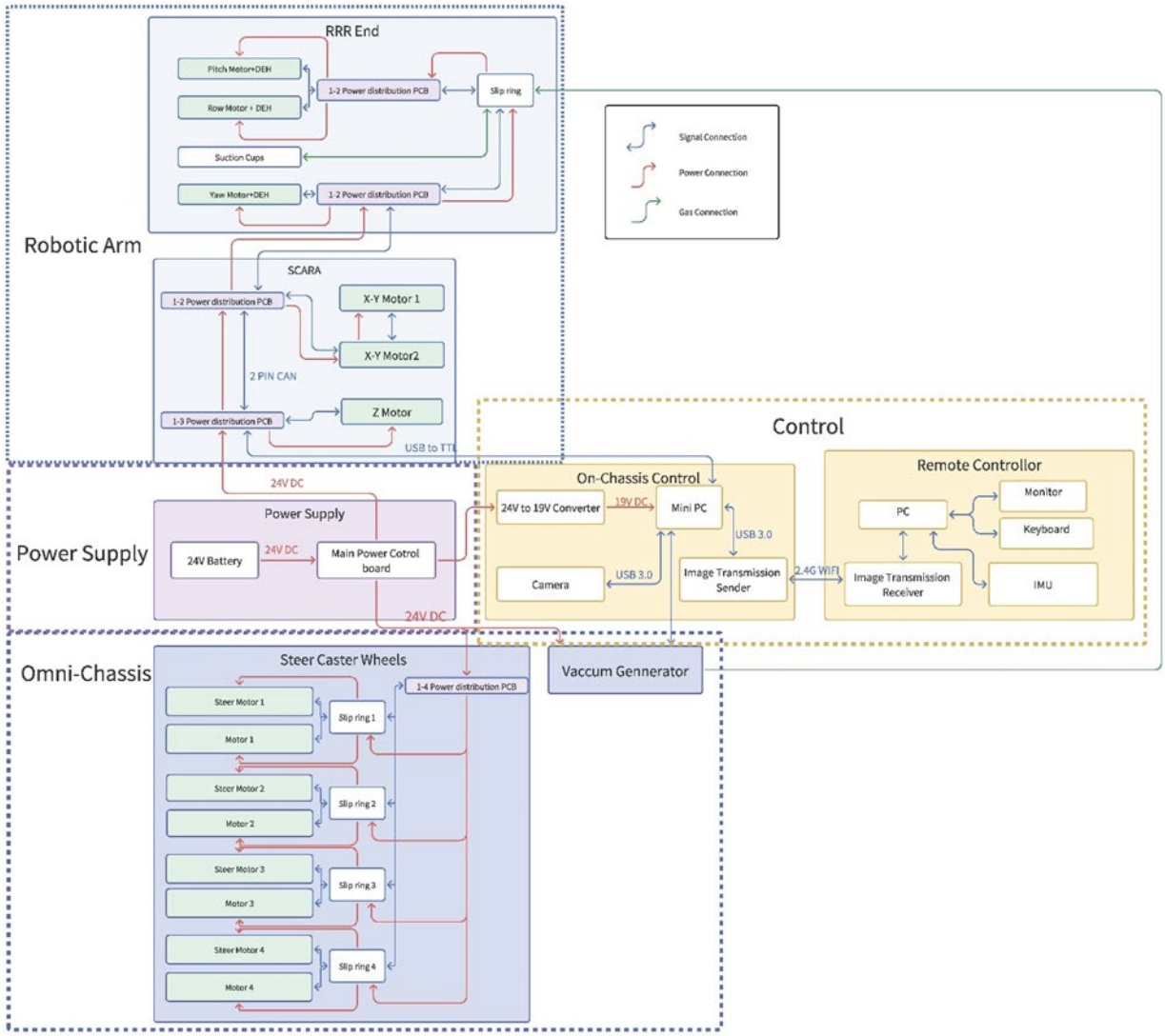

Figure1.2 Block Diagram

The Figure 1.1 simply shows how whole system works. The target for fetching is the yellow cubic with a R logo. And another target is the black box with red lights, which is used to place the cubic after cubic being transport by robot. The camera will recognize the position of the yellow cubic or the posture of the black box. Then this information will be calculated in the mini-PC in the robot side so the angle of the joints in the robotic arm can be determined and locked.

The image transmitter sender in the robot side will transmit real time video record to operator side. And the operator can use the somatosensory controller to record the position change of its arm, this information will be sent back to the robot side by image transmitter.

The block diagram in Figure 1.2 not only shows how the subsystems interact with each other's but also shows the circuit inside the subsystems.

2 Design

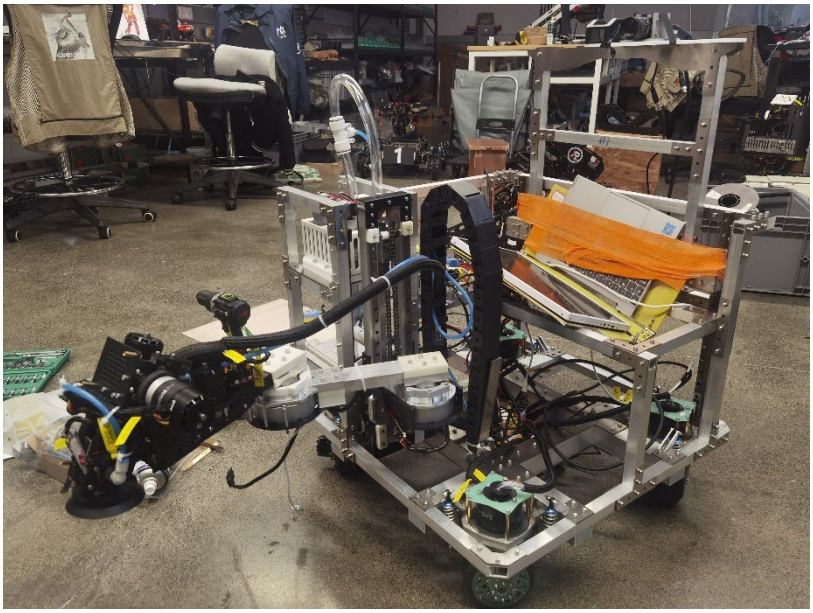

Figure 2.1 The overview of the physical structure of project

Our project has four subsystems as in Figure 1.2 and Figure 2.1 they are: Robotic Arm, Omni-Chassis, Power Supply and Control.

The table 2.1 shows the basic properties of the robot.

Table 2.1 Basic properties of the robot

Property	Value
Weight	40 kg
Folded size (Length* Width*Height)	590 mm * 590 mm * 590 mm
Stretch size (Length* Width*Height)	1100 mm * 835 mm * 995 mm
Actuator Number	M3508 Motor: 8 M2006 Motor: 1 GM 6020 Motor: 4 GO-M8010-6 Motor: 2 Vacuum Generator: 2 Suction Cap: 1
Maximum Moving Speed	3 m/s
Working Vacuum	-80 kPa

2.1 Robotic Arm

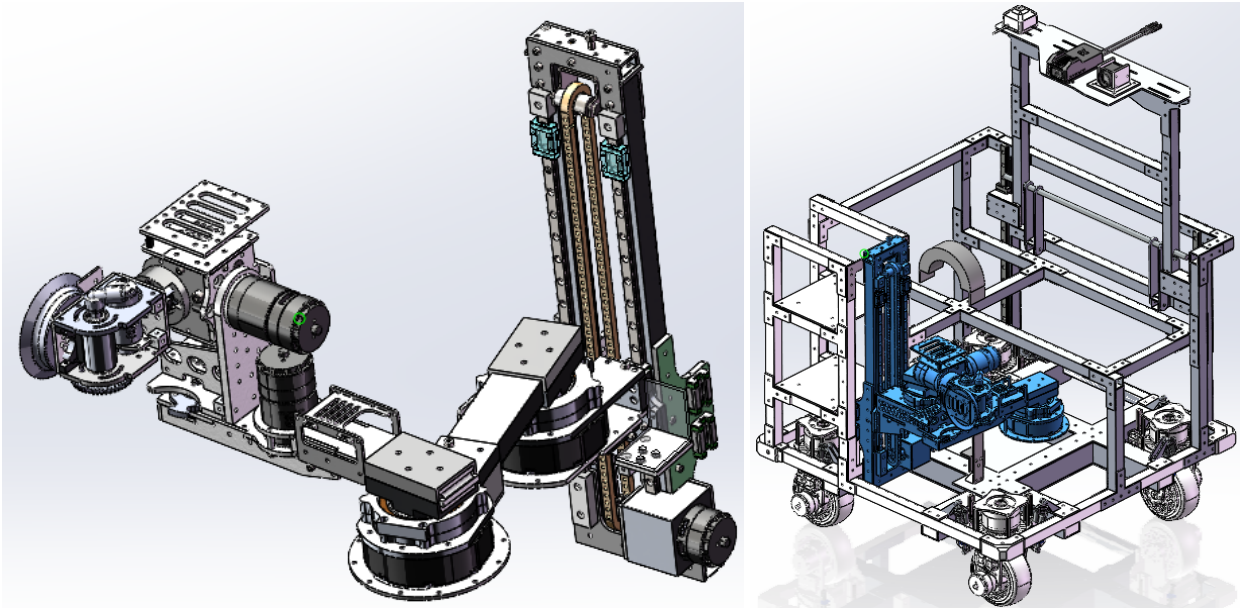

Figure 2.1.1 The robotic arm.

Figure 2.1.1 is the overview of the six-axis arm in the robot.

Robotic Arm System consist of two parts: Six-axis robotic arm and end-effector.

2.1.1 Six-axis robotic arm

The Six-axis robotic arm has six joint, joint 1 to 3 can be viewed as a Scara configuration (RRP, joint 1 is the translational joint P), joint 4 to 6 can be viewed as a RRR configuration (Jointed-arm, show in the Figure 2.1.1.1).

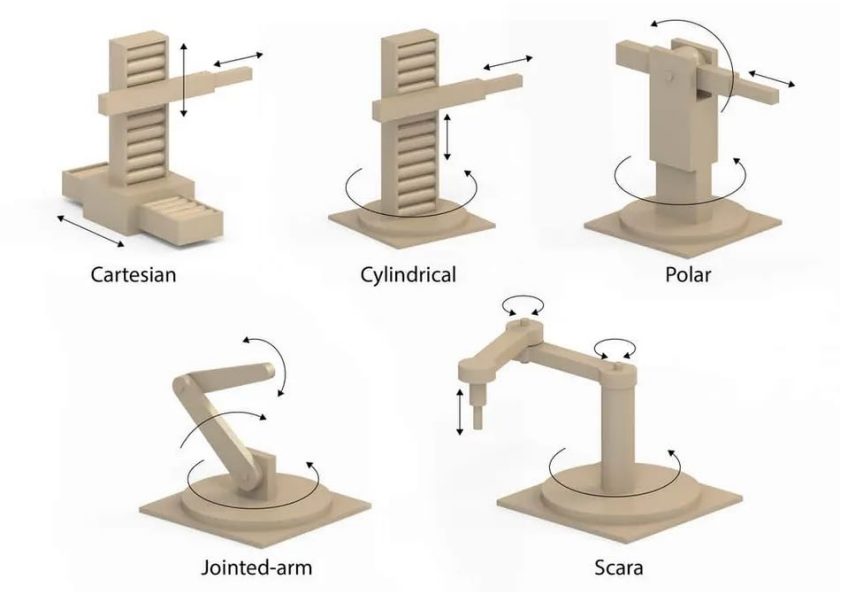

Figure 2.1.1.1 Several manipulators configuration examples

For a traditional RRR robotic arm, the second and third joints need to rotate against the self-gravity of the arm. The Scara robotic arm, however, the self-gravity is borne by the frame instead of the rotational joints. Therefore, that joint 1-3 is designed as Scara structure to decrease the total length and mass of the joint 4-6. Then less self-gravity and moment need to be supported by the joint 5. The robotic arm will have higher responsiveness.

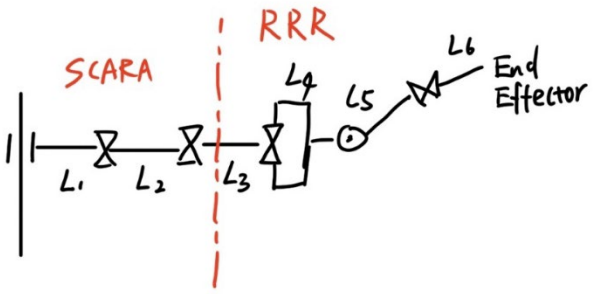

Figure 2.1.1.2 Sketch map for robotic arm

2.1.2 Parameter Determination of Arm

The parameter like the link length, rotation limit angles should be set so the workspace of robotic arm is larger than the target space.

The target space of the arm is generated by the yellow cubic on wood shelf, shows in Figure 2.1.2.1.

Another target space is the black box in the right side of the Figure 2.1.2.1, this box has DOF=6 after connecting a monitor stand.

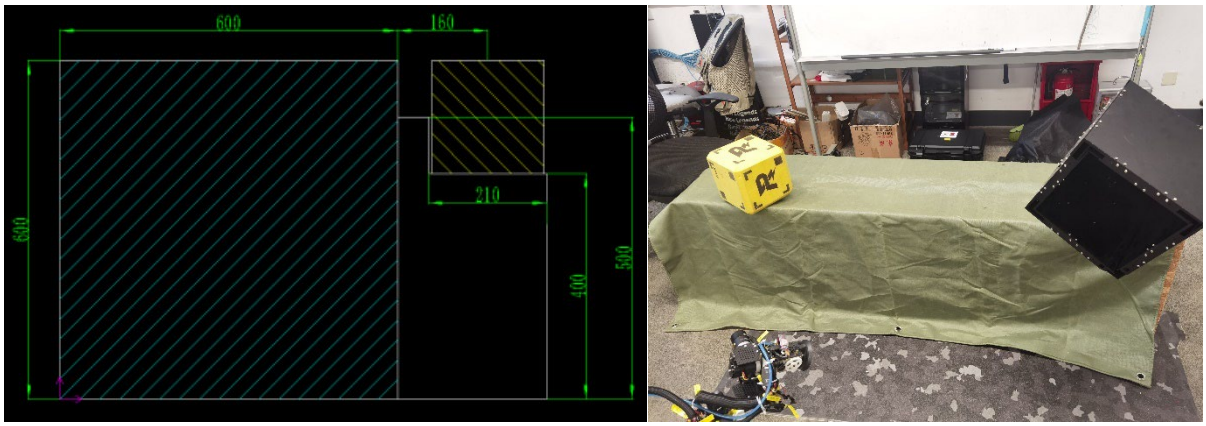

Figure 2.1.2.1 Shelf for placing target object.

To determine the parameters, first, I use D-H principle (table 2.1.2.1) to get the configuration of arm.

Figure 2.1.2.2 is the simplified model of arm to determine the left hand coordinates for each joint.

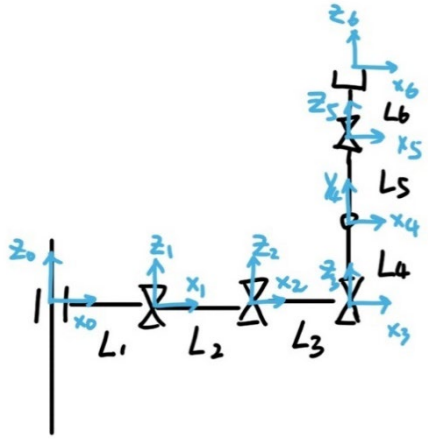

Figure 2.1.2.2 Definition of coordinates

Table 2.1.2.1 D-H principle's parameter table

i	θ	d	a	α
1	0	0	L_1	0
2	θ_2	0	L_2	0
3	θ_3	0	L_3	0
4	θ_4	L_4	0	90
5	θ_5	0	0	90
6	θ_6	L_6	0	0

Then robotic toolbox for MATLAB (Figure 2.1.2.3) is used for simulating the robotic arm.

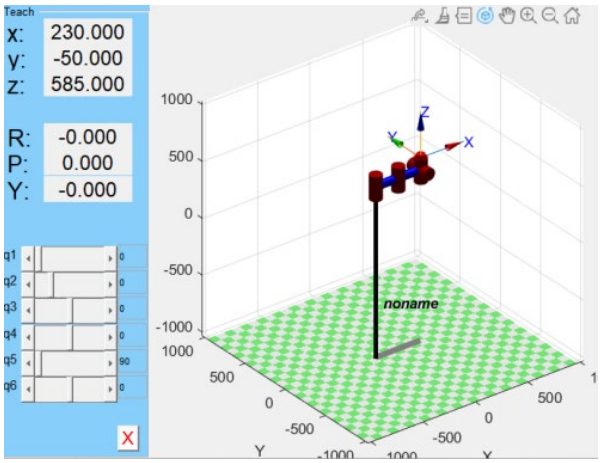

Figure 2.1.2.3 Simulation in MATLAB

Next, generate the target space referring to the dimension in Figure 2.1.2.4, then adjusting the limit of joint angles to generate the workspace.

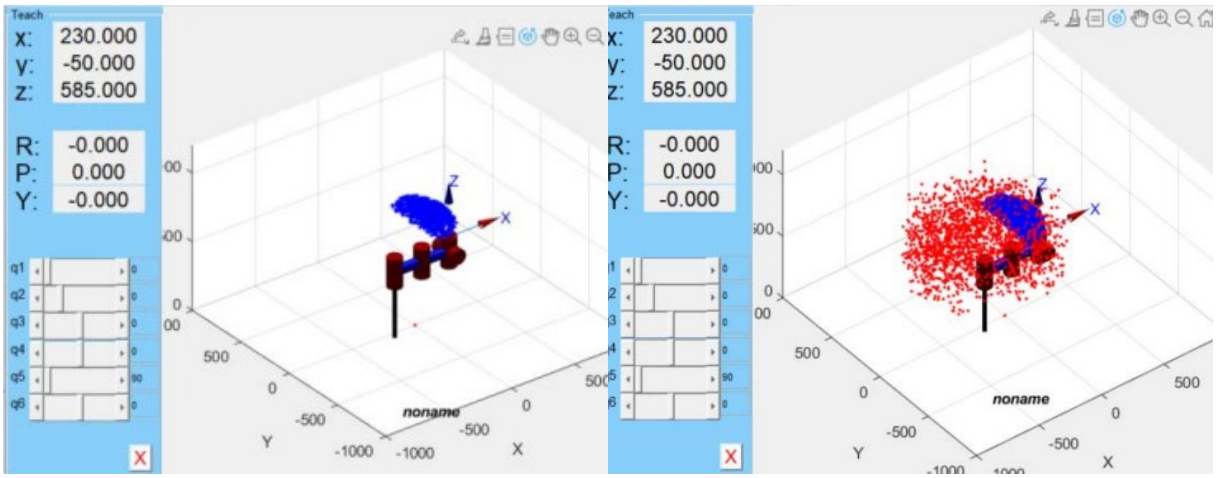

Figure 2.1.2.4 outcome of workspace (red) and target space (blue)

By adjusting the length of each link to make the workspace can fully cover the target space, which means the blue dot cloud can be fully contained by the red cloud. Under this parameter, the robotic arm can fetch the object from shelf and put it in the black box in varying orientation.

2.1.3 Structural design

In this part, the structure of joints 1-6 of the arm will be brief introduced one by one.

Joint 1(2.1.3.1) is a translational joint. The total mass of joint 2 to joint 6 is about 5 kg and may meet impact conditions. Therefore, I use chain transmission to undertake the heavy load from robotic arm length. And two linear rails are used to maintain the movement of joint 1 is stable and enforce the strength of whole structure.

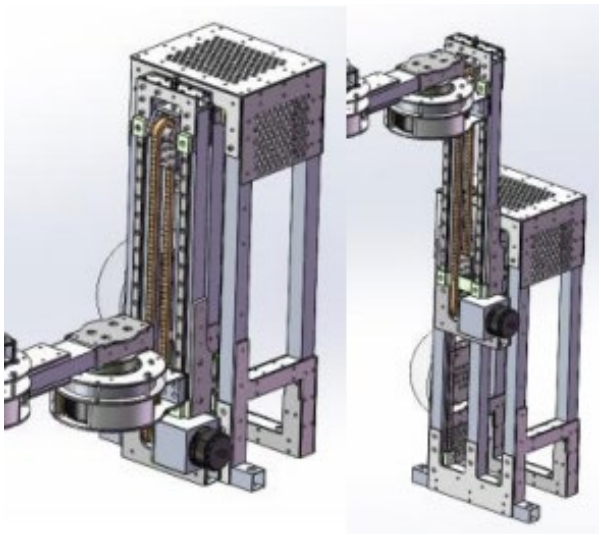

Figure 2.1.3.1 Joint 1 structure

By fixing two ends of the chain to chassis and lifting frame respectively, we can get double displacement in lifting.

The connection (Figure 2.1.3.2) between Joint 1 and Joint 2 is a CNC aluminum part. With two boards assembled vertically, the joint can take large bending moment.

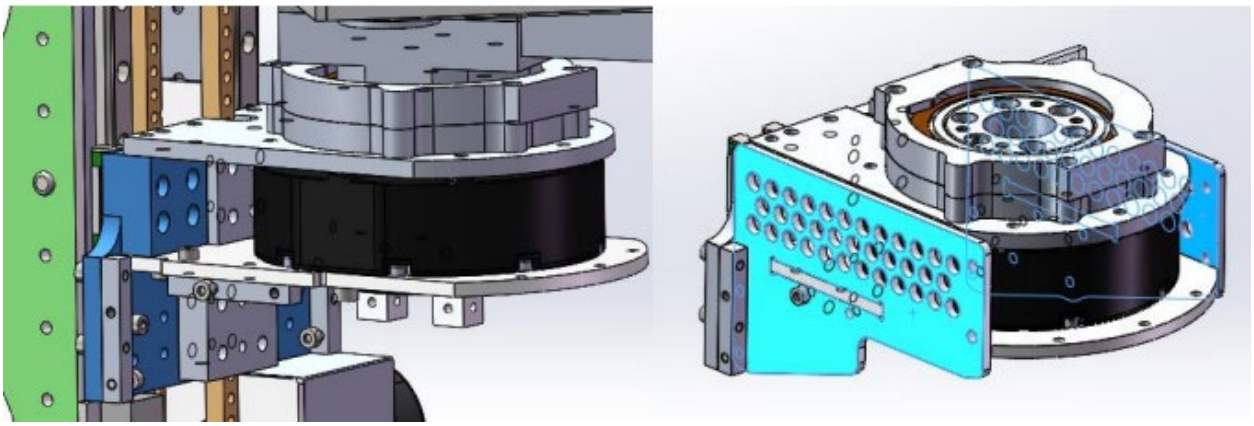

Figure 2.1.3.2 Connection between Joint 1 and Joint 2

Joint 2 and Joint 3 are rotational joint with similar structure (Figure 2.1.3.3). The length of their beams equal to 215mm.

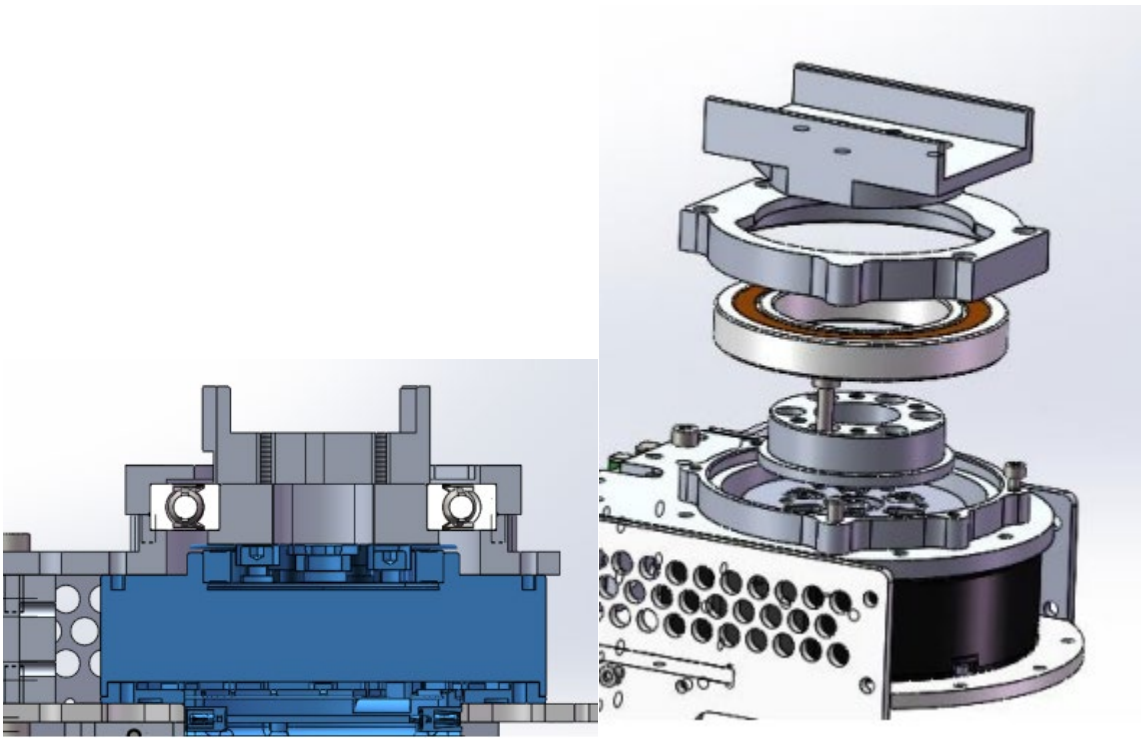

Figure 2.1.3.3 Cross section view and explosion view of joint 2 and joint 3

The 2040*2 square aluminum tubes are assembled with flange of motor directly with screw. The simulation (Figure 2.1.3.4) shows the tube's displacement under bending fit the design request.

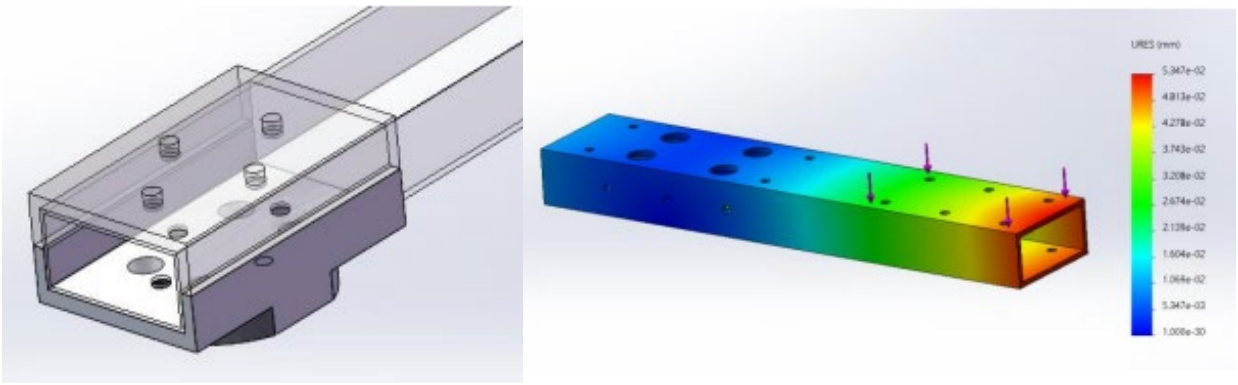

Figure 2.1.3.4 Ways the tube assembles and the simulation outcome.

The part under the tube (Figure 2.1.3.5) has another function of limit the rotation angle of the joint.

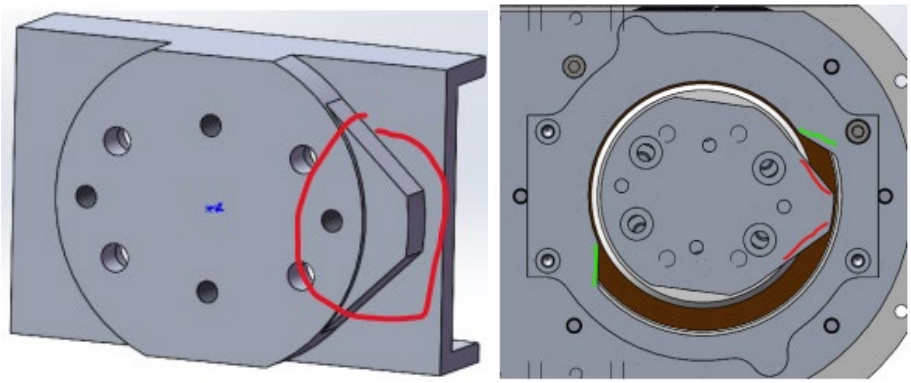

Figure 2.1.3.5 Ways the angle limitation works.

Joint 4 (Figure 2.1.3.6) is made up of four 3mm carbon fiber board to maintain the rigidness while decreasing weight.

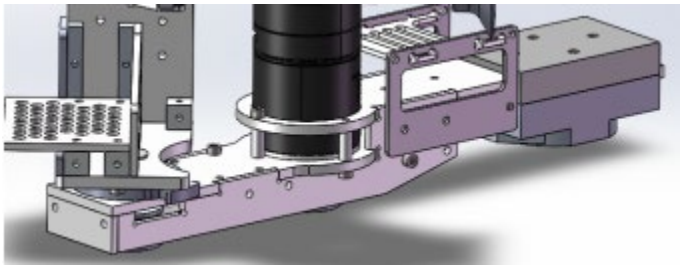

Figure 2.1.3.6 Overview of link 4.

To cut down the vertical height of joint 4, the rotation is transport with 3M timing belt so the motor can be put backward.

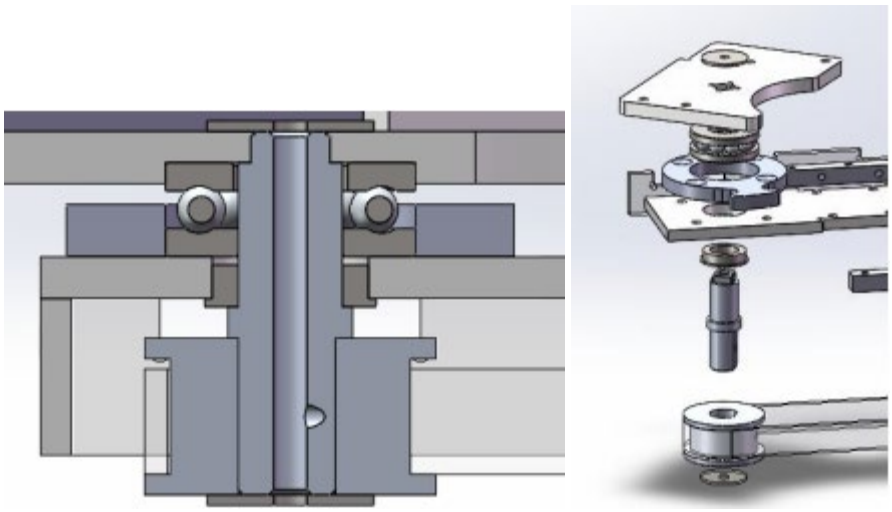

Figure 2.1.3.7 cross section view and explosion view of joint 4

The total mass of joint 5- joint 6 and the object fetching can be 4kg, so here uses a thrust ball bearing for axial load. Another deep groove ball bearing is used to take radial load from the tension of the timing belt.

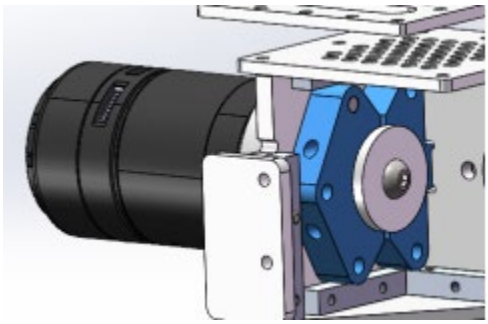

Figure 2.1.3.8 Overview of joint 5.

Joint 5 using a 1:27 planetary gear reducer for M3508 motor and use coupling whose design is open sourced by RoboMaster Team of Shanghai Jiaotong University in 2021 season.

There is a physical limitation designed for Joint 5 lifting motion.

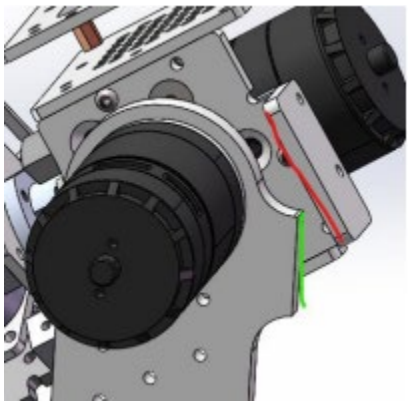

Figure 2.1.3.9 Physical rotation limitation by two boards.

Joint 6 need to decrease distance between the output shaft of motor and coupling, to decline the length to the end. The coupling is designed to have physical limitations function as well.

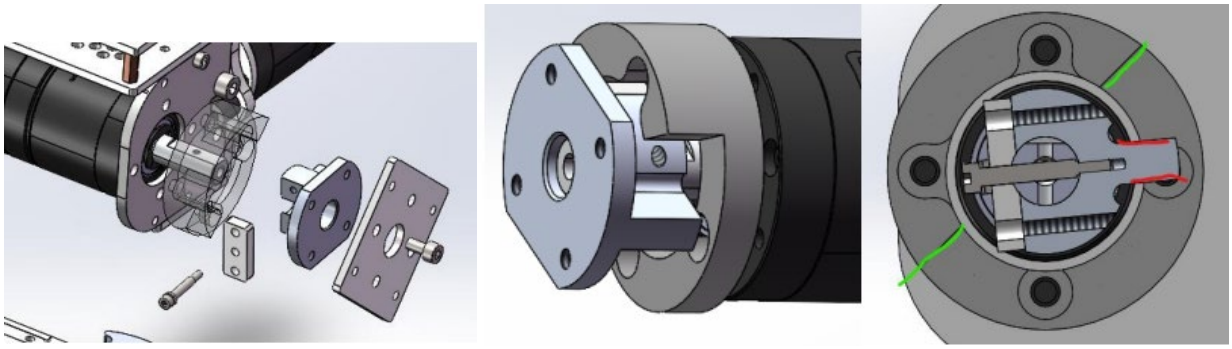

Figure 2.1.3.10 overview of joint 6 and its rotation limitation structure.

End effector is designed to have one rotation DOF so the suction cap can point downwards vertically, then the cap can fetch the object form its top surface rather than side face. It will be less possible to have object knock against the wall, if the object is placed in a notch.

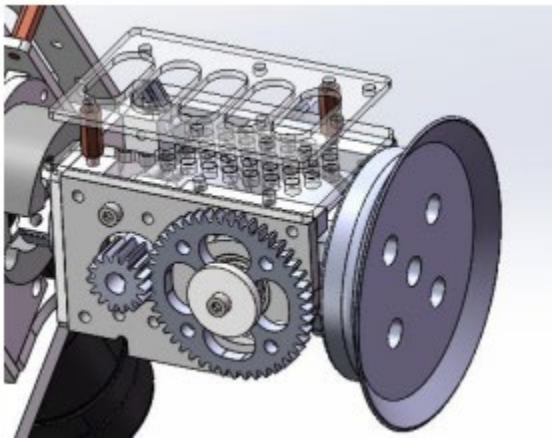

Figure 2.1.3.11 overview of end effector.

2.2 Omni-Chassis

An Omni-Chassis is a robot than can moving omnidirectionally without do much turning movement. most of omni-chassis use four-wheel sliding robot. The sliding robot use wheel with sliders (like McKnum wheels) and can moving omni-directionally by controlling the speed difference and rotation direction for each wheel.

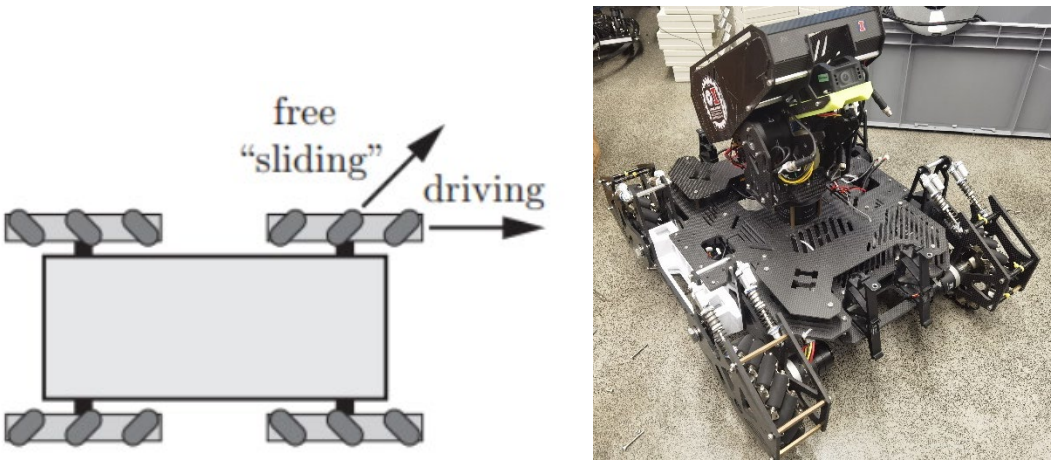

Figure 2.2.1 The McKnum four-wheel sliding robot.

Because the force generated for each wheel is not parallel with the driving direction, the force with cancel with each other's partly, result in low net thrust efficiency for motor torque. Meanwhile this kind of sliding robot cannot hold large weight, which will increase the friction force in the rubber sliders in wheel. The sliders will abrade quickly under such working condition. Therefore, the sliding robot cannot be suitable for our platform, whose design weight is about 40kg.

4WD-4WS (four wheels drive, four wheels steer) robot has same or even better moving flexibility to 4-wheel sliding robot and have highest net thrust efficiency within all the omni-directional chassis design. Therefore, 4WD-4WS design is the most appropriate for robot need hold large load. Though, four steering motors will add more weight to the platform. Considering the platform will install a robotic arm which will make total mass center higher, the weight of these four steering motors can also help balance the weight in the vertical direction. Eventually, the 4WD-4WS structure (or it can be caller steering wheels) is be decided as the chassis design.

The following parts are about my works in iterating the former omni-chassis (denoted as Type ZW-A) to the previous chassis (denoted as Type ZW-B).

Type ZW-A (shown in the Figure 2.2.1) is an old 4WD-4WS chassis designed and built by me in my junior spring semester, and it was used temporarily as the prototype chassis to assemble the robotic arm in the early stage of the project to providing a firm base for the robotic arm. Therefore, the electronic control codes writing and adjusting work for the steering wheels and robotics arm can continue simultaneously while the physical structures of iterated chassis and controller are under development.

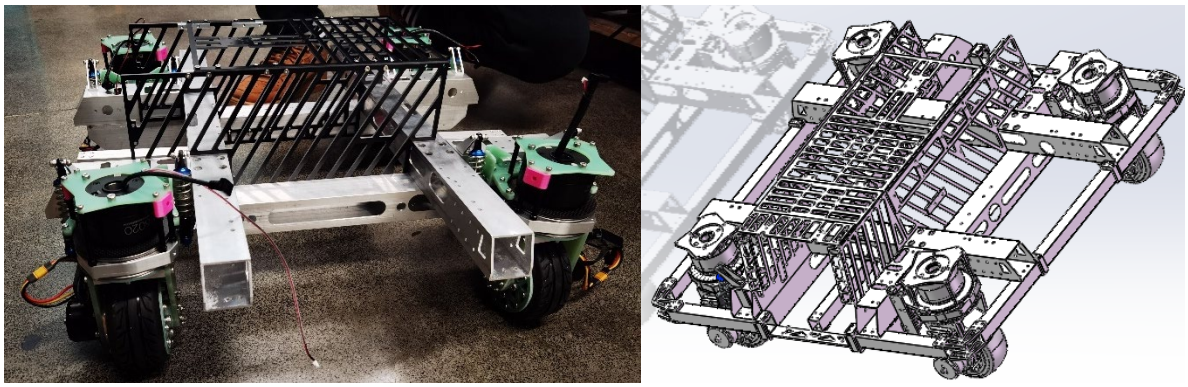

Figure 2.2.2 The iterated omni-chassis (Type ZW-A) and the model in SolidWorks

ZW-A chassis is initially designed for holding a PPP 3-axis sliding platform, whose mass center will be assembled in the same line of the center z axis of the chassis so the gravity of slide platform will be equally distributed to the chassis. The possible torsion and bending caused by the gravity of platform to chassis will be minor. However, because the horizontal rotation movement of SCARA arm, the arm requires an off-center assemble so that there is enough space for arm move to the storage system. Eventually, the moment caused by the gravity of robotic arm when it opens lead to large torsion to its assembling square tube. Also, bias between mass center of arm and chassis results in inequal compression to the spring of the steering wheels in the front side, therefore, there are obvious forward leaning of the chassis ZW-A after the arm is assembled.

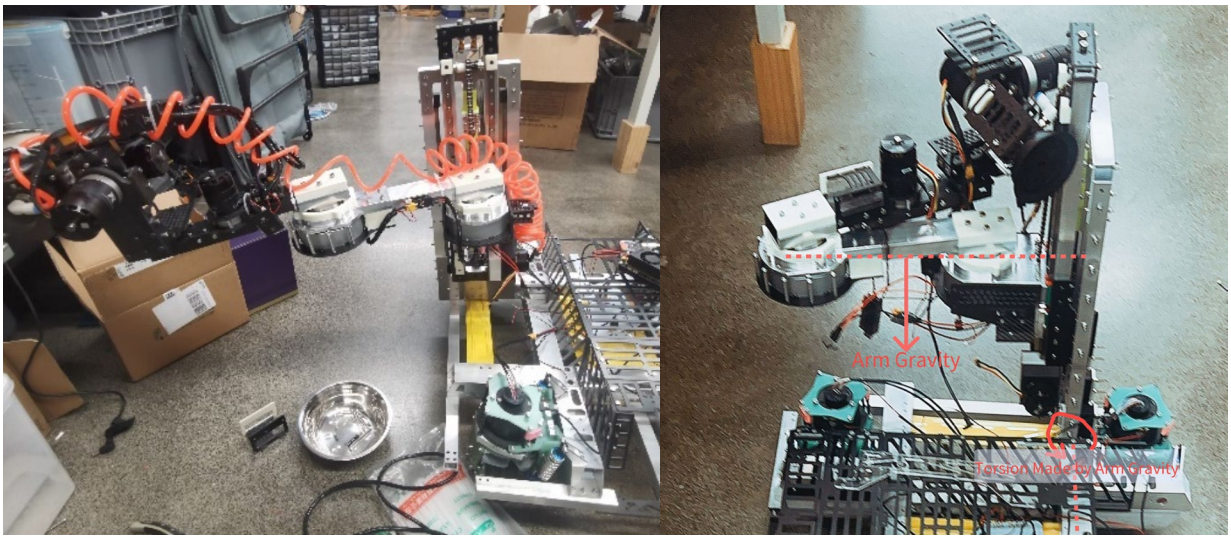

Figure 2.2.3 The forward leaning (Left) and torsion made by arm gravity (Right)

The iterated chassis ZW-B is designed to solve these two problems while integrating other subsystem into chassis. Specifically, I add a lifting camera assemble to assemble image transmission camera and industrial camera for target position detection, frame structure for conveyor belt in storage system, and frame structure for assemble mini-PC and vacuum generators.

Because of the complexity of my work in chassis, these iterations are divided into Main Frame, Steering Wheel, and Other Subsystem Assembling frame.

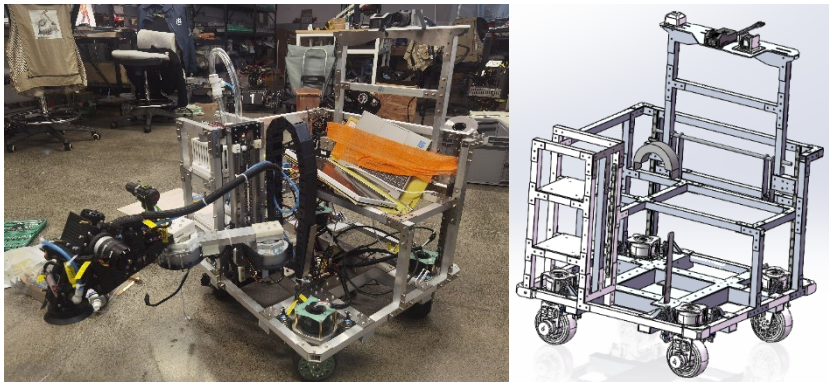

Figure 2.2.4 The iterated omni-chassis (Type ZW-B) and the model in SolidWorks

2.2.1 Main frame

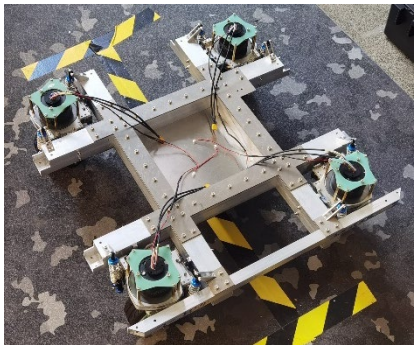

Figure 2.2.1.1 Overview of main frame and steering wheel

The main frame of old ZW-A chassis is made up with two layers of 4040*590mm square aluminum tube with wall thickness equals to 2mm. Two layers of tubes are assembled with rivet nut inside and use carbon fiber board as nut holder to enhance total strength.

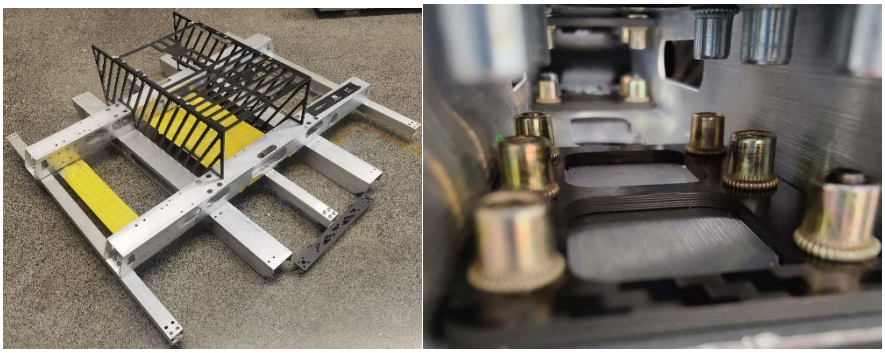

Figure 2.2.1.2 The main frame of ZW-A chassis (left) and rivet nut inside(right)

The two layers of square tube can ensure rigidness of the structure facing vertical load. However, it turns out that structure is weak when an axial torsion is added to a single tube, just as Figure 2.2.3 shows.

The new chassis ZW-B cancels the two-layer tube design, slots in the tube to create tenon joint for each tube and uses two 4mm aluminum board to compress this one-layer tubes in the middle. The static simulation shows that new design has 10 times stronger strength than old design, which will be discussed in the verification part.

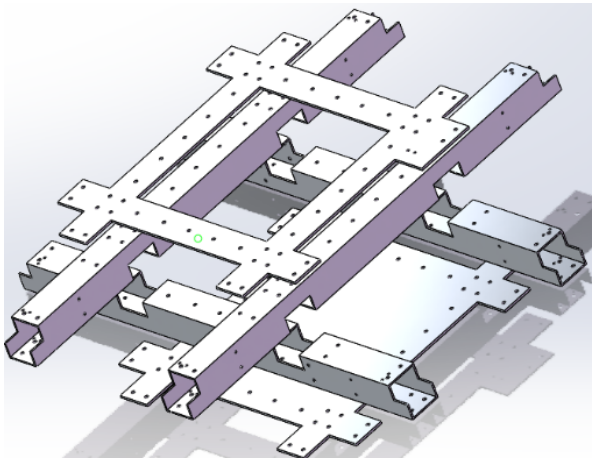

Figure 2.2.1.3 The explosion view of main frame to show its connection.

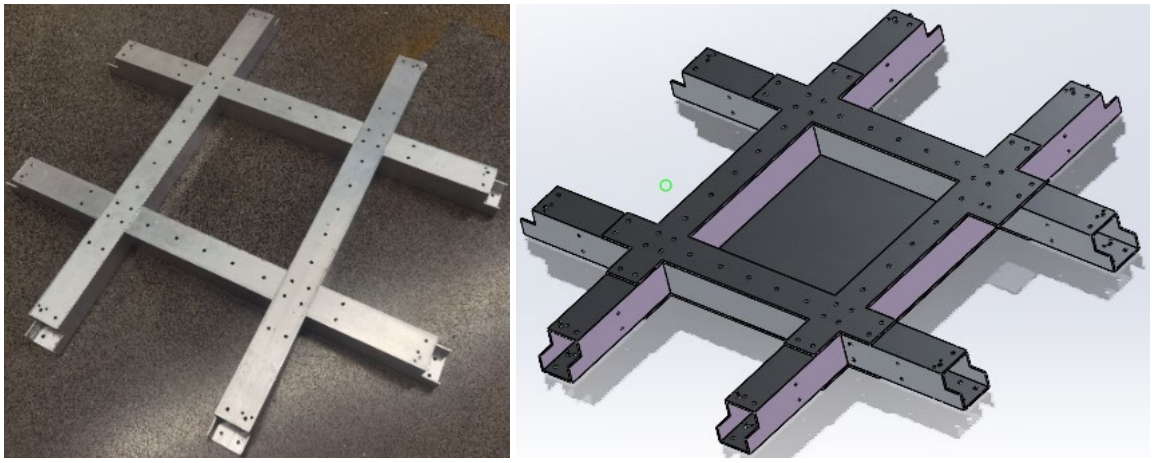

Figure 2.2.1.4 The tenon joint tubes(left) and the assemble CAD of main frame.

2.2.2 Steering Wheel

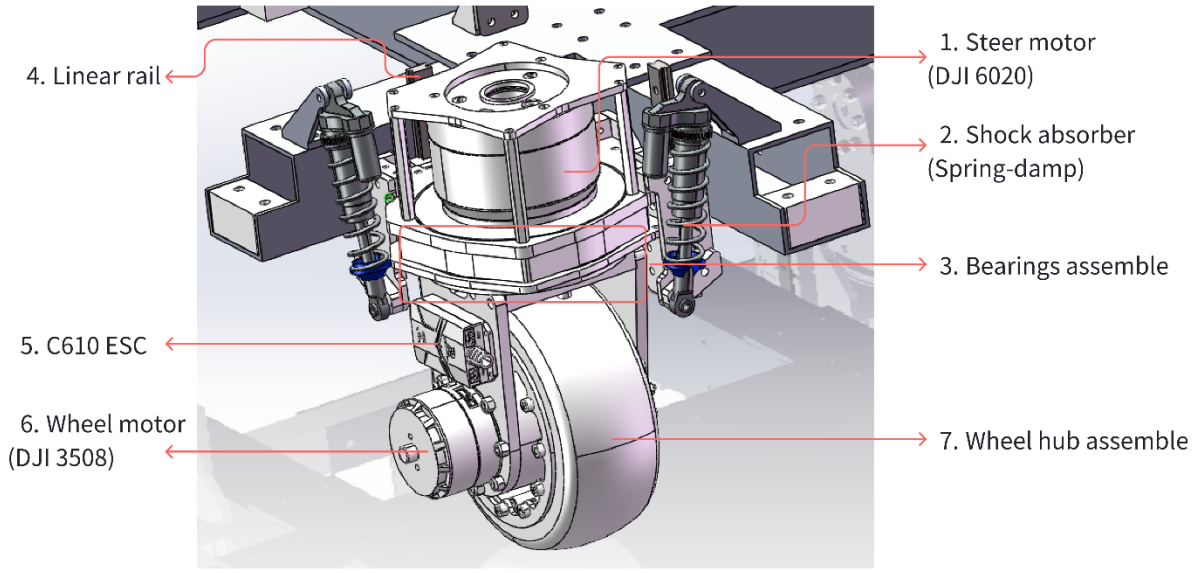

Figure 2.2.2.1 Steering wheel overview in SolidWorks

The steering wheel simply insist of a wheel with wheel motor and its ESC (electronic speed controller), a steer motor control the direction of the wheel and other connection part connect between them or with the main frame of the chassis.

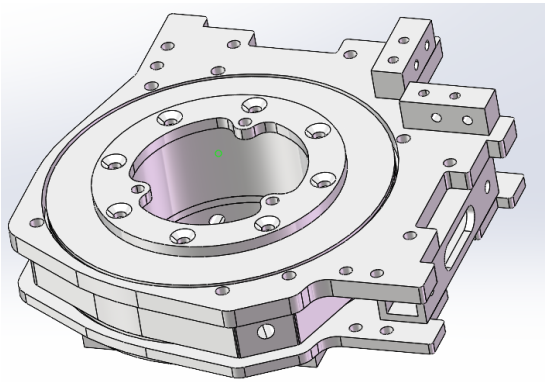

Figure 2.2.2.2 Bearing assemble between wheel and steering motor.

The bearing assemble was used thought to use a crossed roller bearing instead. To decrease cost in crossed roller bearing as well as decrease total weight, I use the structure that two flat needle roller bearings clamp a deep groove ball bearing in the middle. Therefore, the bearings assemble can also take axial and radial load, while keeping the possibility for me to change the assembly holes as I wish.

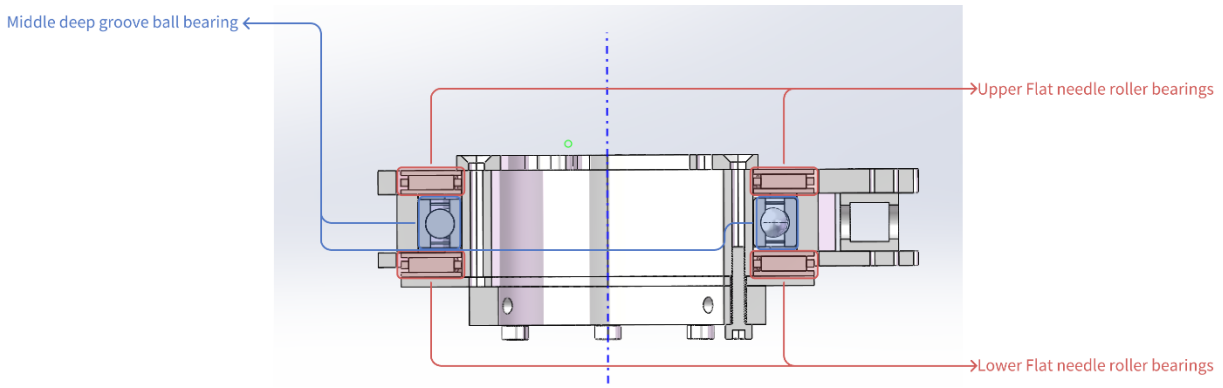

Figure 2.2.2.3 Bearings assemble cross-sectional view.

In the former version of steering wheel in ZW-A, it is found that the friction to steering motor is much larger than we expected, the friction also varies a lot for each wheel, leading difficulties in adjusting PID for each wheel. Moreover, the supportive normal force from ground and press from robot to spring-damper, generate a couple, resulting that whole steering wheel will slope outward.

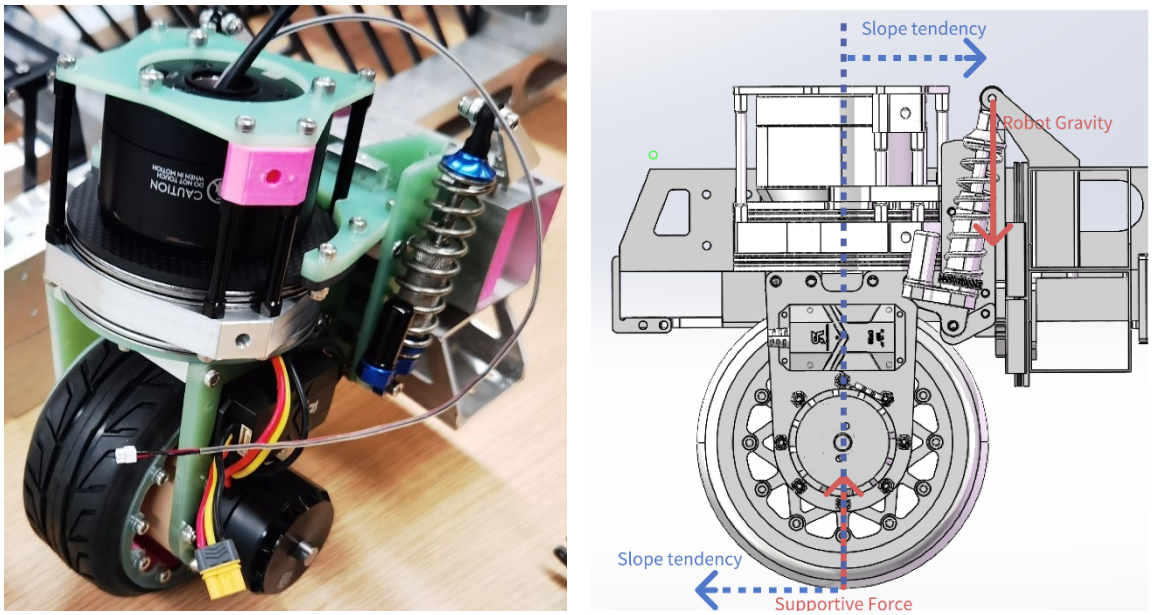

Figure 2.2.2.4 The former steering wheel in ZW-A and its slope tendency

Because there is small, unexpected empty space for parts to move even after parts being assembled. This small displacement will be sum up with the slope tendency, leading wheels to lean outwards. Meanwhile, the friction also comes from leaning, because the boards which should be coplanar will be intersect and rub with each other. Just like Figure 2.2.2.5 shows, in red circle, there is an unexpected gap after assembling. In blue circle, the slope wheel makes board rub with the shell of motor.

To minimize the gap, the floating boards shown in Figure 2.2.2.5 is extended (Figure 2.2.2.2) and add more holes to assemble them with Bearing assemble more firmly, as Figure 2.2.2.1 shows.

Figure 2.2.2.5 The reason for slope and friction

To solve the slope tendency caused by couples generated by supportive force and robot weight, one damper and one linear rail are moved to another plane that is normal to its former assembling plane. The ligature from one damper to another are designed to pass vertical center line of steering wheel, which is also the axis for supportive force acting. Therefore, shown as Figure 2.2.2.6, no coupler will be generated or only a small coupler may exist.

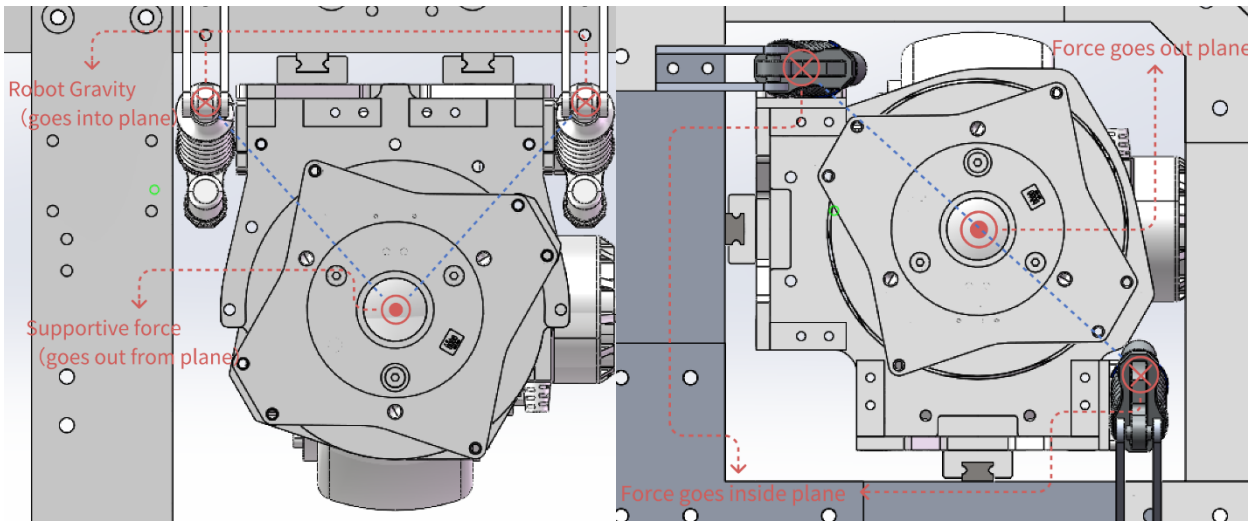

Figure 2.2.2.5 Iteration in damper and rail position (left: former ZW-A; right: present ZW-B)

It turns out that these changes are valid, the friction for each steering motor becomes small and similar, and no visual leaning of the wheel is observed whether they are static or moving.

2.2.3 Other Subsystem Assembling Frame.

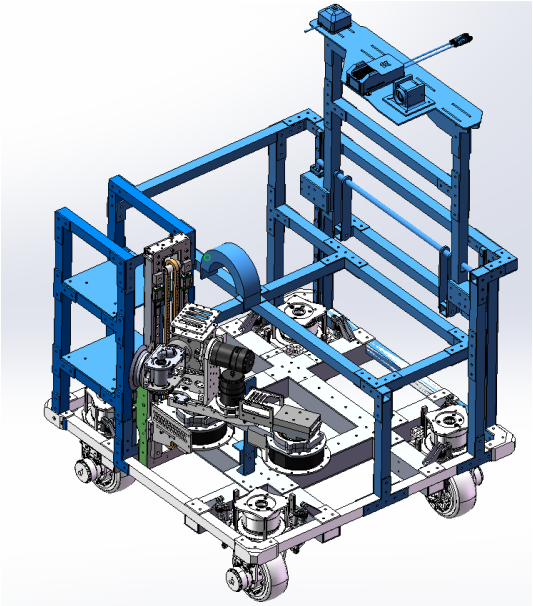

Figure 2.2.3.1 other subsystem assembling frame overview (the frame highlights in blue)

These frames are designed to place hard devices for other subsystems. For example, industrial camera for robotic vision, vacuum generators for end effectors, mini-PC for running all the control codes, conveyor belt. And the horizontal tubes connected between each vertical frames help strengthen the mechanical robustness for whole robots. Most of huge subsystems are placed to the middle or back of the chassis to balancing the weight and moment of stretching robotic arm. Hence, the spring in each wheel can be compressed as equal as possible, and the whole robot can be less oblique.

The assembling frame can be roughly divided into three parts- foldable camera bracket, mini-PC and Vacuum generator assemble tower, and unfinished frame for conveyor belt as object storage space.

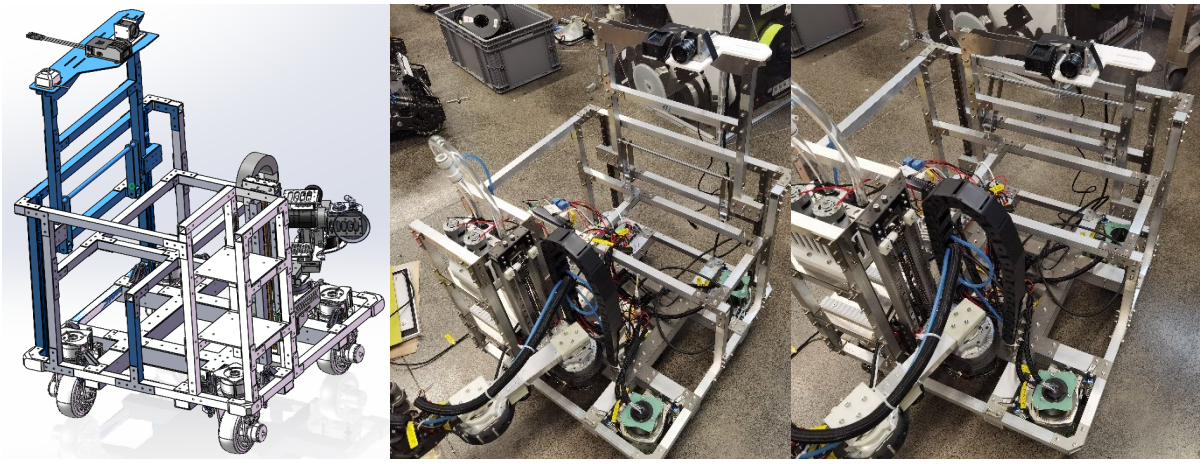

Figure 2.2.3.2 foldable bracket for camera (the frame highlights in blue)

Figure 2.2.3.2 shows the 2 different stages of the frame, the middle image shows the bracket is lifted so that the industrial camera has larger field of vision to recognize the target position, and operator can also see the movement of robotic arm from image transmission.

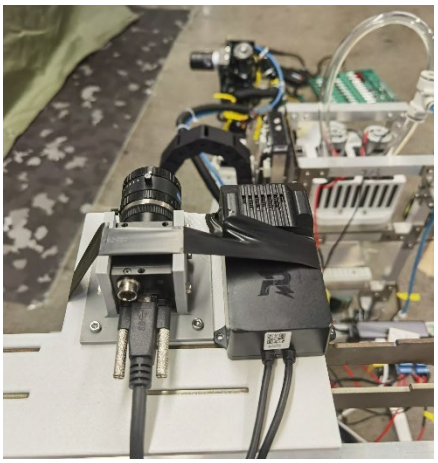

Figure 2.2.3.3 Left: MindVision Industrial camera; Right: DJI UAV image transmission

The right image of Figure 2.2.3.2 shows the folded stage of the frame, at this case, whole size of the robot is limited to 590*590*590 (unit: mm). This size can be put into a 600*600*600 flight case when the robot needs long journey transportation.

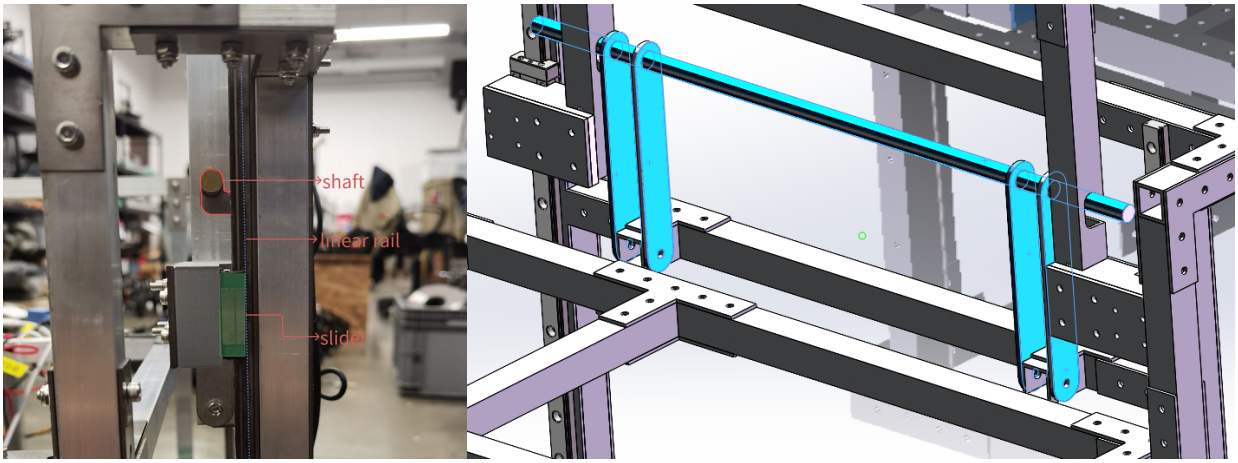

Figure 2.2.3.4 Shaped hole on tube to stuck shaft after rotating.

The frame can be lifted along two linear rails. And the shaped hole on the square tube can let the shaft slide in and stuck it with frame self-mass when the frame drops down a bit. With this foldable design, the camera can be lifted from ground for 800mm, and its vision will not be block by any frame on robot.

The assemble tower (Figure 2.2.3.5) is used for placing two vacuum generators and mini-PC. The generators are simply fixed to the tower with a PLA 3D printing shell. At same time, the tower connects to the backside of the z-axis (Link 1) of robotic arm, providing tensile against the moment of stretched arm.

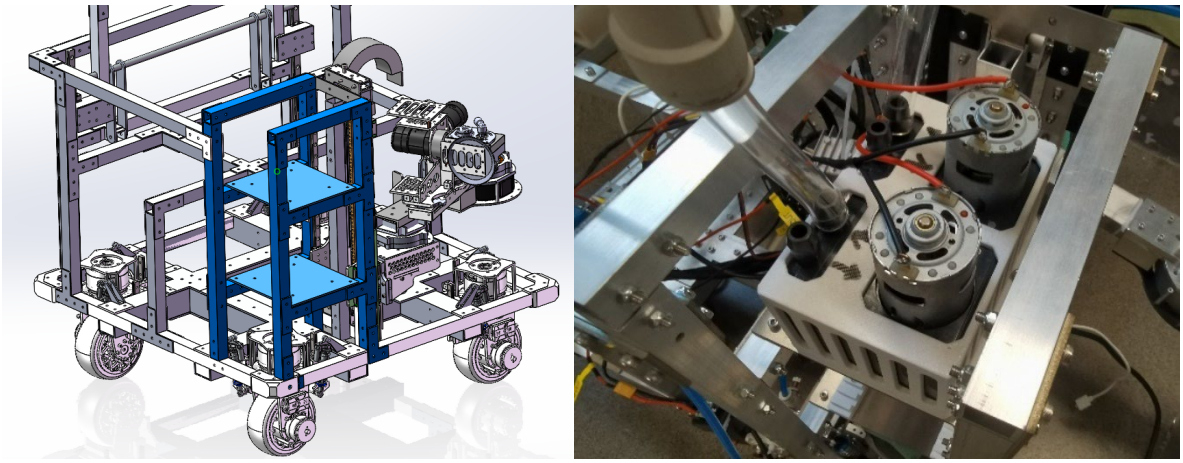

Figure 2.2.3.5 the assemble tower (the frame highlights in blue) and two vacuum generators.

Since the control and vision program requires lots of adjustments, it is necessary to make sure that the mini-PC can be removed from robot easily while fixed with tower firmly even when the robot is moving. Therefore, two 3D printing picatinny rails are used for setting up the mini-PC. The mini-PC can be quickly fixed or removed from rails by tightening or unscrewing four hand screws without requiring any tools. Because all the nuts have buried inside slider when they were printing.

Figure 2.2.3.6 The picatinny rails(left), the mini-PC with sliders(middle) and the hand screw(right).

The left frames are used to install a conveyor belt so the robotic arm can fetch or put object it gets, and the belt will transport the objects into or out of the storage space as needed. The space can store more than 400*200*200 (unit: mm) volume of objects. And the mass of the belt and its driving motors can significantly balance the mass center position of whole robot.

However, it is pity that the delivery of conveyor belt system is delayed and teammates for coding robot control program had not enough time to integrate such subsystem into whole robot.

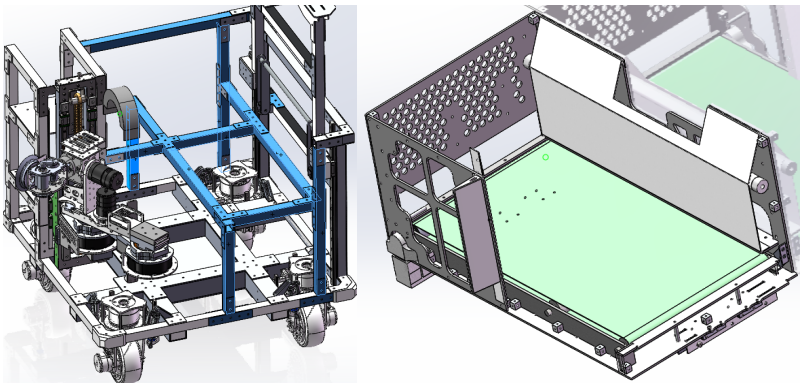

Figure 2.2.3.6 Frames for fixing conveyor belt (the frame highlights in blue) and storage space with belt.

2.3 Power supply

2.3.1 Overview

The Power Supply (PS) system is crucial for the operational efficacy of the movable robotic arm, featuring a simplistic yet highly efficient design composed of a battery and a power control board. The system's primary component, the battery, ensures a steady supply of 22.2V DC power. This pivotal system not only energizes the robotic arm but also seamlessly integrates with and powers the three other major subsystems: the Robotic Arm system, the Control System, and the Omni-Chassis system, guaranteeing a continuous 22.2V DC power supply to these systems.

2.3.2 Requirement

Central to the PS system are two key components: a 22.2V battery (DJI TB47D) and a Main Power Control board. The Main Power Control board functions as the intermediary between the battery and the subsystems, facilitating the distribution of power necessary for their uninterrupted function.

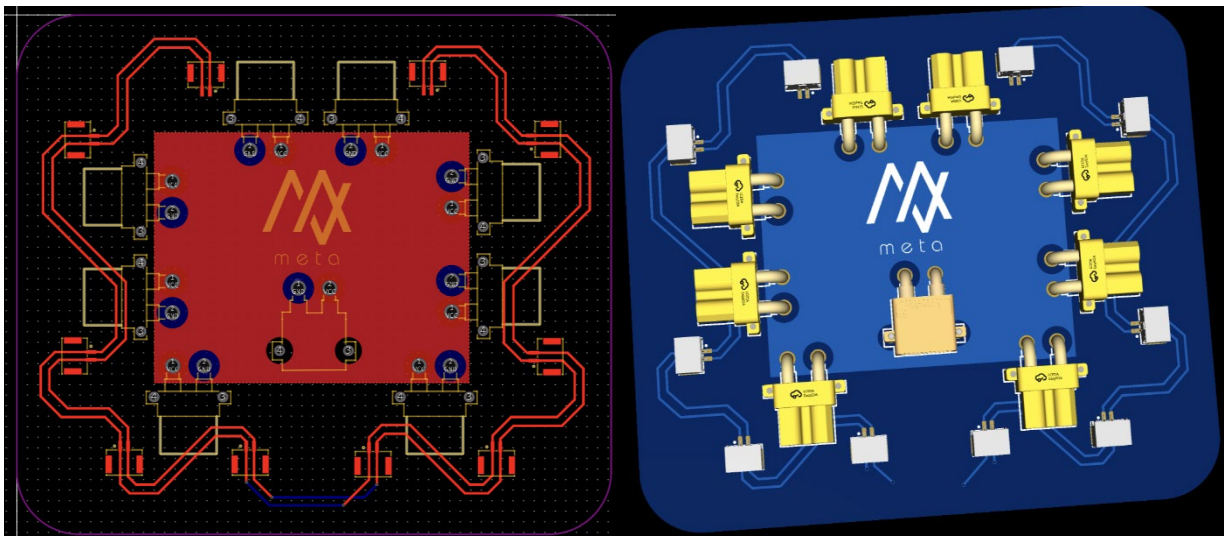

Figure 2.3.2.1 The main power control PCB board

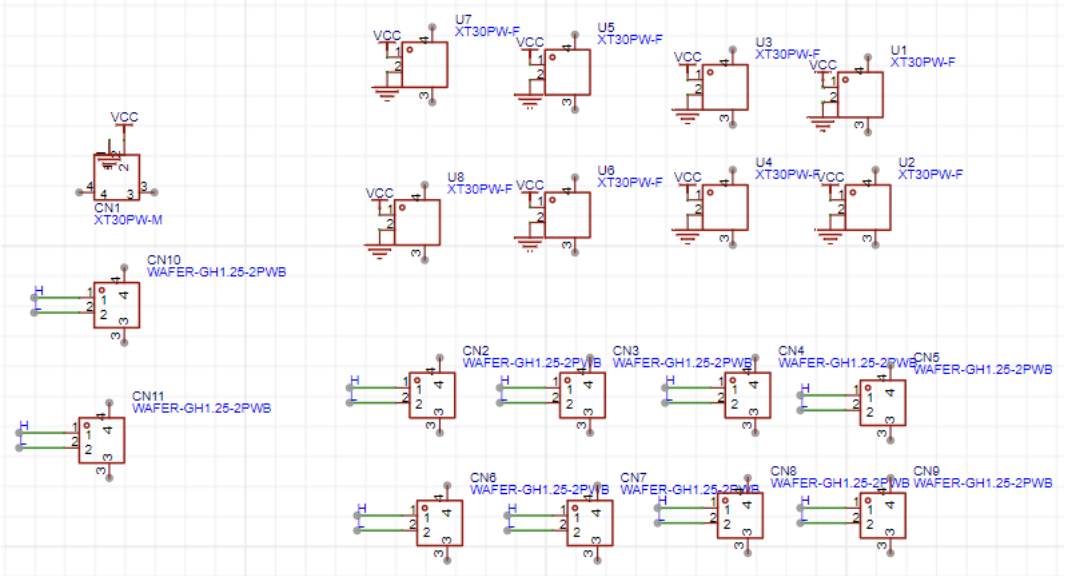

Figure 2.3.2.2 PCB schematic

The power supply subsystem also contains the electronic circuit, which can be divided into power distribution and signal serial. CAN communication method is used as signal connection. After test, CAN communication with 250 Hz can transmit most 8 motors in series. Therefore, two ways of CAN communication are used, one is for the six motors in robotic arm, another one connects eight motors of steering wheels in the chassis. Finally, these two ways of CAN wires are connected to a USB-CAN analyzer to transmit the instruction with USB serial.

Figure 2.3.2.3 USB-CAN analyzer (left) and USB relay (right)

The vacuum generators are controlled with a USB relay to make the suction cap normally closed and operating only when controller give instruction.

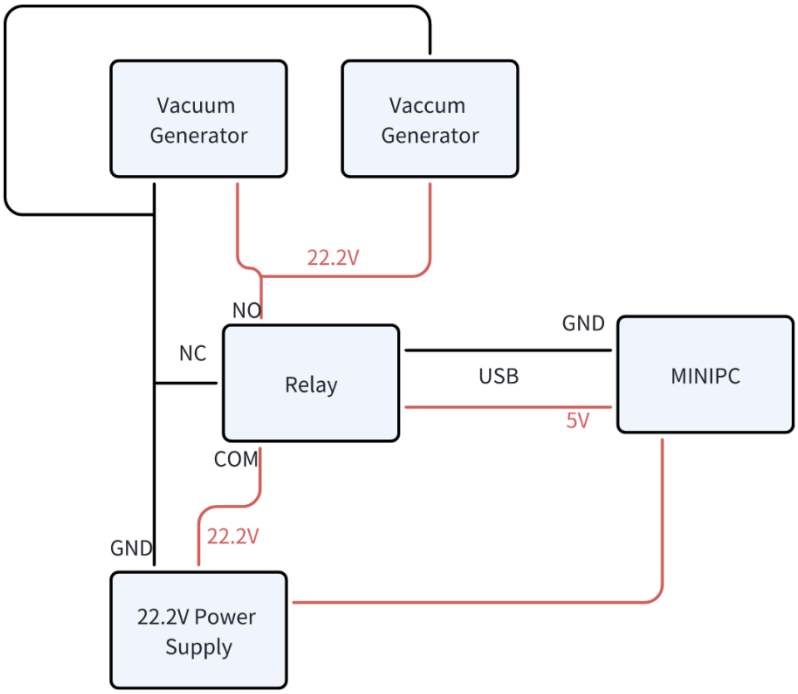

Figure 2.3.2.4 Detail connection graph for vacuum generator and relay.

The power supply is used with a TB47D battery with 22.2V DC out. To make mini-PC working stably, a DC24V to DC19V8A transformer is used.

Figure 2.4.2.5 Power supply and DC transformer.

2.3.3 Tolerance Analysis

The PS system is engineered to support a variety of critical components to ensure the robotic arm's high performance and reliability. It must efficiently power at least 3 STM32F04 chips located in the handle, each paired with an acceleration sensor for enhanced precision and control. Additionally, the system powers a main control chip (STM32F04) that orchestrates the operation of the arm, and three servos on the robotic arm side, crucial for the arm's movement and functionality. This requirement highlights the PS system's integral role in sustaining the arm's responsiveness and operational integrity across a range of tasks.

2.4 Control

2.4.1 Overview

The main function of the control system is to control the action of the robotic arm. The system recognizes the hand movement of the manipulator through the and then controls the robotic arm to simulate the hand movement.

The code for the control system is detailed in the Appendix section.

The control system consists of two subsystems: On-Chassis Control System and Remote System.

2.4.2 Remote System

The robotic arm has six degrees of freedom, which is toughing if controlling robotic arm with classical rockers set. The human's arm also has degrees of freedom equals to 6, therefore, it is possible and reasonable to map the translation and rotation movement of human hand to the robotic arm. Meanwhile, we have finished the inverse kinematic to the robotic arm. Therefore, the aim for the controller is to collecting data of operator's arm movement and send the position information of wrist as well as the Euler angle of operator's hand after calculation. Then we can combine the position and angle information as a transform matrix T. Matrix T can be map as the end position and orientation of the end effector of the robotic arm. Hence, we do not need to control the robotic arm joint by joint, which will be more intuitive and easier to operate.

2.4.2.1 Controller Operating Principles

The human's arm can be simplified as Figure 2.4.2.1.1 The original point of base coordinate S_0 refers to the operator's shoulder. Each joint can be view as omni-directional joint with DOF=3. The big arm is simplified as link1 with length L_1, the forearm is the link2 with length L_2. The aim for calculation is to find the position of J_3 in base coordinate S_0, denote as P_3. With three 3 axis gyroscopes fixing with hand, forearm, and big arm respectively, we can know the orientation for each link as three Euler angles, which can be integrated from three axis angular velocity measured by gyroscopes, denote a set of Euler angles as ψ_i, θ_i, φ_i (angles rotate around x, y, z axis respectively), i =1, 2, 3.

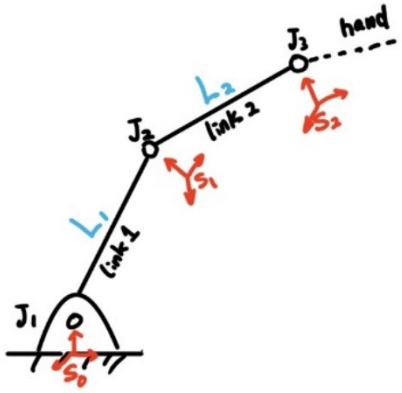

Figure 2.4.2.1.1 Model of a human's arm

For each link, its rotational matrix R_i (to coordinate S_0) can be.

$$R(\psi,\theta,\varphi)=\begin{pmatrix}\cos\psi\cos\varphi-\sin\psi\sin\varphi\cos\theta & \sin\psi\cos\theta+\cos\psi\sin\varphi\cos\theta & \sin\varphi\sin\theta\\ -\cos\psi\sin\varphi-\sin\psi\cos\varphi\cos\theta & -\sin\psi\sin\varphi+\cos\psi\cos\theta & \cos\varphi\sin\theta\\ \sin\psi\sin\theta & -\cos\psi\sin\theta & \cos\theta\end{pmatrix}$$

Then the transformation matrix T_i for each link can be calculated from R_i

$$T_i=\begin{pmatrix}R_i & P_i\\ 0 & 1\end{pmatrix}, T_i\in R^4$$

For link1 and link2, the translation matrix P_i (in the coordinate S_{i-1}) can be generated from length L_i

$$P_i=\begin{pmatrix}L_i\\ 0\\ 0\end{pmatrix}$$

Then the position of J_3 in coordinate S_0 can be expressed as

$$P_{30}=\overrightarrow{J_1J_2}+\overrightarrow{J_2J_3}=R_1P_1+R_2P_2$$

Here we get the position of twist (J_3) with original point in shoulder.

With Euler angles $\psi_3,\theta_3,\varphi_3$ of hand, we can get its rotation matrix R_3 and transformation matrix T_3

The transformation T_3 can be send to mini-PC in the robot to map as the end position of the robotic arm. Then the inverse kinematics will give out each joints' angle automatically.

2.4.2.2 Controller Physical Structures

Figure 2.4.2.2.1 the overview of motion controller

The motion controller is made up with three RoboMaster Development Board Type C, which integrates a STM32F04 chip and a six axis IMU BMI 088 inside.

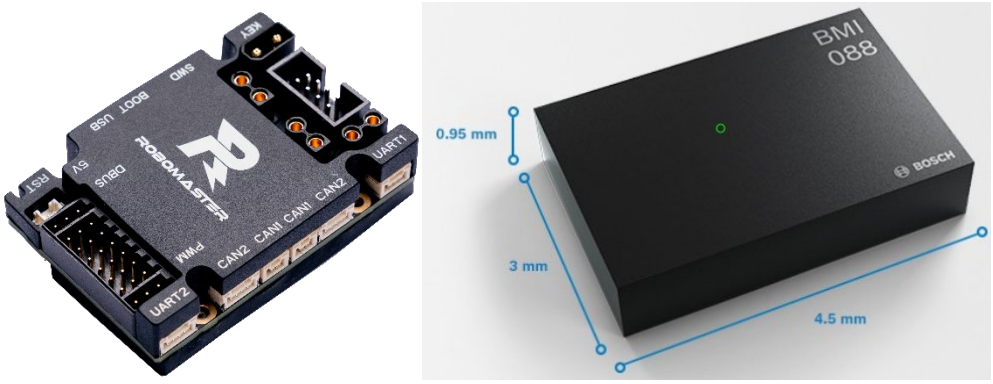

Figure 2.4.2.2.2 develop board type C and Bosch BMI 088 IMU

Initially, there are three schemes that I am firstly collected after reviewing some of relative reading materials.

First scheme is using three gyroscopes to get three sets of Euler angles for three links whose lengths have been known. Then we can calculate the end position and orientation. Our project finally chooses this idea.

Second scheme is using Intel T265 camera to obtain 6DOF orientation of camera. Since T265 can use V-SLAM algorithm in Movidius Myriad 2 VPU directly. Basically, is a machine vision project. This project can be easiest and request least to mechanical design. However, T265 is too expensive comparing three STM32F04 and BMI 088 IMU. Therefore, this idea is finally denied.

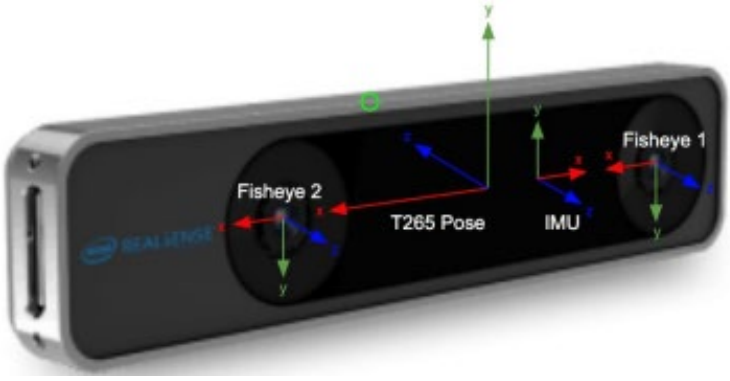

Figure 2.4.2.2.3 Intel T265 camera

Third scheme is using a six axis IMU to obtain the 6DOF orientation of end. We have done some test about this idea by directly using one BMI 088 in hand. However, two problems come out. One is that the x, y, z value of the position is integrated from the linear acceleration read by IMU, the double integral of acceleration results in unacceptable biasing speed of the position value. Another one problem is more severe that the linear acceleration sensor of BMI 088 cannot work while we lean the IMU that changes its orientation in the space, which means the 3-axis gyroscope and 3-axis force acceleration meter cannot work at same time in the IMU.

There are more schemes, like a delta structure controller, building a 6-axis arm with 6 encoders. But eventually, we decide the idea of using three gyroscopes with given length.

Initially, I decided to make a 6DOF bracket (like the table lamp of Pixar) and fix the IMU on the links. Just like Figure 2.4.2.2.4

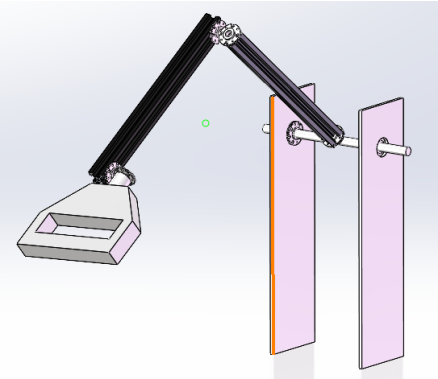

Figure 2.4.2.2.4 Initial draft of the controller

Then I built a real prototype of the controller by 3D printing the links, then I found that the maneuverability of the prototype is bad. Because the rotation axis for the final three joints is not intersect and has similar rotation radius, resulting some orientation being hard or impossible to reach.

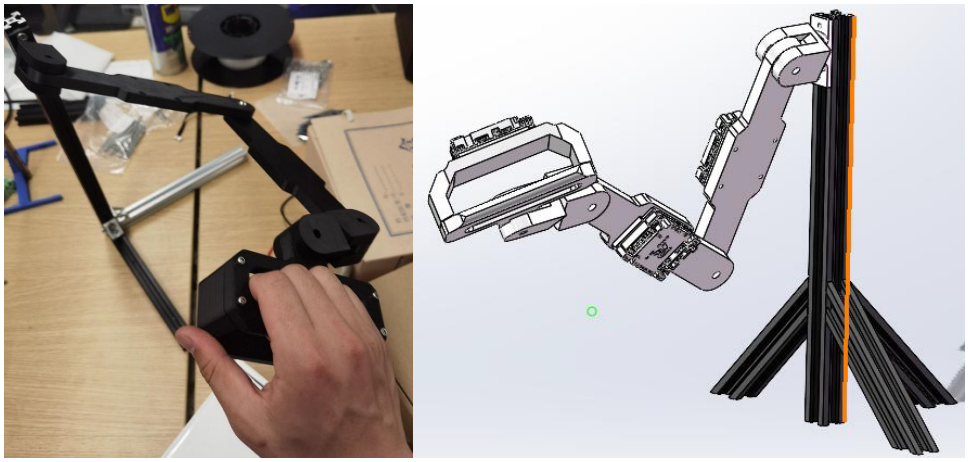

Figure 2.4.2.2.5 The first prototype of controller.

Finally, after considering the process that operator use motion controller. We found that L_1 and L_2 length of the controller links as well as the end point position can be no precise. What we send to robot, is the difference between end point position so the robotic arm will know what direction move next rather than just moving to a given position. And operator can watch from image transmission to decide whether to stop moving his hand or change moving direction. Therefore, if L_1 and L_2 is not precisely fit with real physical situations, the difference between point position is still accurate by mapping with scaling or shifting the values.

Hence, the simplest way is to bind three IMUs in human's arm. I design three flexible bases for IMU, which are printed with TPU A95 and use Velcro belt to bind on hand and arm.

Figure 2.4.2.2.6 The design for a single IMU base.

2.4.3 On-Chassis Control System

On-Chassis Control is mainly responsible for receiving signals from the Remote system and exercising control over the control robot arm.

The system also has a subsystem computer vision component.

The computer vision system consists with a Logitech C270i HD webcam camera and a x86 NUC mini-PC. The camera is mounted on the chassis and connected to the mini-PC on the car. The camera captures images of the workspace and sends them to the mini-PC. The mini-PC, running Ubuntu 22.04, should processes the images and extract the location and orientation of the box, solve the inverse kinematics, and pass the control signal to the robotic arm to move the box to the desired location.

The computer vision system should be able to: detect the box (location) in the image and recognized the orientation of the box in the image.

The computer vision system is implemented using the craves framework for robotic arm controlling and the YOLO model for object detection and recognition. For the orientation of the box, we plan to use 6Do-f pose estimation and PnP algorithm.

3. Design Verification

The most important and the process that took us the longest time in this loop section was the PID parameter adjusting and the redesign of the chassis.

3.1 The Adjustment to PID parameters

3.1.1 Problems and Difficulties

The adjusting process was very long and there were many difficulties.

When the PID parameters are too small, the test phenomenon is that the feedback is too slow. In this case I need to use the control variable method to increase the six PID parameters in turn to test whether the slow feedback is caused by the parameter being too small. Debugging six parameters is time-consuming, not to mention the fact that we have so many motors (two motors for each of the four wheels of the chassis plus the robotic arm has seven jointed motors). And every time we adjust the parameter, we can only improve a little bit (less than 10%), because if the PID parameter is too large, the test phenomenon is not only very fast feedback, but also accompanied by motor oscillation, and in serious cases, it will cause the motor to lose control of the motor and start spinning wildly and other problems, which is very dangerous.

When I was testing the 3508 motors in the fourth joint of the arm, I accidentally adjusted the KP parameter of the speed to a very large size, and when I turned on the power, the whole fourth joint started to rotate at a fast speed, and the whole joints before the fourth joint started to move around. I almost got hurt because of this, but luckily my teammate who was testing with me quickly turned off the power. But in the end, the trachea of the robotic arm was still hurt. From then on, I was careful with the debugging and spent a lot of time with each motor.

At the same time, the adjusting of PID parameters is easily affected by unknown variables. For example, the left rear wheel of the chassis could not achieve a feedback speed to my satisfaction in any case, but the other three wheels had completed their tests. I spent nearly an hour tuning the left rear wheel to improve its feedback speed, but it still didn't work. Finally, Mr. Zhizhan, who is in charge of the mechanics, found that one of the screws on the left rear wheel had fallen out, which caused the obstruction.

Anyway, debugging the PID parameters is a long process that requires us to keep working on it.

3.1.2 Test and Verification

In the process of verifying the PID parameters, we need to test each motor individually. So we wrote a TEST code for individual clicks separately, as shown in Figure 3.1, which allows us to test them individually by simply entering the number of the corresponding motor.

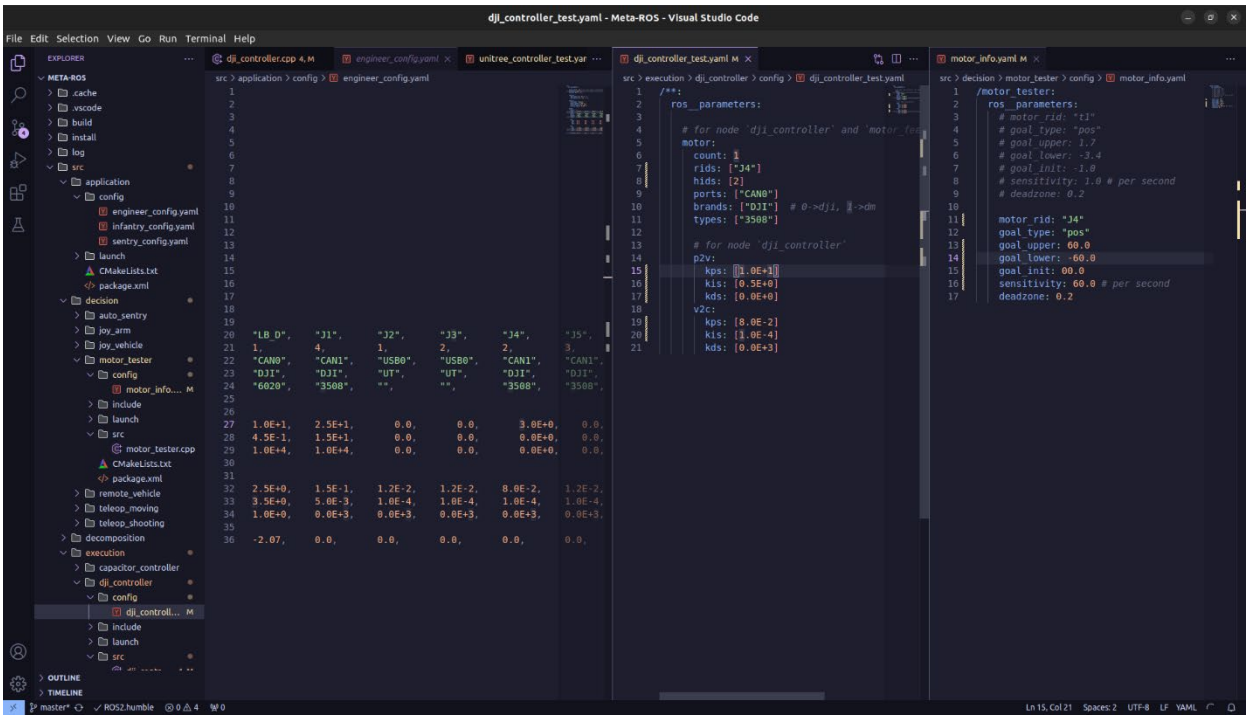

Figure 3.1 Test Code for PID

To observe the effect of the PID parameters more intuitively, we use PlotJuggler plotting to show the motion state of the motor. From Figure3.2 we can see that after connecting the output signal of the motor and the input signal of MINI_PC, PlotJuggler can plot and compare the target position of the motor with the actual position. From the comparison plot of the two we can clearly know the error and thus the correctness of the PID parameters.

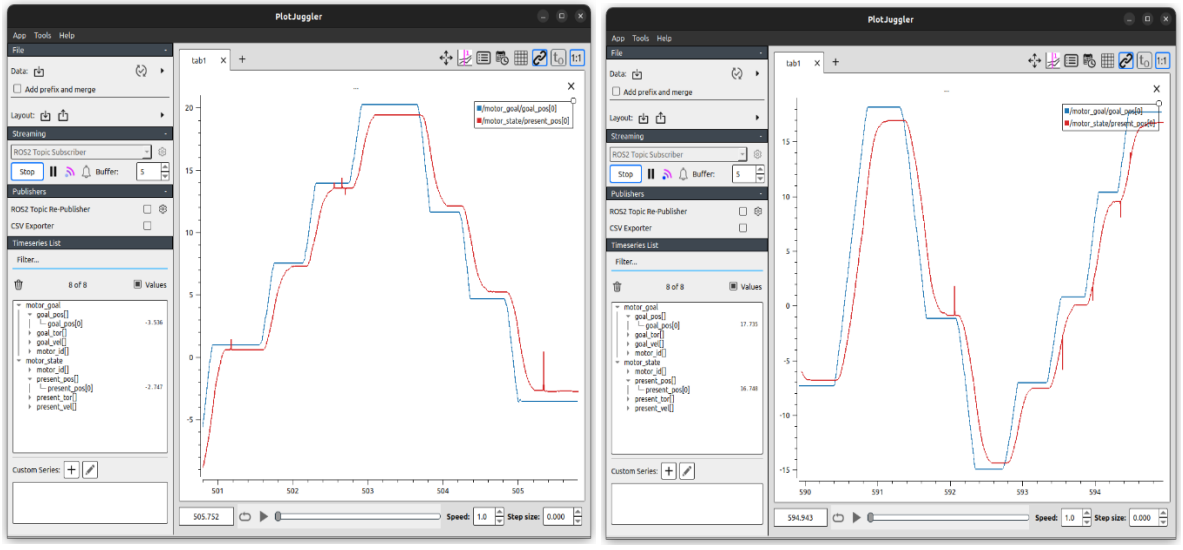

Figure 3.2 Plot Juggler

In conjunction with the actual operation of the motors and the PlotJuggler's schematic, we completed the commissioning of the PID parameters for each motor individually. After all the motors have been parameterized, we run all the motors together at the same time, and we find that some of the motors are not running perfectly, and we make corresponding adjustments for this situation. Finally, under the unremitting efforts, the debugging task of PID parameters came to an end.

3.2 The Redesign of Omni Chassis

The following part shows the SolidWorks simulation outcome comparison for old and iterated frame, in facing vertical load and axial torsion to a single tube separately. Table 3.2.1 shows the material properties of the aluminum 6061 alloy used for simulation.

Table 3.2.1 6061 alloy properties used for simulation.

Properties	Unit (SI)	Value
Elastic Modulus	N/m^2	6.9e+10
Poisson's ratio	1	0.33
Shear modulus	N/m^2	2.6e+10
Tensile strength	N/m^2	124, 084, 000
Mass density	N/m^2	2700
Yield strength	N/m^2	55, 148, 500

In first vertical load simulation, 1000N (safety factor = 2) load is added vertically and normally to the upper face of two frame to test the frame rigid when another subsystem was assembled to them.

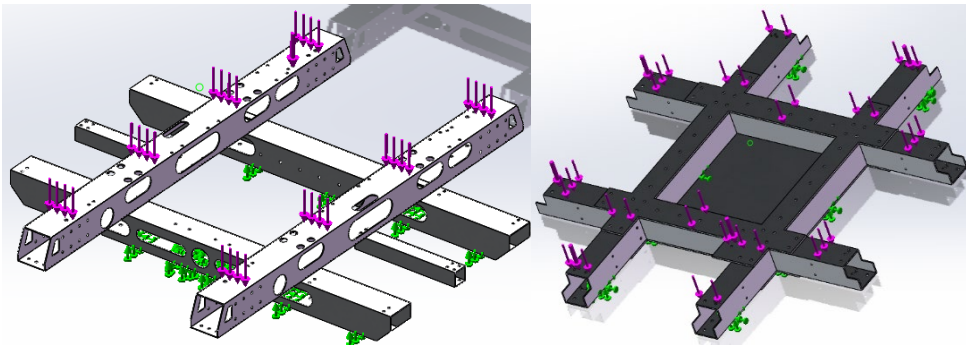

Figure 3.2.1 ways the vertical load is added (left: old chassis; right: present chassis)

In second axial torsion simulation, 10Nm (safety factor = 3) axial torque is added along a single tube to imitate the situation when the robotic arm is fully stretched.

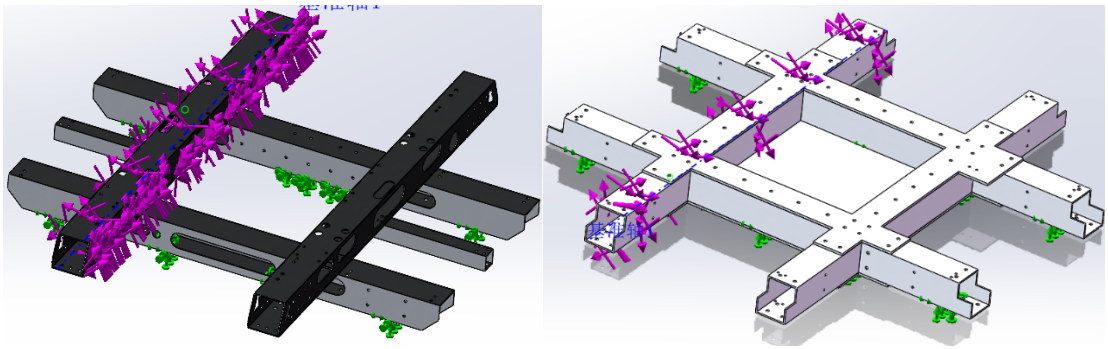

Figure 3.2.2 ways the axial torque is added (left: old chassis; right: present chassis)

The comparison of the vertical load outcome shows in Table3.2.2 and comparison of axial load outcome shows Table 3.2.3.

Decrease ratio (r) is calculated by.

$$r = \frac{x_f - x_p}{x_f} * 100\%$$

Where x_f and x_p mean the outcome value comes from former design and present design respectively.

Table 3.2.2 vertical Load outcome comparison

Properties	Unit (SI)	Former	Present	Decrease ratio (100%)
Max Stress	N/m^2	4.061e+7	1.624e+6	96
Max Displacement	mm	1.756e-1	5.943e-3	96.6
Max strain	1	4.153e-4	1.302e-5	96.9

Table 3.2.3 axial torsion outcome comparison

Properties	Unit (SI)	Former	Present	Decrease ratio (100%)
Max Stress	N/m^2	2.704e+7	1.179e+6	95.6
Max Displacement	mm	7.985e-1	3.465e-3	99.6
Max strain	1	2.478e-4	1.255e-5	94.9

The outcome of simulation shows that the new design of main frame in ZW-B have close 10-time strength to former main frame design, either in vertical load or axial torsion. The real test results in assemble robot also verified the iteration successfully, there is only 3 degrees leaning forward (former is 10) and the leaning towards chassis mass center which is caused by torsion is too small to measure (former is about 5 degree).

4. Costs

4.1 Non-standard Parts and Equipment

Our project incorporates a blend of standard and non-standard components, the latter of which include:

•**Laptops**: Each team member is equipped with a personal laptop, pivotal for programming, design, and communication throughout the project. These laptops are pre-existing resources and, thus, not factored into our project costs.

•**Batteries and Power Lines**: For powering our prototypes, we utilize batteries and power lines graciously provided by our lab. Their cost is absorbed by the lab's resources, emphasizing the supportive infrastructure of our educational institution.

4.2 Labor Costs

Zhuohao Xu:

Table 4.1 Labor cost for Zhuohao

Initial Setup and Configuration	**Software Installation**: Zhuohao Xu begins by installing necessary software tools such as STM32 CubeMX, which facilitates the creation of a programming framework specific to the STM32F407 microcontrollers. This software aids in configuring the microcontrollers' hardware features and in initializing the project with the appropriate middleware and libraries. **Development Environment Setup**: Xu selects CLion as the Integrated Development Environment (IDE) for programming and flashing the STM32 microcontrollers. The choice of CLion is strategic, offering robust features for code development, debugging, and management, ensuring an efficient workflow. **Environment Configuration**: The process involves meticulous setup and configuration of the development environment, including the installation of necessary drivers, toolchains, and utilities for microcontroller programming. Understanding the STM32 programming architecture and tailoring it to the specific needs of the robotic arm project forms the foundation of this phase.
Programming and Optimization	**Control Input Handling**: With the setup in place, Xu's primary responsibility is programming the STM32F407 microcontrollers to handle input from the control handle accurately. This involves writing code that efficiently processes input signals, translating them into movements by the robotic arm with minimal delay. **Responsive Movement**: Implementing sophisticated sensing and control systems to detect and respond to handle movements promptly, ensuring the robotic arm accurately follows the movements of the handle without unnecessary delays or unintended movements. **Directional Accuracy**: Developing complex algorithms that interpret the handle's movements and translate them into precise motor actions in the robotic arm. This ensures that the direction of the handle's movement is consistently mirrored by the robotic arm.
Calibration and Mapping	**Proportional Scaling**: Xu must ensure the movements from the input handle to the robotic arm are scaled proportionally. This involves calibrating the system to

	maintain a consistent ratio of movement between the handle and the robotic arm, accommodating different operation modes or payloads as necessary. **Minimized Delay**: Employing optimization strategies to minimize system latency. This includes algorithmic optimizations, adjusting parameters for efficiency, selecting more efficient communication protocols, and distributing the computational load effectively across the system components.
Auxiliary Control Integration	**Force Feedback**: Integrating force feedback mechanisms, potentially informed by visual recognition systems, to provide tactile feedback to the user. This requires programming the microcontrollers to process feedback from the environment and adjust the control signals accordingly. **Calibration Mechanisms**: Implementing and programming periodic calibration mechanisms to refine the system's understanding of the robotic arm's position and movements. This is crucial for maintaining ongoing accuracy and reliability in movement reflection and mapping.
Collaboration and Coordination	Throughout these tasks, Zhuohao Xu must coordinate closely with the rest of the project team, ensuring that the control input handle's programming and optimization are aligned with the overall goals of the “Movement Reflection” project. This includes collaborating on selecting and adjusting communication protocols to minimize latency and optimizing integrators based on real-world performance and feedback.

In summary, Zhuohao Xu's role is multifaceted, requiring a deep understanding of microcontroller programming, control theory, and system optimization. Through meticulous programming, calibration, and optimization of the STM32F407 microcontrollers, Xu plays a critical role in achieving a high degree of fidelity between the movement of the input handle and the corresponding action of the robotic arm, ensuring responsive, accurate, and intuitive control for a wide range of precision tasks.

Zhizhan Li:

Table 4.2 Labor cost for Zhizhan

Overall Design and System Architecture	Zhizhan Li is tasked with creating the overall flowchart and design of the project. As the only mechanical engineering major, his profound understanding of mechanical frameworks and structures enables him to oversee the project from a high-level perspective, ensuring that all mechanical components work in harmony to achieve the project's goals. His responsibilities include: **Designing the Mechanical Architecture**: Crafting a comprehensive blueprint that outlines the robotic arm's mechanical structure, ensuring it aligns with the project's objectives of responsive and accurate movement. **Integration with Control Systems**: Collaborating closely with the team responsible for the arm's sensing and control systems to ensure seamless integration. His role is crucial in ensuring that the mechanical design supports the sophisticated algorithms required for directional accuracy and responsive movement.
Component Selection for	Given the project's emphasis on responsive movement, directional accuracy, and mapping accuracy, Zhizhan Li must meticulously select components that meet the demands of real-world application. This involves:

Real-world Application	**Choosing Appropriate Materials and Parts**: Selecting materials that offer the necessary strength and flexibility, while also considering weight, as it impacts the system's responsiveness and accuracy. **Custom Component Design**: Designing custom parts where off-the-shelf components do not meet the specific needs of the project, such as specific actuators or joints that provide the precise degree of movement and force feedback required.
Mechanical Stability and Force Analysis	To ensure the robotic arm performs with minimized delay and maximum fidelity, Zhizhan Li must conduct a thorough analysis of the system's mechanical stability and force dynamics. This includes: **Developing Mechanical Models**: Creating models that simulate the robotic arm's behavior under various loads and movements to predict and enhance its stability and performance. **Force Feedback Integration**: Designing mechanisms that can accurately convey tactile feedback through the control handle, allowing for nuanced control and interaction with different environments. This involves understanding the mechanical implications of integrating such systems and ensuring they do not compromise the arm's responsiveness or accuracy.
Optimization for Real-time Performance	In alignment with the project's goal to minimize delay, Zhizhan Li's expertise will also be leveraged to: **Material and Design Optimization**: Selecting materials and designing components that minimize latency and inertia, thereby enhancing the system's real-time performance. **Collaboration on Optimization Strategies**: Working closely with the team to implement hardware optimizations that reduce system delay, ensuring the robotic arm's movements are as close to real-time as possible.
Support for Auxiliary Controls	Zhizhan Li's role extends to integrating auxiliary controls into the mechanical design, facilitating enhanced control and functionality: **Designing for Calibration**: Incorporating mechanisms that allow for easy calibration of the system, ensuring long-term accuracy and reliability of the robotic arm's movements. **Accommodating Force Feedback Mechanisms**: Ensuring the design supports the integration of force feedback, enabling users to receive tactile feedback that enhances control precision and safety.

In summary, Zhizhan Li is a cornerstone of the “Movement Reflection” project, responsible for the core framework's construction and verification. His role involves a deep collaboration with other team members to ensure that the robotic arm not only meets but exceeds the project's stringent requirements for movement fidelity, responsiveness, and user interaction.

Chenxi Wang:

Table 4.3 Labor cost for Chenxi

Decoding and Control Precision	My work begins with decoding the movements and positions relayed by the input handle. This involves sophisticated algorithms and computational strategies that interpret the handle's posture and trajectory, translating these into exact instructions for the robotic arm's movements. This translation process is crucial for maintaining the integrity of the movement reflection, ensuring that every nuance of the handle's motion is mirrored by the robotic arm with utmost precision.
Responsive Movement and Directional Accuracy	In line with the project's emphasis on responsive movement and directional accuracy, my role demands the implementation of real-time processing capabilities. This ensures that movements of the handle are followed by the robotic arm without unnecessary delay, maintaining a seamless and intuitive control experience. By fine-tuning the control algorithms, I ensure that the directional intent of the handle's movements is accurately reflected in the robotic arm's actions, thus achieving a high degree of movement synchronization and directional fidelity.
Mapping Accuracy and Minimized Delay	Achieving mapping accuracy involves complex calculations to maintain a consistent ratio between the movements of the input handle and the robotic arm. My responsibility includes adjusting the scaling factor to ensure proportional movement replication, which may require frequent calibration to adapt to different operation modes or payloads. Moreover, minimizing delay is a critical aspect of my role, necessitating ongoing optimization of the system's computational and communication protocols to ensure real-time responsiveness.
Integration of Auxiliary Controls	Another aspect of my responsibility is the integration of auxiliary controls to enhance the system's functionality. This includes the incorporation of force feedback mechanisms to provide tactile feedback to the user, enhancing control precision and safety. Additionally, I am involved in implementing calibration mechanisms that refine the system's accuracy over time, addressing any potential drift in sensor data to maintain reliable movement tracking.

In summary, my role in the “Movement Reflection” project is multifaceted and integral to its success. It encompasses the precise decoding of input movements, ensuring responsive and accurate output, and integrating advanced control features to enhance user interaction. Through my contributions, the project aims to achieve a level of control and interaction that sets new standards for robotic arm technology, ensuring its applicability in tasks that demand precision, real-time control, and intuitive operation.

Shihua Zeng

Camera Calibration

Due to the forward installation position of the camera, the working distance is approximately 500mm. To meet the visual recognition requirements of all five levels of exchange, lenses with a focal length of 2.8mm or even shorter are needed to obtain a larger field of view. However, a problem arises: the lens's fisheye effect is particularly severe (distortion), but the camera calibration method based on chessboard

pattern detection is not very effective. An inaccurate distortion matrix not only affects the results of SolvePNP calculations but can even impact contour recognition. Therefore, an accurate distortion matrix and intrinsic matrix are very important.

First, estimate an approximate value of f/dx and f/dy by dividing the focal length by the camera's image sensor size, then set u0, v0 to half of the screen width and height. This is just a rough adjustment and will be modified later.

Given that the distortion matrix only contains five values, it might be easier to create five sliders within the frame to manually adjust the distortion matrix. The callback function of the slider modifies the distortion matrix, and then two matrices, map1 and map2, are initialized. During the while loop, use `initUndistortRectifyMap(camera matrix, distortion matrix, Mat(), new camera matrix, image size, CV_16SC2, map1, map2)`. The `Mat () ` is input directly as is. Since the correction of the image will change the size of the image, the new camera matrix should ideally be calculated using the `getOptimalNewCameraMatrix()` function, but tests show that using the original camera matrix can also achieve high-accuracy pose estimation results (average error of 2mm under a working distance of 500mm). The `initUndistortRectifyMap` function assigns values to map1 and map2, and the image is adjusted using `remap (original image, new image, map1, map2, INTER_LINEAR)`. After setting camera parameters such as exposure and gain, place a calibration plate in front of the camera for visual adjustment. Due to the principle of projection, the image will appear larger near and smaller far away, but this can be ignored.

The adjustment only needs to ensure that any straight lines on the calibration plate are also straight in the image. The adjustment of the distortion matrix should now be completed. Slot Pose Estimation (Traditional Vision, Recognizing the Front Four Corners)

1. Preprocessing Correct image distortion, separate channels, and extract red and blue channels to overlay and then binarize. This binary image includes both red and blue, making both red and blue exchange slots recognizable. Due to the unique nature of engineering exchange vision, there won't be situations where both red and blue appear simultaneously during operation, avoiding potential bugs like auto describe red and blue switch, and making debugging more convenient for testing the algorithm's robustness.

2. Corner Contour Filtering In the initial screening, I used the following criteria:

a. The ratio of contour width to height and height to width is less than 4.5 (height/width < 4.5 && width/height < 4.5)

b. Contour area is larger than 400 but smaller than 12000 pixels (area > 400 && area < 12000)

c. The number of sides from polygon fitting is more than 5 but less than 9 (edges > 5 && edges < 9)

Contours that meet the above conditions are placed into a vector for secondary selection, with the selection criteria being:

a. The vector element count equals 4

b. The largest contour area is less than 10 times the smallest contour area (max.size() < 10 * min.size())

3. Corner Vertex Positioning

a. Perform triangular fitting and find the minimum enclosing circle for contours in the vector. Use the centers of the four circles to determine the midpoint of the front surface of the exchange slot. Note that due to the image characteristic of appearing larger near and smaller far, the largest angle from triangular fitting may not necessarily be the contour vertex. Similarly, the triangle vertex farthest from the midpoint may not always be the contour vertex. Observation shows that incorrect largest angle degrees generally do not exceed 100 degrees, and vertices only appear closer to the midpoint than the other two angles when exceeding 120 degrees. Based on these results, the following judgment criteria are formulated.

b. Calculate the largest angle degrees of the four triangles. If this angle is greater than 130 degrees, it is considered as the contour vertex.

c. If there are no angles greater than 130 degrees in the triangle, calculate the distance of each vertex to the midpoint and choose the farthest as the contour vertex. At this point, all four vertices have been identified.

4. Corner Sorting When the exchange slot is directly facing, assign the top-right corner as point 0, and sort in a counterclockwise direction: top-left, bottom-left, bottom-right as points 1, 2, 3, respectively. It has been observed that the order of corners in the vector is always counterclockwise (possibly due to characteristics of the OpenCV findContour function), meaning the vector corner has four possible sequences: 0123, 1230, 2301, 3012. The sequence in the vector can be determined by finding point 0.

Due to the image characteristic of appearing larger near and smaller far, the contour area corresponding to point 0 may not be the smallest. Point 0's position is determined by finding two small squares beside it, using the following criteria:

a. Area is less than 200 but greater than 50 (area < 200 && area > 50)

b. The area of the rectangle fitted from the contour is less than 1.2 times the contour area (rectangle.shape.area() < 1.2 * contour.area())

c. Both the width to height and height to width ratios are less than 2 (height/width < 2 && width/height < 2)

Now having found two small squares (or possibly only one), find the midpoint between them, and the nearest vertex to this midpoint is point 0. However, if no small square is recognized, then point 0 is definitely away from the camera, meaning the contour with the smallest area corresponds to point 0. An area judgment easily identifies point 0, after which the corners are reordered.

5. Recognition Point Filtering Since the rotation vector from PNP solution and the quaternion for final interfacing with the electronic control have certain dependencies, designing such a filter can be complex. Therefore, it's simpler to filter the recognized Point2f points, i.e., filter before SolvePnp. The principle is simple, set the recognition data of the first frame as (0,0), and the current frame data as the previous frame data multiplied by 0.9 plus the current frame data. X(0) = (0,0) X(t) = X(t-1) * 0.9 + X(t) * 0.1

6. PNP Use the SOLVEPNP_IPPE_SQUARE method, sorting the corners in the order required for square calculation. Testing showed that SOLVEPNP_IPPE_SQUARE's accuracy and speed are far superior to the common IPPE with 12 calculation points. Even with slight recognition jitter and errors, SOLVEPNP_IPPE_SQUARE can still solve correctly (with minor errors).

7. Post-Processing (Correctness Judgment of Recognition) No misrecognition occurred in the field exchange station environment, but home tests showed a chance of misrecognition due to exchange station light leakage, thus additional judgment criteria were added. The recognized quadrilateral, due to incomplete filtering fitting or misrecognition, can cause the quadrilateral to appear concave or its area to momentarily be smaller than the correct rectangular frame. Hence, it's necessary to judge the shape and area of the quadrilateral. Furthermore, since the exchange slot is angled upwards, the calculated quaternion x should be negative. Additionally, due to the filtering principle, the current frame data, if present, should not be identical to the previous frame's data, and the edge intensity of corner contours being weak can cause recognition points to have adjacent pixel jumps, leading to data jittering within a very small range (0.1 degrees, 0.5mm), so different data between adjacent frames is expected. Combining the above, the following judgment conditions are given

a. Any internal angle of the recognized quadrilateral is not less than 40 degrees and not more than 130 degrees (angle < 130 && angle > 40)

b. The area of the recognition frame is greater than 30000 pixels (rectangle.shape.area > 30000)

c. The calculated quaternion x value does not exceed 0.05 (quaternion.x < 0.05)

d. The calculation value of the current frame does not match the previous frame (quaternionNow != quaternionLater)

With an estimated 200 hours of contribution from each team member and an hourly wage of $10—reflective of a typical TA salary at UIUC—our labor calculation per partner is: $10/hour x 2.5 x 200 hours = $5,000. Thus, the total labor cost for our team amounts to $20,000.

4.3 Parts

Table 4.4 Key components for project

STM32F407 Chips	For the core operations of our project, three STM32F407 microcontroller units are essential. These high-performance chips are at the heart of our system, enabling sophisticated control algorithms and ensuring real-time processing of sensory data and control commands. The STM32F407, with its advanced architecture and features

	like high-speed memory interfaces, multiple communication interfaces, and an extensive set of peripherals, is particularly suited for our application. It ensures a seamless interface between user inputs and mechanical responses, facilitating the precise control required for the robotic arm's operations.
BMI 088 IMU Sensors	Coupled with the STM32F407 chips are IBM088 sensors, which play a critical role in detecting handle movement. These sensors are pivotal for interpreting the user's manual inputs accurately. They provide high-precision measurements of orientation and movement, enabling the system to translate the user's intentions into precise movements of the robotic arm. This ensures a highly responsive and intuitive control experience, essential for tasks requiring fine manipulation and control.
Power Management	The utilization of batteries and power lines provided by our laboratory underscores our project's integration with existing resources. This approach not only fosters innovation but also minimizes additional costs. By effectively managing power through both batteries and direct power lines, we ensure that the system is versatile and adaptable to various operational contexts, whether it requires portability or continuous operation over extended periods.
Chassis	A key addition to our project is a remotely controlled omnidirectional gear system for the chassis. This innovative feature allows our robotic arm to move its base flexibly, enabling it to adapt and respond to complex environments in real life. The ability to reposition the entire unit effortlessly enhances the system's overall versatility and efficiency, making it suitable for a wide range of applications, from precision tasks to operations in challenging or constrained spaces.
• Control System Input Handle	The control system incorporates a remote-operable handle, which, in conjunction with the three STM32F407 chips, works to capture and process user hand input. This setup is crucial for achieving a high degree of control over the robotic arm, allowing for precise manipulation based on the user's manual inputs. The handle's design and functionality are tailored to ensure that the control inputs are intuitively translated into accurate movements by the main control unit, enhancing the user experience and the system's effectiveness in performing delicate tasks.
Visual Recognition Main Control	The visual recognition system is powered by a Thor MIX 1362H000 mini-PC equipped with an Intel i7-13620H CPU. This high-performance computing unit is dedicated to processing the visual data captured by the cameras. It analyzes the information to adjust the fine angles of the robotic arm's end effector. The use of advanced image processing algorithms and the powerful computational capabilities of the i7-13620H CPU ensure that the system can recognize and interpret visual data with high precision, facilitating sophisticated manipulation and interaction with the environment.
Cameras	For capturing high-definition images, we utilize the Logitech C270i HD webcam. This camera is selected for its ability to deliver clear 720p high-definition images, essential for precise position recognition. The high-quality visual input from the Logitech C270i is vital for the visual recognition system, enabling accurate and responsive control of the robotic arm based on visual cues. This capability is fundamental for tasks that require a high degree of precision and adaptability, further enhancing the system's utility and performance.

These components collectively form a sophisticated system designed to achieve precise control and high adaptability, meeting the demands of various applications and environments.

5. Conclusion

5.1 Ethical and Safety

5.1.1 Ethical Considerations

The development and deployment of our robotic arm system, designed to replicate human hand movements and operate in hazardous environments, raise significant ethical considerations. In alignment with the IEEE Code of Ethics and ACM Code of Ethics, our project commits to prioritizing the safety, health, and welfare of the public, ensuring that our technology enhances the quality of life, fairness, and dignity of all stakeholders involved.

Privacy and Surveillance: Given the system's reliance on camera technology to capture human movements, there is a potential risk of privacy invasion. To mitigate this, our project will implement strict data handling protocols, ensuring that all captured data are anonymized and securely stored, with access restricted to authorized personnel only.

Autonomy and Misuse: The potential for the robotic arm to be repurposed for unintended or harmful activities, such as surveillance or in military applications, necessitates a robust ethical framework. We will incorporate failsafe mechanisms and operational limits to prevent misuse, ensuring the system's use remains within the scope of humanitarian and industrial assistance.

Accessibility and Inclusivity: Aligning with the principles of fairness and avoiding discrimination, our project aims to ensure that the robotic arm system is accessible and adaptable to a wide range of users, regardless of their physical abilities. This commitment extends to providing equitable access to the benefits of our technology, fostering inclusivity in its application.

5.1.2 Safety Concerns and Regulatory Compliance

Our project's design and operational framework will be developed in strict compliance with relevant safety and regulatory standards to mitigate potential risks associated with its functionality and deployment in various environments.

Operational Safety: The robotic arm will be engineered with built-in safety features, including emergency stop functions, collision avoidance systems, and adaptive response mechanisms to unexpected obstacles or failures. This approach ensures the protection of both the operator and bystanders during its operation.

Regulatory Standards: Compliance with state and federal regulations, industry standards, and campus policies will be rigorously pursued. This includes adhering to the Occupational Safety and Health Administration (OSHA) guidelines for robotic equipment and the American National Standards Institute (ANSI) safety standards for industrial robots and robot systems.

Risk of Injury: To address the potential for injury from mechanical failure or operational error, our system will undergo extensive testing and validation to meet high reliability and safety metrics. Training

protocols will be developed to educate operators on safe handling and operation procedures, minimizing the risk of accidents.

Environmental Impact: Consideration will be given to the environmental impact of our system, ensuring that materials and processes used in the construction and operation of the robotic arm are sustainable and minimize ecological footprints. This includes evaluating the lifecycle of the system and implementing recycling or disposal protocols for end-of-life units.

5.2 Conclusion

In conclusion, our project presents a versatile solution to address the pressing need for safe and efficient task execution in hazardous or inaccessible environments. By developing a movable multifunctional robotic arm system controlled through intuitive hand movements, we have overcome many of the limitations associated with traditional methods and existing robotic solutions.

Through the integration of six-axis sensors and a custom controller, our system enables precise replication of hand movements, offering operators a natural and intuitive interface. The inclusion of autonomous or remote navigation capabilities further enhances the system's adaptability to diverse terrains and scenarios, from industrial settings to disaster relief missions.

Moreover, our emphasis on user-friendly interaction ensures that our solution can be readily adopted by operators with varying levels of expertise, making it a valuable tool in a wide range of applications. Whether it's conducting complex tasks in unsafe environments or rapidly responding to emergencies, our robotic arm system offers a combination of finesse and power that is essential for success.

In summary, our project represents a significant step forward in the field of robotics, providing a flexible and efficient solution to meet the evolving challenges of modern-day operations. We believe that our innovative approach has the potential to revolutionize how tasks are performed in hazardous or inaccessible environments, ultimately enhancing safety and productivity across various industries.

6.References

[1] M. A. Islam, M. A. Hannan, M. M. H. Bhuiyan, and M. F. Rabbi, "Design and simulation of SCARA type robotic arm," in *Proceedings of the 8th International Conference on Electrical and Computer Engineering (ICECE)*, Dhaka, Bangladesh, Dec. 2014, pp. 635-638.

[2] Y. K. Park and S. S. Yi, "Integrated control of 4WS and 4WD systems for enhanced vehicle stability," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 4, pp. 660-667, Jul. 2007.

[3] T. Guo, Y. Zhang, S. Hu, and G. Zhang, "Design and implementation of a MEMS-based inertial measurement unit using BMI088," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 1, pp. 124-134, Feb. 2021.

[4] H. A. Salem and M. F. Z. Abidin, "A review on the development of 4WS and 4WD technology in vehicles," *IEEE Access*, vol. 7, pp. 15909-15921, 2019.

[5] P. E. Pius and M. F. Selekwa, "Dynamic Modelling of a 4WD/4WS Ground Vehicle by Using Gibbs-Appell Approach," *Volume 5: Dynamics, Vibration, and Control*, 2022, doi: 10.1115/imece2022-95436.

[6] "Robots in the Manufacturing Industry: Types and Applications." *Robots in the Manufacturing Industry: Types and Applications*. Available: https://www.wevolver.com/article/manufacturing-robots (accessed: May 28, 2024).

[7] "Common chassis and kinematics for mobile robots." *Common chassis and kinematics for mobile robots*. Available: https://blog.csdn.net/u011178262/article/details/128895172 (accessed: May 28, 2024).

[8] S. Kim, "Development of a fuzzy logic controller for coordination of 4WS and 4WD systems in off-road vehicles," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, Paris, France, Jun. 2018, pp. 682-687.

[9] MathWorks, "Robotics System Toolbox," MathWorks, [Online]. Available: https://www.mathworks.com/products/robotics.html. (accessed Jan. 15, 2023).

[10] T. Fukuda, T. Kudoh, and T. Nakamura, "A new SCARA-type manipulator: system design and dynamic characteristics," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 4, pp. 479-485, Aug. 1991.

[11] P. I. Corke, "A Robotics Toolbox for MATLAB," *IEEE Robotics & Automation Magazine*, vol. 3, no. 1, pp. 24-32, Mar. 1996.

[12] K. M. Lynch and F. C. Park, *Modern Robotics*. Cambridge University Press, 2017.

[13]S. Taneja, “Analysis and implementation of 4WS and 4WD systems for autonomous vehicles,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Montreal, QC, Canada, May 2019, pp. 5261-5266.

[14]J. Chen and X. Hu, “Adaptive control strategy for 4WS and 4WD electric vehicles based on neural network,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8612-8622, Sep. 2019.

[15]Robotics Online, “SCARA robot design: A beginner's guide,” Robotics Online, [Online]. Available: https://www.robotics.org/blog-article.cfm/SCARA-Robot-Design-A-Beginners-Guide/73. (accessed Jan. 15, 2023).

[16]S. M. LaValle, “Planning Algorithms,” Cambridge, UK: Cambridge University Press, 2006.

[17]M. S. Z. S. Alavi, M. L. S. V. B. G. Machado, J. G. Pereira, and P. M. A. Santos, “Evaluation of BMI088 inertial measurement unit for pedestrian navigation,” *IEEE Sensors Journal*, vol. 20, no. 11, pp. 6221-6230, Jun. 2020.

[18]S. Kazi, S. Mahmood, M. S. Baig, and M. K. Hassan, “A review on MATLAB toolbox for robotics: Robotics System Toolbox,” in *Proceedings of the International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, Sukkur, Pakistan, Mar. 2019, pp. 1-6.

[19]O. Khatib, “Springer Handbook of Robotics,” B. Siciliano and O. Khatib, Eds. Berlin, Germany: Springer-Verlag, 2008.

[20]“[RM2022-Engineering Robot Mechanical Structure Open Source] Guangdong University of Technology DynamicX[RoboMaster Forum-Geek Heaven],” bbs.robomaster.com. [Online]. Available: https://bbs.robomaster.com/forum.php?mod=viewthread&tid=22169. (accessed Mar. 26, 2024).

[21]“[RM2021-Engineering Robot Mechanical Structure Open Source] Northeastern University - T-DT Team [RoboMaster Forum-Geek Heaven],” bbs.robomaster.com. [Online]. Available: https://bbs.robomaster.com/forum.php?mod=viewthread&tid=12291. (accessed Mar. 26, 2024).

[22]“[RM2023-Engineering Robot Complete Form Technical Document Open Source] Southern University of Science and Technology [RoboMaster Forum-Geek Heaven],” bbs.robomaster.com. [Online]. Available:https://bbs.robomaster.com/forum.php?mod=viewthread&tid=22662&extra=page%3D1%26filter%3Dtypeid%26orderby%3Ddateline. (accessed Mar. 26, 2024).

[23]“[RM2022-Engineering Robot Mechanical Structure Open Source] Nanjing University of Science and Technology - Alliance [RoboMaster Forum-Geek Heaven],” bbs.robomaster.com. [Online]. Available: https://bbs.robomaster.com/forum.php?mod=viewthread&tid=22197. (accessed Mar. 26, 2024).

[24]“[RM2023-Engineering Open Source] Shanghai Jiao Tong University - Cloud Han Jiao Long Team [RoboMaster Forum-Geek Heaven],” bbs.robomaster.com. [Online]. Available:https://bbs.robomaster.com/forum.php?mod=viewthread&tid=22758. (accessed Mar. 26, 2024).

Appendix

The outcome of vertical load test is shown below.

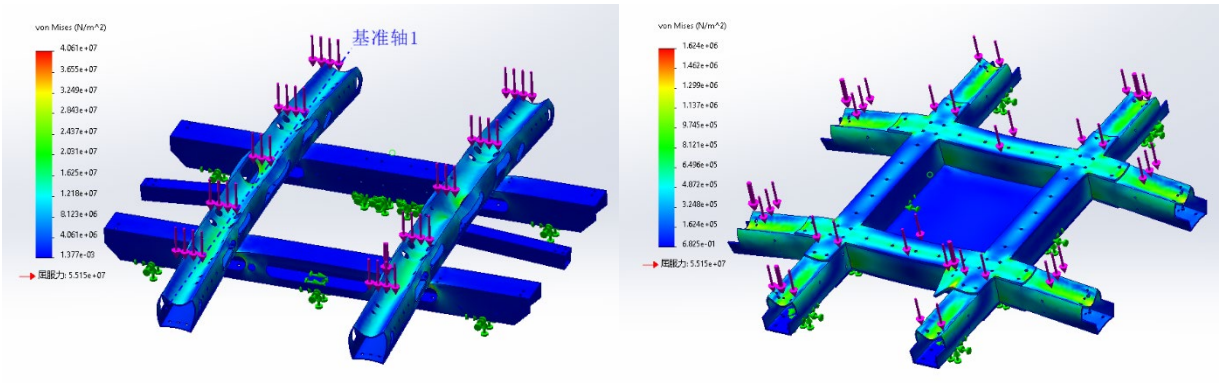

Figure 1 Von Mises stress outcome (left: former, scale = x200; right: present, scale = x5000)

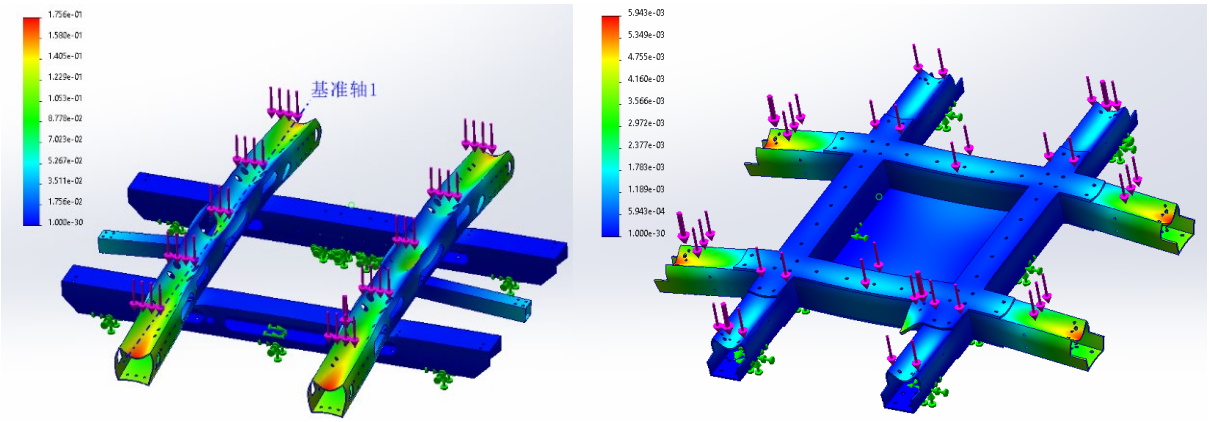

Figure 2 Total displacement outcome (left: former, scale = x200; right: present, scale = x5000)

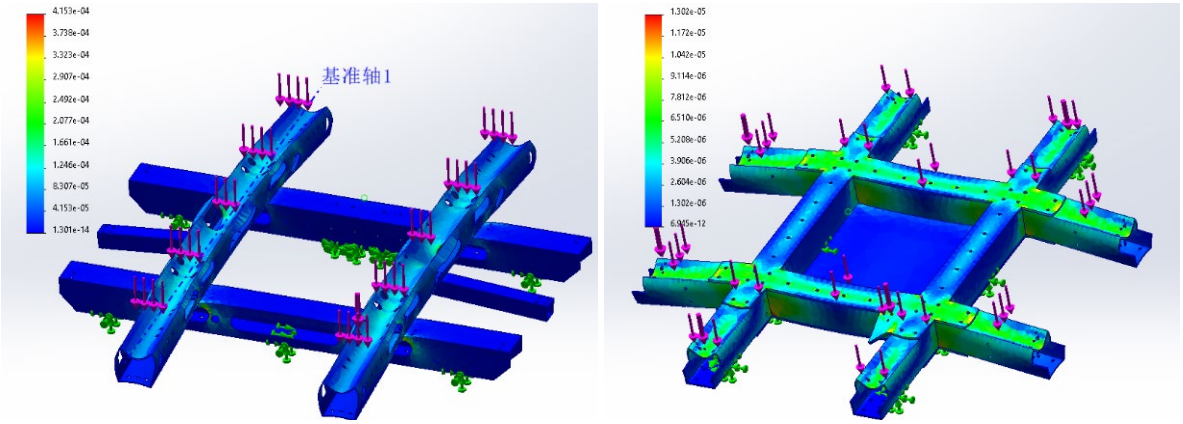

Figure 3 strain outcome (left: former, scale = x200; right: present, scale = x5000)

The outcome of axial torsion test is shown below.

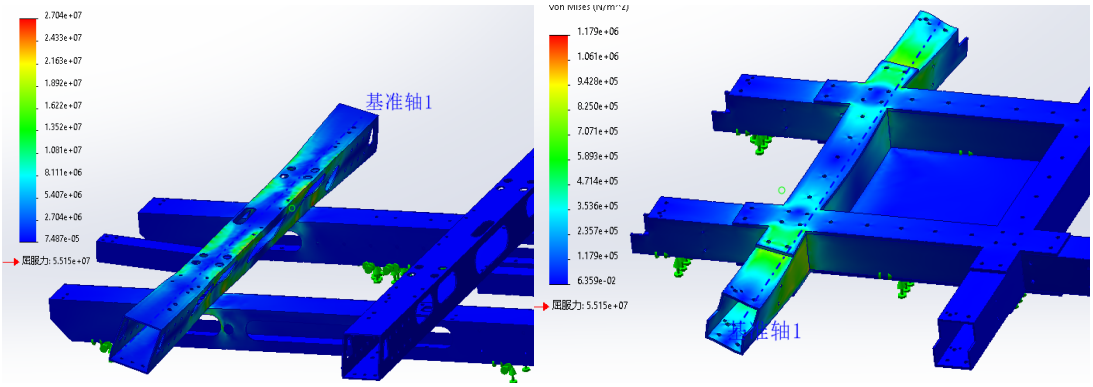

Figure 4 Von Mises stress outcome (left: former, scale = x100; right: present, scale = x5000)

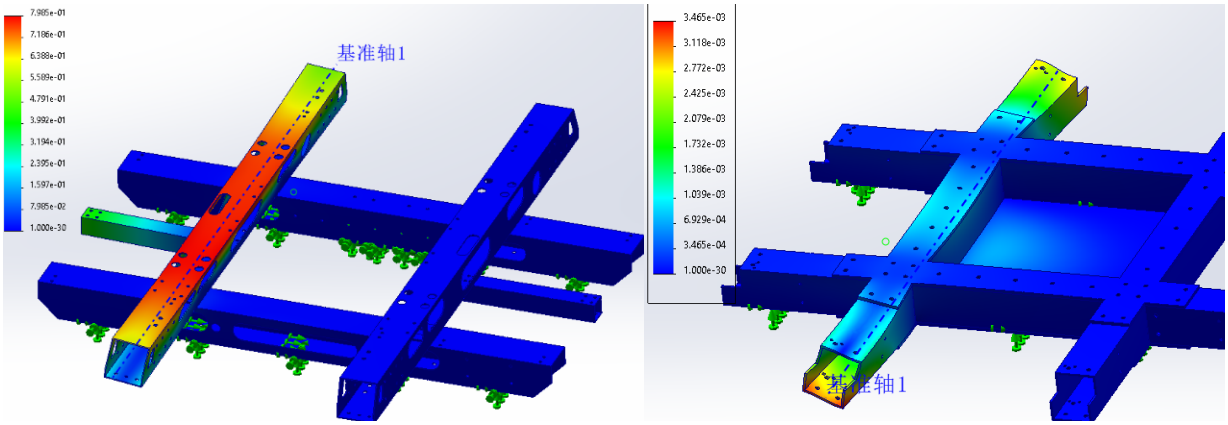

Figure 5 Total displacement outcome (left: former, scale = x100; right: present, scale = x5000)

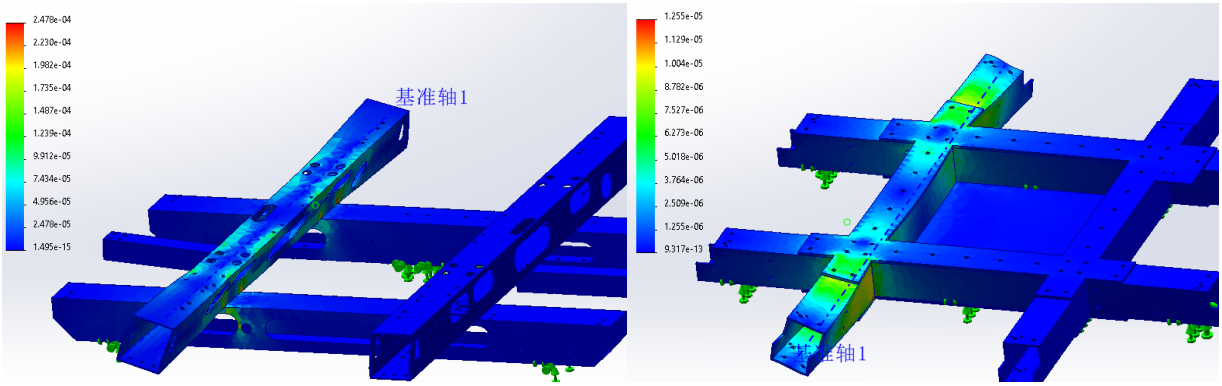

Figure 6 strain outcome (left: former, scale = x100; right: present, scale = x5000)

```
#include “imu.h”
#include “spi.h”
#include “tim.h”
#include <math.h>
```

```c
#define PI (3.1415927F)
/*经验误差*/
const float YAW_GYRO_COMPENSATION = -0.29;
const float X_ACCEL_COMPENSATION = +0.01124;
const float Y_ACCEL_COMPENSATION = -0.00063;
const float _pi_over_180_ = PI/180.0f;

/* ----------------------------  Variables  ---------------------------- */
static imu_raw_data_t imu_raw_data = {0};  // IMU 原始数据结构体
IMU_Type imu = {0,0};

/* ---------------------------- Fuction Declaration ----------------------------
--- */
void IMU_Get_Data(void);
float invSqrt(float x);
void MahonyAHRSupdateIMU(float q[4], float gx, float gy, float gz, float ax,
float ay, float az);
//陀螺仪
void BMI088_Write_Gyro_Single_Reg(uint8_t const reg, uint8_t const data);
uint8_t BMI088_Read_Gyro_Single_Reg(uint8_t const reg);
void BMI088_Read_Gyro_Multi_Reg(uint8_t *bmi_rx_data);
uint8_t BMI088_Gyro_Init(void);
//加速度计
void BMI088_Write_Acc_Single_Reg(uint8_t const reg, uint8_t const data);
uint8_t BMI088_Read_Acc_Single_Reg(uint8_t const reg);
void BMI088_Read_Acc_Multi_Reg(uint8_t *bmi_rx_data);
uint8_t BMI088_Acc_Init(void);
//微秒延时
void bmi_delay_us(uint16_t us);

uint8_t BMI088_Init(void)
{
    uint8_t state = BMI088_NO_ERROR;
    state |= BMI088_Gyro_Init();
    state |= BMI088_Acc_Init();
    if(state==BMI088_NO_ERROR){
        HAL_TIM_Base_Start_IT(&htim5);//1ms 一次中断用于数据处理
    }else{
        HAL_GPIO_WritePin(Red_GPIO_Port,Red_Pin,GPIO_PIN_SET);
    }
    return state;
}


void IMU_Get_Data(void)
```

```
{
    uint8_t gyro_buff[6];
    uint8_t acc_buff[6];
    float tmp;
    /* 陀螺仪数据读取 */
    BMI088_Read_Gyro_Multi_Reg(gyro_buff);
    tmp = (int16_t)((gyro_buff[1] << 8) | gyro_buff[0]);
    imu_raw_data.gx = (tmp / GYRO_SENSITIVITY_1000) * _pi_over_180_ ; //统一采用
弧度制
    tmp = (int16_t)((gyro_buff[3] << 8) | gyro_buff[2]);
    imu_raw_data.gy = (tmp / GYRO_SENSITIVITY_1000) * _pi_over_180_ ;
    tmp = (int16_t)((gyro_buff[5] << 8) | gyro_buff[4]);
    imu_raw_data.gz = (tmp / GYRO_SENSITIVITY_1000 + YAW_GYRO_COMPENSATION) *
_pi_over_180_ ; //误差补偿
    /* 加速度计数据读取 */
    BMI088_Read_Acc_Multi_Reg(acc_buff);
    tmp = (int16_t)((acc_buff[1] << 8) | acc_buff[0]);
    imu_raw_data.ax = (tmp / ACCEL_SENSITIVITY_3)   +   X_ACCEL_COMPENSATION; //
误差补偿
    tmp = (int16_t)((acc_buff[3] << 8) | acc_buff[2]);
    imu_raw_data.ay = (tmp / ACCEL_SENSITIVITY_3)   +   Y_ACCEL_COMPENSATION; //
误差补偿
    tmp = (int16_t)((acc_buff[5] << 8) | acc_buff[4]);
    imu_raw_data.az = (tmp / ACCEL_SENSITIVITY_3);
}


void BMI088_Write_Gyro_Single_Reg(uint8_t const reg, uint8_t const data)
{
    uint8_t bmi_rx_byte, bmi_tx_byte;
    //开始
    HAL_GPIO_WritePin(CS1_GYRO_GPIO_Port, CS1_GYRO_Pin, GPIO_PIN_RESET);  // 拉低
片选信号，开始传输 NSS_Low
    //数据交换
    bmi_tx_byte = reg & 0x7f;//首位是 0，表示此操作为写入，reg 后七位即为所写地址
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1,
BMI088_SPI_TIME_THRES);
    bmi_tx_byte = data;
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1,
BMI088_SPI_TIME_THRES);
    //结束
    HAL_GPIO_WritePin(CS1_GYRO_GPIO_Port, CS1_GYRO_Pin, GPIO_PIN_SET);  // 拉高片
选信号，结束传输 NSS_High
}
```

```c
uint8_t BMI088_Read_Gyro_Single_Reg(uint8_t const reg)
{
    uint8_t bmi_rx_byte, bmi_tx_byte;
    //开始
    HAL_GPIO_WritePin(CS1_GYRO_GPIO_Port, CS1_GYRO_Pin, GPIO_PIN_RESET);  // 拉低片选信号，开始传输 NSS_Low
    //数据交换
    bmi_tx_byte = reg | 0x80;  // 第一位为 1 表示读操作
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1, BMI088_SPI_TIME_THRES);
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1, BMI088_SPI_TIME_THRES);
    //结束
    HAL_GPIO_WritePin(CS1_GYRO_GPIO_Port, CS1_GYRO_Pin, GPIO_PIN_SET);  // 拉高片选信号，结束传输 NSS_High
    return bmi_rx_byte;
}

void BMI088_Read_Gyro_Multi_Reg(uint8_t *bmi_rx_data)
{
    uint8_t bmi_rx_byte, bmi_tx_byte, len = 6;
    HAL_GPIO_WritePin(CS1_GYRO_GPIO_Port, CS1_GYRO_Pin, GPIO_PIN_RESET);  // 拉低片选信号，开始传输 NSS_Low
    bmi_tx_byte = BMI088_GYRO_X_L | 0x80;  // 第一位为 1 表示读操作
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1, BMI088_SPI_TIME_THRES);
    while (len != 0) {
        HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_data[6 - len], 1, BMI088_SPI_TIME_THRES);
        len--;
    }
    HAL_GPIO_WritePin(CS1_GYRO_GPIO_Port, CS1_GYRO_Pin, GPIO_PIN_SET);  // 拉高片选信号，结束传输 NSS_High
}

uint8_t BMI088_Gyro_Init(void)
{
    uint8_t BMI088_Gyro_Init_Config[3][3] = {
        {BMI088_GYRO_RANGE, BMI088_GYRO_1000, BMI088_GYRO_RANGE_ERROR},
        {BMI088_GYRO_BANDWIDTH, BMI088_GYRO_1000_116_HZ | BMI088_GYRO_BANDWIDTH_MUST_Set, BMI088_GYRO_BANDWIDTH_ERROR},
        {BMI088_GYRO_LPM1, BMI088_GYRO_NORMAL_MODE, BMI088_GYRO_LPM1_ERROR}
    };
    static uint8_t read_value;
```

```
    BMI088_Write_Gyro_Single_Reg(BMI088_GYRO_SOFTRESET,
BMI088_GYRO_SOFTRESET_VALUE);
    HAL_Delay(BMI088_LONG_DELAY_TIME);
    // check commiunication is normal after reset
    read_value = BMI088_Read_Gyro_Single_Reg(BMI088_GYRO_CHIP_ID);
    bmi_delay_us(BMI088_COM_WAIT_SENSOR_TIME);
    // check the “who am I”
    if (read_value != BMI088_GYRO_CHIP_ID_VALUE) {
        return BMI088_NO_SENSOR;
    }
    // 根据 BMI088_Gyro_Init_Config 的配置，写入相应寄存器
    for (uint8_t i = 0; i < 3; i++) {
        BMI088_Write_Gyro_Single_Reg(BMI088_Gyro_Init_Config[i][0],
BMI088_Gyro_Init_Config[i][1]);
        bmi_delay_us(BMI088_COM_WAIT_SENSOR_TIME);
        read_value = BMI088_Read_Gyro_Single_Reg(BMI088_Gyro_Init_Config[i][0]);
        bmi_delay_us(BMI088_COM_WAIT_SENSOR_TIME);
        if (read_value != BMI088_Gyro_Init_Config[i][1]) {
        return BMI088_Gyro_Init_Config[i][2];  // 若有错误立即返回，不会进行后续配置
        }
    }
    return BMI088_NO_ERROR;
}

void BMI088_Write_Acc_Single_Reg(uint8_t const reg, uint8_t const data)
{
    // TODO(Hello World):
    uint8_t bmi_rx_byte, bmi_tx_byte;
    //开始
    HAL_GPIO_WritePin(CS1_ACCEL_GPIO_Port, CS1_ACCEL_Pin, GPIO_PIN_RESET);  // 拉
低片选信号，开始传输
    //数据交换
    bmi_tx_byte = reg & 0x7f;
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1,
BMI088_SPI_TIME_THRES);
    bmi_tx_byte = data;
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1,
BMI088_SPI_TIME_THRES);
    //结束
    HAL_GPIO_WritePin(CS1_ACCEL_GPIO_Port, CS1_ACCEL_Pin, GPIO_PIN_SET);  // 拉高
片选信号，结束传输
}


uint8_t BMI088_Read_Acc_Single_Reg(uint8_t const reg)
```

```
{
    uint8_t bmi_rx_byte, bmi_tx_byte;
    //开始
    HAL_GPIO_WritePin(CS1_ACCEL_GPIO_Port, CS1_ACCEL_Pin, GPIO_PIN_RESET);  // 拉低片选信号，开始传输
    //数据交换
    bmi_tx_byte = reg | 0x80;  // 第一位为 1 表示读操作
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1, BMI088_SPI_TIME_THRES);//bit0 读，bit1-7 确定读取地址
    bmi_tx_byte = 0x55; //芝士 magic number，反正官方例程是这么写的
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1, BMI088_SPI_TIME_THRES);//bit8-15 无效值
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1, BMI088_SPI_TIME_THRES);//bit16-23 以及之后为有效值
    //结束
    HAL_GPIO_WritePin(CS1_ACCEL_GPIO_Port, CS1_ACCEL_Pin, GPIO_PIN_SET);  // 拉高片选信号，结束传输
    return bmi_rx_byte;
}


void BMI088_Read_Acc_Multi_Reg(uint8_t *bmi_rx_data)
{
    uint8_t bmi_rx_byte, bmi_tx_byte, len = 6;
    //开始
    HAL_GPIO_WritePin(CS1_ACCEL_GPIO_Port, CS1_ACCEL_Pin, GPIO_PIN_RESET);  // 拉低片选信号，开始传输
    //数据交换
    bmi_tx_byte = BMI088_ACCEL_XOUT_L | 0x80;  // 第一位为 1 表示读操作，寄存器地址为加速度计 0x12，x 轴方向低八位
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1, BMI088_SPI_TIME_THRES);
    HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_byte, 1, BMI088_SPI_TIME_THRES);//额外的读取操作，筛选掉无效读数
    while (len != 0) {
        HAL_SPI_TransmitReceive(&hspi1, &bmi_tx_byte, &bmi_rx_data[6 - len], 1, BMI088_SPI_TIME_THRES);
        len--;
    }
    //结束
    HAL_GPIO_WritePin(CS1_ACCEL_GPIO_Port, CS1_ACCEL_Pin, GPIO_PIN_SET);  // 拉高片选信号，结束传输
}
```

```c
uint8_t BMI088_Acc_Init(void)
{
    /* 加速度计配置 */
    // 配置一些寄存器，并定义对应的错误
    uint8_t BMI088_Acc_Init_Config[4][3] = {
        {BMI088_ACC_RANGE, BMI088_ACC_RANGE_3G, BMI088_ACC_RANGE_ERROR},
        {BMI088_ACC_CONF, BMI088_ACC_800_HZ | BMI088_ACC_CONF_MUST_Set,
BMI088_ACC_CONF_ERROR},
        {BMI088_ACC_PWR_CTRL, BMI088_ACC_ENABLE_ACC_ON,
BMI088_ACC_PWR_CTRL_ERROR},
        {BMI088_ACC_PWR_CONF, BMI088_ACC_PWR_ACTIVE_MODE,
BMI088_ACC_PWR_CONF_ERROR}
    };
    static uint8_t read_value;  // 写入上述寄存器后再读取的值，用来检查是否正确配置
    //软件复位，复位后必须开启，byd 不然用不了
    BMI088_Write_Acc_Single_Reg(BMI088_ACC_SOFTRESET,
BMI088_ACC_SOFTRESET_VALUE);
    HAL_Delay(BMI088_LONG_DELAY_TIME);
    BMI088_Write_Acc_Single_Reg(BMI088_ACC_PWR_CONF,BMI088_ACC_PWR_ACTIVE_MODE);
    bmi_delay_us(BMI088_COM_WAIT_SENSOR_TIME);
    // check commiunication is normal after reset
    read_value = BMI088_Read_Acc_Single_Reg(BMI088_ACC_CHIP_ID);
    bmi_delay_us(BMI088_COM_WAIT_SENSOR_TIME);
    // check the “who am I”
    if (read_value != BMI088_ACC_CHIP_ID_VALUE) {
        return BMI088_NO_SENSOR;//**?
    }
    // 写入相应寄存器
    for (uint8_t i = 0; i < 4; i++) {
        BMI088_Write_Acc_Single_Reg(BMI088_Acc_Init_Config[i][0],
BMI088_Acc_Init_Config[i][1]);
        bmi_delay_us(BMI088_COM_WAIT_SENSOR_TIME);
        read_value = BMI088_Read_Acc_Single_Reg(BMI088_Acc_Init_Config[i][0]);
        bmi_delay_us(BMI088_COM_WAIT_SENSOR_TIME);
        if (read_value != BMI088_Acc_Init_Config[i][1]) {
        return BMI088_Acc_Init_Config[i][2];  // 若有错误立即返回，不会进行后续配置
        }
    }
    return BMI088_NO_ERROR;
}

void bmi_delay_us(uint16_t us)
{
    uint32_t ticks = 0;
    uint32_t told = 0;
```

```
    uint32_t tnow = 0;
    uint32_t tcnt = 0;
    uint32_t reload = 0;

    reload = SysTick->LOAD;
    ticks = us * 168;
    told = SysTick->VAL;

    while (1) {
        tnow = SysTick->VAL;
        if (tnow != told) {
        if (tnow < told) {
            tcnt += told - tnow;
        } else {
            tcnt += reload - tnow + told;
        }
        told = tnow;
        if (tcnt >= ticks) {
            break;
        }
        }
    }
}

/* ------------------------------ 姿态解算 ------------------------------ */
//变量
const float sampleFreq = 1000.0f;       // sample frequency in Hz
const float twoKp = (2.0f * 0.3f);  // 2 * proportional gain (Kp)
const float twoKi = (2.0f * 0.0f);  // 2 * integral gain (Ki)
volatile float integralFBx = 0.0f, integralFBy = 0.0f, integralFBz = 0.0f;  //
integral error terms scaled by Ki

float invSqrt(float x)
{
    float halfx = 0.5f * x;
    float y = x;
    long i = *(long *)&y;
    i = 0x5f3759df - (i >> 1);
    y = *(float *)&i;
    y = y * (1.5f - (halfx * y * y));
    return y;
}

void MahonyAHRSupdateIMU(float q[4], float gx, float gy, float gz, float ax,
float ay, float az)
```

```
{
    float recipNorm;
    float halfvx, halfvy, halfvz;
    float halfex, halfey, halfez;
    float qa, qb, qc;

    // Compute feedback only if accelerometer measurement valid (avoids NaN in
accelerometer normalisation)
    if (!((ax == 0.0f) && (ay == 0.0f) && (az == 0.0f))) {
        // Normalise accelerometer measurement
        recipNorm = invSqrt(ax * ax + ay * ay + az * az);
        ax *= recipNorm;
        ay *= recipNorm;
        az *= recipNorm;

        // Estimated direction of gravity and vector perpendicular to magnetic
flux
        halfvx = q[1] * q[3] - q[0] * q[2];
        halfvy = q[0] * q[1] + q[2] * q[3];
        halfvz = q[0] * q[0] - 0.5f + q[3] * q[3];

        // Error is sum of cross product between estimated and measured direction
of gravity
        halfex = (ay * halfvz - az * halfvy);
        halfey = (az * halfvx - ax * halfvz);
        halfez = (ax * halfvy - ay * halfvx);

        // Compute and apply integral feedback if enabled
        if (twoKi > 0.0f) {
        integralFBx += twoKi * halfex * (1.0f / sampleFreq);  // integral error
scaled by Ki
        integralFBy += twoKi * halfey * (1.0f / sampleFreq);
        integralFBz += twoKi * halfez * (1.0f / sampleFreq);
        gx += integralFBx;  // apply integral feedback
        gy += integralFBy;
        gz += integralFBz;
        } else {
        integralFBx = 0.0f;  // prevent integral windup
        integralFBy = 0.0f;
        integralFBz = 0.0f;
        }

        // Apply proportional feedback
        gx += twoKp * halfex;
        gy += twoKp * halfey;
```

```
        gz += twoKp * halfez;
    }

    // Integrate rate of change of quaternion
    gx *= (0.5f * (1.0f / sampleFreq));  // pre-multiply common factors
    gy *= (0.5f * (1.0f / sampleFreq));
    gz *= (0.5f * (1.0f / sampleFreq));
    qa = q[0];
    qb = q[1];
    qc = q[2];
    q[0] += (-qb * gx - qc * gy - q[3] * gz);
    q[1] += (qa * gx + qc * gz - q[3] * gy);
    q[2] += (qa * gy - qb * gz + q[3] * gx);
    q[3] += (qa * gz + qb * gy - qc * gx);

    // Normalise quaternion
    recipNorm = invSqrt(q[0] * q[0] + q[1] * q[1] + q[2] * q[2] + q[3] * q[3]);
    q[0] *= recipNorm;
    q[1] *= recipNorm;
    q[2] *= recipNorm;
    q[3] *= recipNorm;
}
void Get_IMU_Yaw(void)
{
    static float q[4]={1.0f , 0.0f , 0.0f , 0.0f};  // 四元数
    static float Old_Yaw = 0;
    static int32_t Circle = 0;
    float newYaw;
    IMU_Get_Data();
    /* 由于板子是躺着放的，所以必须对部分数据取反 */
    MahonyAHRSupdateIMU(q,
    imu_raw_data.gx,
    -imu_raw_data.gy,
    -imu_raw_data.gz,
    imu_raw_data.ax,
    -imu_raw_data.ay,
    -imu_raw_data.az);
    newYaw = atan2f(2.0f * (q[0] * q[3] + q[1] * q[2]), 2.0f * (q[0] * q[0] +
q[1] * q[1]) - 1.0f);
    //*pitch = asinf(-2.0f * (q[1] * q[3] - q[0] * q[2])); //暂时用不到 pitch 角度
    if(newYaw-Old_Yaw<-PI){Circle+=1;}
    if(newYaw-Old_Yaw> PI){Circle-=1;}
    Old_Yaw = newYaw;//更新旧角度
    //累计的弧度广播到全局
    imu.Yaw_Angle = 2.0f*PI*Circle + newYaw;
```

```
    //云台角速度广播到全局，单位：rad/s
    imu.Yaw_Velocity = -imu_raw_data.gz; //板子躺着放，取反。
}
```