Multiple Mobile Manipulator System

Scientific work within the practical course
from the Department of Electrical and Computer Engineering at the Technical University of Munich.

Supervised by	Univ.-Prof. Dr.-Ing. Sandra Hirche Dr. Yi Ren Chair of Information-Oriented Control
Submitted by	Ziwei Xu Chengjie Yuan Cong Wang
Submitted on	Munich, 05.07.2019

Abstract

The purpose of this project, is to let the mobile robots cooperate with each other in a multi-robot system. The system is composed of a camera system *qualisys* and two mobile robots *mofa*. In the project, we build a communication framework between the two mobile robots, the motion capture system *qualisys* and a manually implemented node *planner*.In chapter 2 we realize the close-loop control of the mobile robots, and get the experiment results of different proportional gain of the position-loop. In chapter 3 we realize that, the robot *mofa-1* follows the movement of *mofa-2*. We also build a visualization system, which is written in chapter 4, to visualize the whole system.

Contents

Chapter 1

Introduction

In this chapter, we will briefly introduce the background, major tasks and structure of this project. Furthermore, we will indicate the task division between the team mates and introduce the structure of major part of this report.

1.1 Project Description

Multi-robot systems, compared with single-robot systems, have several unique advantages on providing solutions in industrial and scientific tasks, e.g. when the task is too complex for a single-robot system to solve, or the task is essentially distributed [Par08]. Since the multi-robot system requires collaborative work of each single robot unit, the information communication between each unit, which requires very high precision and real-time capability, would be an important task in the construction of the system.

In this project practical course, we work on a multiple mobile robot system which consists of two mobile robots and a motion capture system *qualisys* with multiple cameras. Each of the mobile robot is integrated by a Franka Emika Panda robot arm and a mobile platform *mofa*. The final goal of this project is to construct a communication framework inside the multiple mobile robot system and realize distributed close-loop control of the motion of mobile robot. The programming task is based on ROS(Robot Operating System) and the programming language C++.

Figure 1.1 has described the distributed structure of the multiple mobile robot system. For each mobile robot, we use a 4x4-inch mini PC *NUC* to control its motion. That is, NUC-1 and NUC-2 independently controls the movement of the robot mofa-1 and mofa-2, so we don't have a central controller in the system. NUC-5 is a manager PC that monitors the status of each individual mobile robot. It is connected with NUC-1 and NUC-2 through the secure shell(SSH), which allows NUC-5 operating the terminals in NUC-1 and NUC-2 directly. The control package of each individual mobile robot as well as the SSH connection are initially implemented

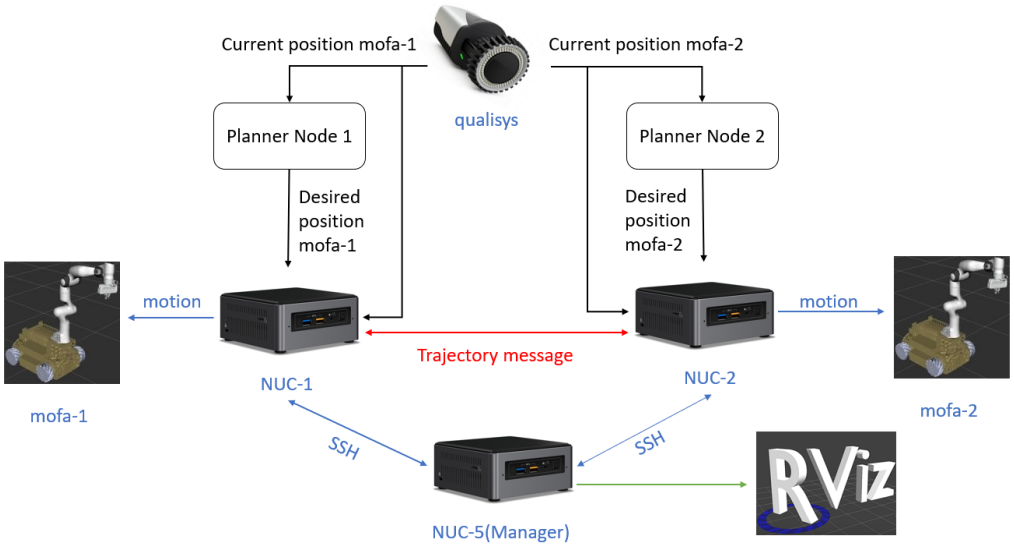


Figure 1.1: Structure of multiple mobile robot system

(blue texts and arrows in figure 1.1).

Our major task in this project is to connect the existing parts in this multiple mobile robot system together through a communication framework. This could be divided in three parts. The first part is the construction of connection between the mobile robot, the motion capture system *qualisys*, and a manually implemented planner node (black texts and arrows in figure 1.1). In this part we can realize the close-loop control of the motion of each individual mobile robot based on its current position and desired position. The second part is construction of communication between the two mobile robot units. In this part we can realize the distributed control of multiple mobile robots based on the trajectory message communication (red texts and arrow in figure 1.1). The last part would be the visualization of this mobile robot system through the visualization tool *RViz*. This allows the user to monitor the position and motion of each individual mobile robot based on the coordinate system of *qualisys* in real time (green texts and arrow in figure 1.1). The communication framework is based on ROS messages, which can be published and subscribed by all devices in the local area network. In the next chapters, we will introduce each part of the project explicitly.

1.2 Mofa-Robot

1.2.1 The Structure of Mofa Robot

Each mofa robot is composed of two parts: a omnidirectional mobile platform and a franka robot arm. Figure 1.2 is the model of our mofa robot. The franka robot arm

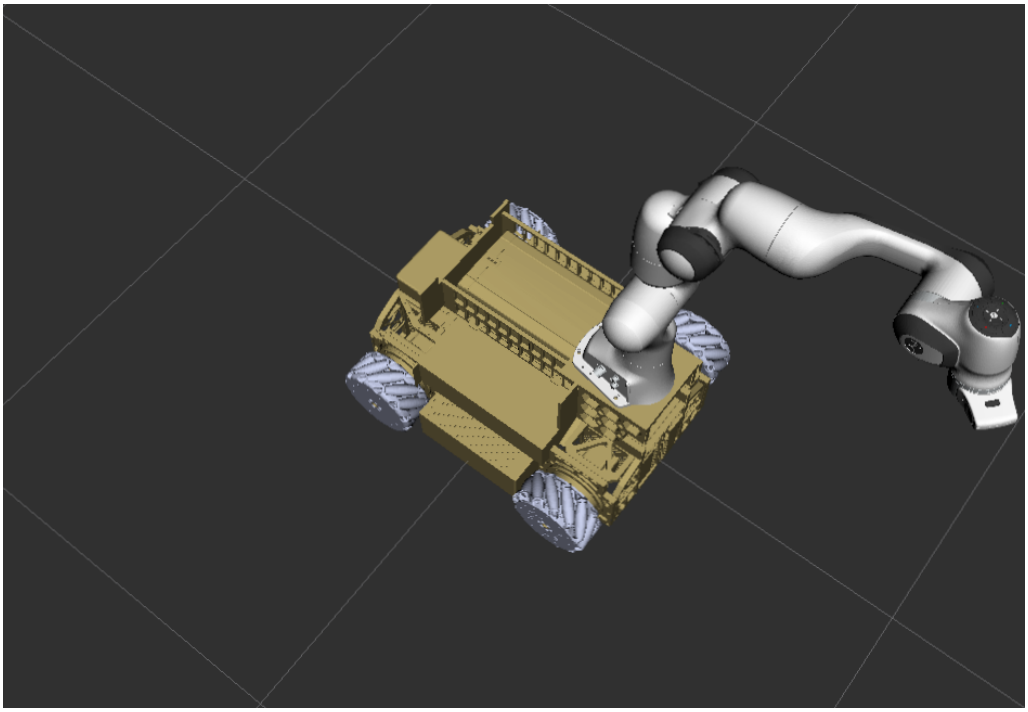


Figure 1.2: Model of mofa-platform

is placed on the front of the mobile platform. The mofa platform has four mecanum wheels, each diagonal is the same kind of mecanum wheels, as shown in figure 1.3 and 1.4. With this four mecanum wheels, the mofa platform can move omnidirectionally. For example, when all wheels move forward, the mofa platform will move forward; when wheels of mean diagonal move forward, vice diagonal move backward, the mofa platform will move horizontally; when wheels of left side move backwards, wheels of right side move forwards, the mofa platform will rotate counterclockwise.

1.2.2 Kinematic Analysis of Mofa Platform

Figure 1.7 is the model of mofa platform. In the mofa platform, we name the front of the platform is the x-direction of the coordinate system. And the front-right wheel is the wheel-1, the other wheels are arranged counterclockwise. In order to control the mofa platform, we use the inverse kinematic model to calculate the angular velocity of the four wheels. [CR17]
Figure 1.5 is the model of mofa platform frame in the world frame. We have:

$$\dot{\mathbf{x}}_W = \mathbf{R}^{-1}(\theta) \cdot \dot{\mathbf{x}}_R \tag{1.1}$$

$$with \quad \dot{\mathbf{x}}_R = \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta} \end{bmatrix}, \quad \dot{\mathbf{x}}_W = \begin{bmatrix} \dot{x}_W \\ \dot{y}_W \\ \dot{\theta} \end{bmatrix}, \quad \mathbf{R}(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 1.3: Mecanum wheel type 1

Figure 1.4: Mecanum wheel type 2

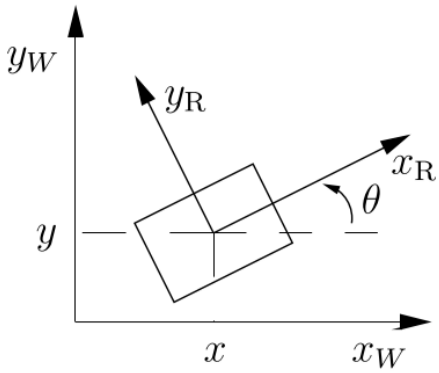


Figure 1.5: Mofa platform in world frame

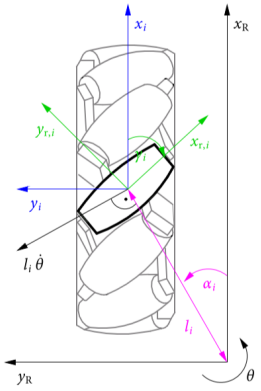


Figure 1.6: Kinematic model of Mecanum wheels with frames

The inverse kinematics of the platform can be written by:

$$\begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \\ \dot{\varphi}_3 \\ \dot{\varphi}_4 \end{bmatrix} = \mathbf{J}_{inv} \cdot \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta} \end{bmatrix} \tag{1.2}$$

with $\dot{\varphi}_i$ the angular velocity of the wheel(i=1,2,3,4), and the $\mathbf{J}_{inv}$ is the inverse Jacobian matrix of the platform. With the kinematic model of the Mecanum Wheel(Figure1.6), we can calculate the inverse Jacobian matrix:

$$\mathbf{J}_{inv} = \frac{1}{r} \cdot \begin{bmatrix} 1 & 1 & (a+b) \\ -1 & 1 & (a+b) \\ -1 & -1 & (a+b) \\ 1 & -1 & (a+b) \end{bmatrix} \tag{1.3}$$

with a and b are the distances between the center of the platform and the center of

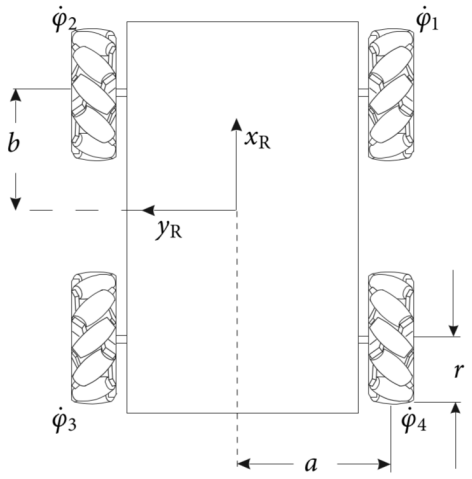


Figure 1.7: Model of mofa platform

the wheel, r is the radius of the wheel. With the mofa platform, we need to invert angular velocities of wheel-2 and wheel-3 to control the motors of these two wheels, so the second and third rows of the Jacobian inverse matrix have been inverted (Equation 1.3).

In order to control the mofa platform with the inverse kinematic model, we need the velocity of the platform in x and y direction, and also the angular velocity of the mofa platform. This will be explained in the next chapter.

1.3 Structure of the report and task division

According to the three parts of the project introduced above, each of our team mates will be mainly responsible for one part. Following is the task division and the structure of the major part of our report:

- Kinematic analysis of mofa platform, construction of connection between the mobile robot and *qualisys*— Cong Wang (Chapter 1, Chapter 2.1, 2.2)
- Construction of connection between the mobile robot, *qualisys* and planner node, construction of communication between the two mobile robot units — Ziwei Xu (Chapter 2.2, 2.3, Chapter 3)
- Robot modeling and simulation, visualization of real mobile robot system using *RViz* — Chengjie Yuan (Chapter 4)

Chapter 2

Communication Framework on Single Mobile Robot

In this chapter, we will introduce the contents about the first part of the project: the communication framework between the single mobile robot, the motion capture system *qualisys* and the manually implemented planner node. First we will introduce the communication between the robot and qualisys system, and then illustrate our implementation of the planner node for realizing the close-loop motion control of the mobile robot. In the last section we will show some results of the close-loop control experiment.

2.1 Communication between Mobile Robot and Qualisys System

The motion capture system *qualisys* can track the coordinate of the markers within its coordinate frame. Using three or more markers at the same time, we can define a rigid body and measure its orientation and the position of the center point. In this task, we use six markers to determine the mobile platform. The point position measured by the qualisys system is the 3D-Cartesian-coordinate (x, y, z), and the orientation of the rigid body is given in form of a quaternion number $w+xi+yj+zk$. In our ROS packages, the publisher of qualisys message is initially implemented, and inside the local area network, the messages could be recognized by all devices. To construct the communication between the mobile robot and qualisys system, we just need to implement a subscriber inside the control package of mobile platform, so that the robot could extract its position information from the message. The information we need from the qualisys system is the 2D-position(coordinate (x, y)) and the yaw angle(rotation angle around z-axis) of the mobile robot. Therefore, we need to transfer the quaternion number into euler angle and extract the yaw angle. First, transfer the quaternion number into a rotation matrix $\mathbf{R}$ [Zha18]:

$$\mathbf{R} = \begin{pmatrix} 1-2(y^2+z^2) & 2(xy-wz) & 2(xz+wy) \\ 2(xy+wz) & 1-2(x^2+z^2) & 2(yz-wx) \\ 2(xz-wy) & 2(yz+wx) & 1-2(x^2+y^2) \end{pmatrix} \tag{2.1}$$

Then, calculate the yaw angle from the rotation matrix:

$$\theta_{yaw} = atan2(r_{21}, r_{11}) = atan2(2(xy+wz), 1-2(y^2+z^2)) \tag{2.2}$$

Besides deriving the mathematical formula directly, the transformation between the quaternion number and euler angle can be calculated using the functions inside the ROS package *tf*. In conclusion, after the transformation of the qualisys message, we have the 2D-position of the mobile platform (x, y, θ_{yaw}) inside the control package.

2.2 Planner Node and Close-loop motion control

So far we are able to detect the current position of the mobile platform in the coordinate frame of qualisys, but it is still not enough for realizing a close-loop motion control of the robot. To control the position of the robot, the three-loop control method, which is widely used in the servo motor control, is adopted in this task [WLW13].

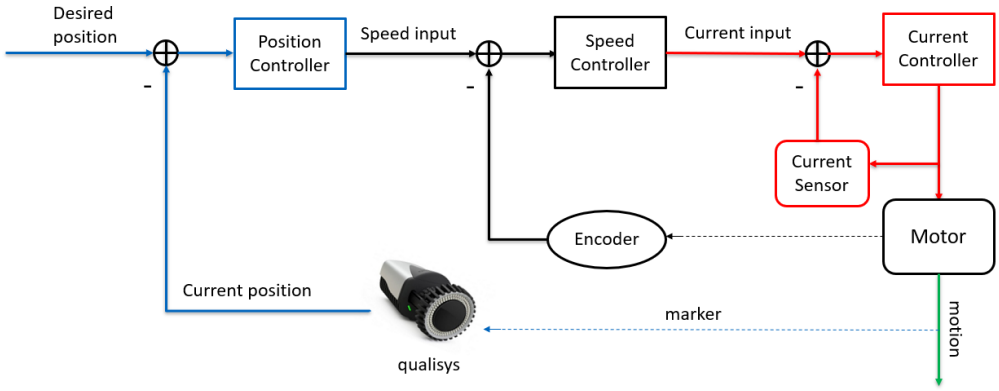


Figure 2.1: Structure of three-loop control method

The three-loop control method contains position-loop, speed-loop and current-loop(from outside to inside). In the control package of the mobile platform, the speed-loop and current-loop is initially implemented. That is, the user can manually give the speed in x-, y-direction and the angular speed of yaw angle as input to control the rotation speed of the four Mecanum wheels on the platform, without knowing the structure of the current-loop. In this task, the implementation of a position-loop in the control package is needed. In other words, we need to transfer the position information into speed information and input it to the speed-loop. According to figure 2.1, the current position of the mobile platform we have is the feedback message in the

position-loop, and what still missing is the position input and controller. Obviously, the position input should be the desired trajectory of the mobile platform. To obtain the desired trajectory, a solution of motion planning is needed. Therefore, we manually implement a planner node for the motion planning task.

The planner node is designed to calculate the desired trajectory $x_d(t)$ based on the initial position x_0 and goal position x_f of the mobile robot, which could be received from the qualisys system using markers. A common and simple way to generate a smooth point-to-point trajectory is the cubic polynomial method, which is adopted in this task. [SHV05]

First consider a cubic trajectory:

$$x_d(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{2.3}$$

And the velocity function could be derived from the trajectory:

$$v_d(t) = \frac{\partial x_d(t)}{\partial t} = a_1 + 2a_2 t + 3a_3 t^2 \tag{2.4}$$

Since the velocity of the robot at initial and goal position should be 0, including the coordinate of initial position x_0 and goal position x_f, we have four constraints with four unknown coefficients in this equation system:

$$x_0 = x_d(0) = a_0 \tag{2.5}$$

$$v_0 = v_d(0) = a_1 = 0 \tag{2.6}$$

$$x_f = x_d(t_f) = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \tag{2.7}$$

$$v_f = v_d(t_f) = a_1 + 2a_2 t_f + 3a_3 t_f^2 = 0 \tag{2.8}$$

Solving the equation system above, we can calculate the coefficients according to x_0, x_f and t_f:

$$a_0 = x_0, \quad a_1 = 0, \quad a_2 = \frac{3(x_f - x_0)}{t_f^2}, \quad a_3 = \frac{2(x_0 - x_f)}{t_f^3}$$

So far is the desired trajectory function $x_d(t)$ completely derived. In the main loop of our planner node, we calculate the desired position in real time and send it to the local area network. At the same time, the control package of the mobile robot subscribes the message for further calculation. The last step of the close-loop motion control is to implement a position controller inside the control package to complete the position-loop. In this task, a P-controller is adopted:

$$v = k_{pcl}(x_d(t) - x_{mofa}) \tag{2.9}$$

In the equation above, x_{mofa} is the current position of the mobile robot, and k_{pcl} is the proportional gain. Finally, we can conclude the structure of our position-loop:

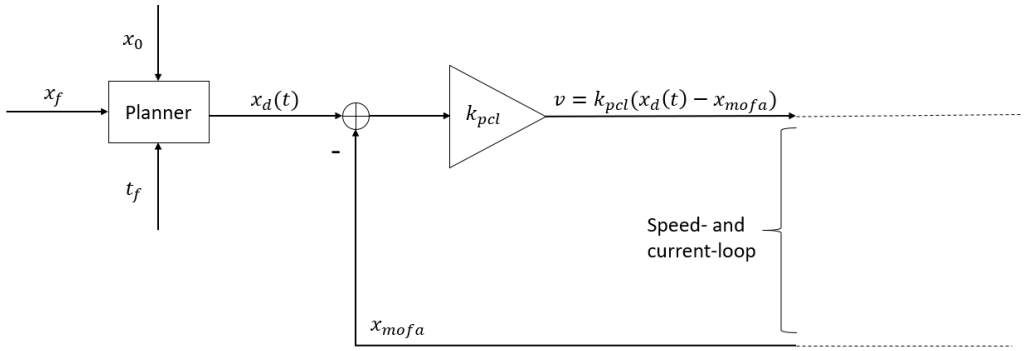


Figure 2.2: Structure of the position-loop

2.3 Experimental Results

In order to adjust the parameter value k_{pcl} of the position controller, we designed an experiment to measure the tracking error $x_d(t) - x_{mofa}$ of the close loop motion control. In this experiment, our trajectory length $x_f - x_0$ is 1.5m along the x-axis(straight forward movement) and total movement time t_f is 20s. For safety reason, we have set the maximum speed of the mobile robot to 0.5m/s. Since we measure the tracking error in millimeters and our speed input is in units of meters per second, we have to select k_{pcl} with a very small value to match different units. In this experiment, we compared three sets of measurement which k_{pcl} varies from 0.001 to 0.003.

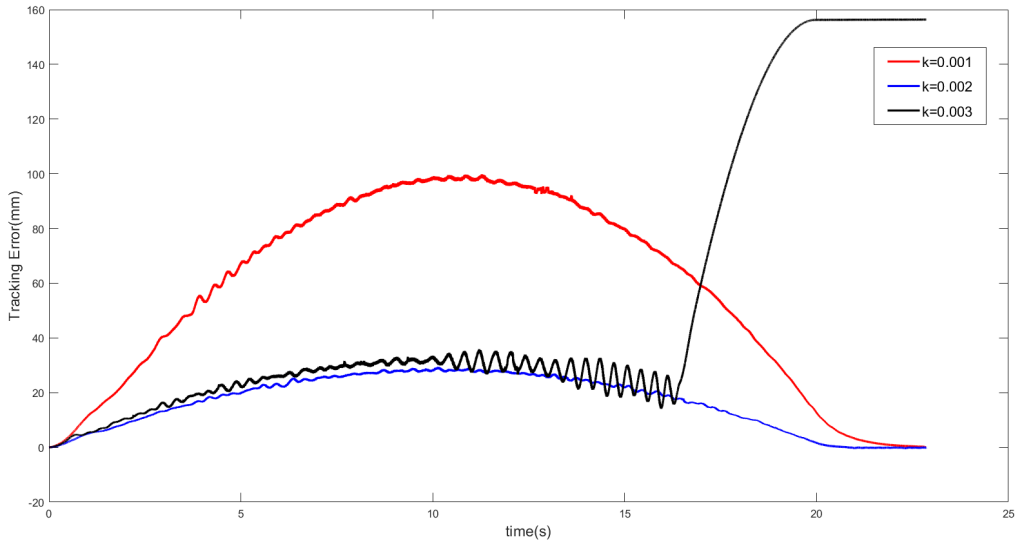
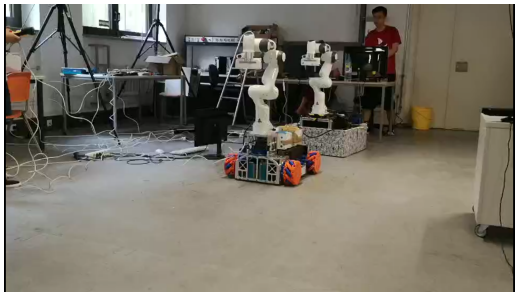


Figure 2.3: Tracking error measurement

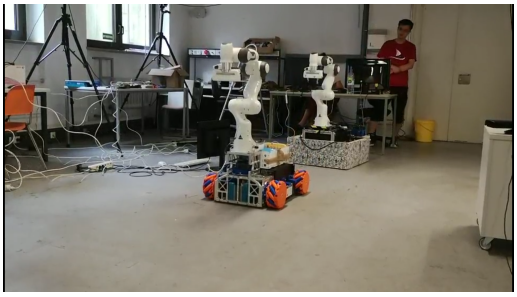
Figure 2.3 shows the results of our experiment. Comparing the results of $k_{pcl} = 0.001$(red curve) and $k_{pcl} = 0.002$(blue curve), we can observe that the static error

at the goal position($t > 20s$) is approximately equal to 0. That means the precision of the close-loop motion control is reliably high. Furthermore, compared with $k_{pcl} = 0.001$, the maximum tracking error(approx. 25mm) in the case of $k_{pcl} = 0.002$ is notably lower. It shows that less tracking error is needed in this case to drive the mobile robot. It is worth noting that in the measurement of $k_{pcl} = 0.003$, an intense oscillation existed in our mobile robot and the tracking error suddenly raised up to an abnormally high value of 156mm. We can consider that the value $k_{pcl} = 0.003$ is too big under our experiment condition which causes overshoot in our system and makes the control loop unstable. Therefore, we can conclude that $k_{pcl} = 0.002$ is the optimal value under our experiment condition.

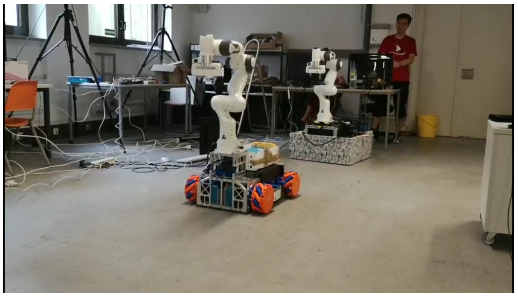
Using this close-loop control method, we can theoretically control the movement of the mobile robot in $x-$, $y-$direction and the rotation of yaw-angle. At the end of this chapter, we would like to show the real scene of motion control of the mobile robot. Figure 2.4 and 2.5 shows the straight-forward movement(along $x-$axis) and the self-spinning(the rotation of yaw-angle) respectively.



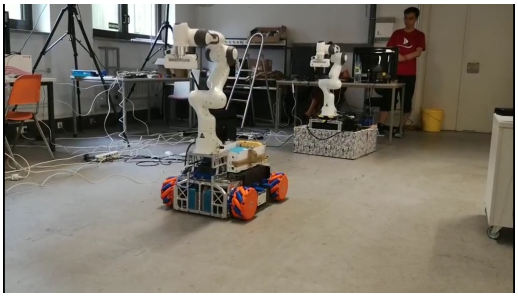
(a)



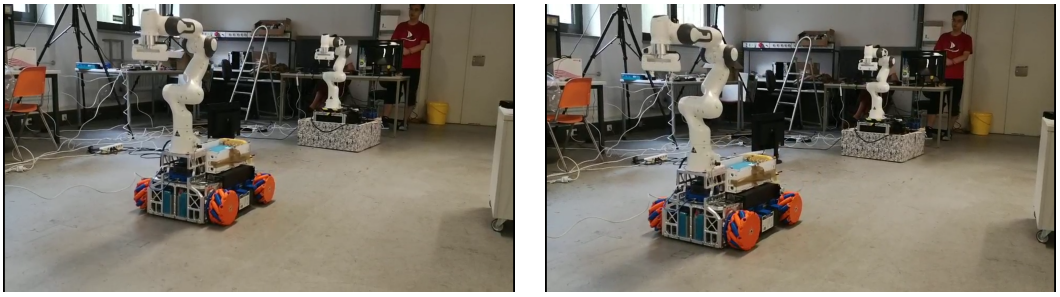
(b)



(c)

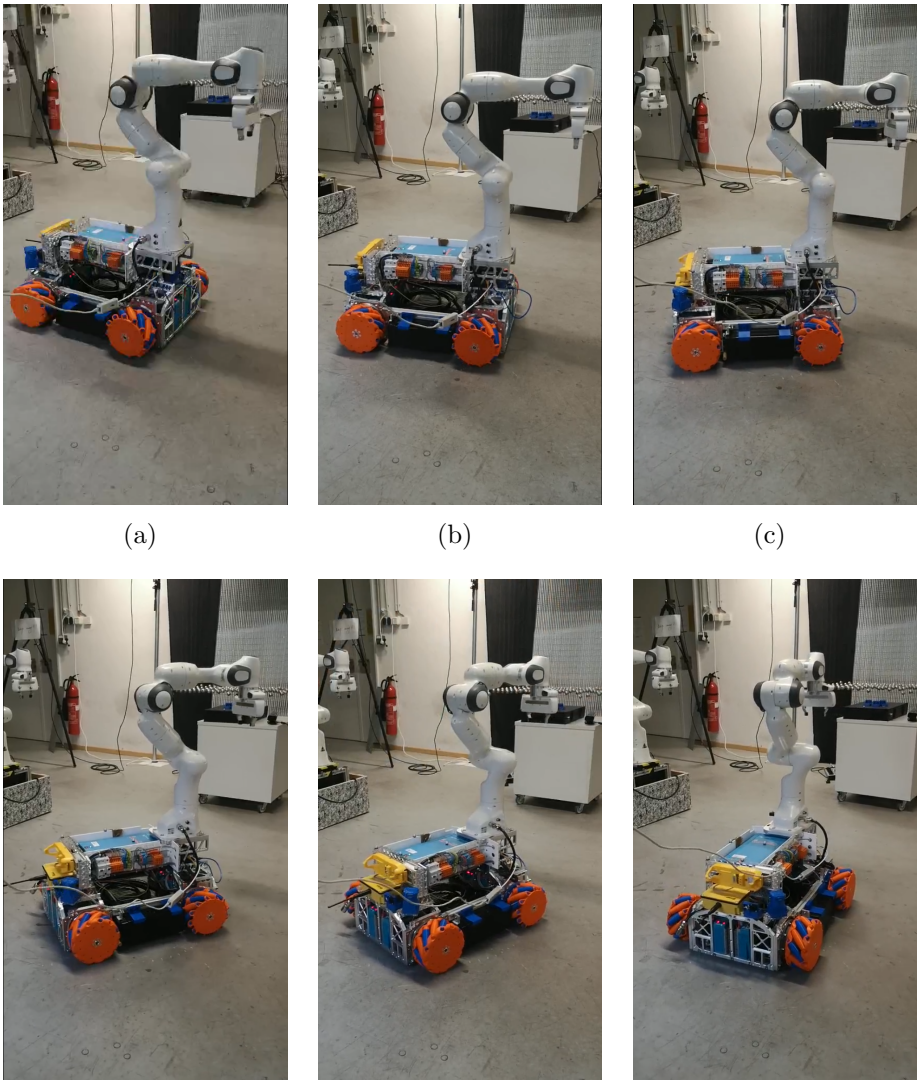


(d)



(e) (f)

Figure 2.4: Straight-forward movement of the mobile robot



(a) (b) (c)

(d) (e) (f)

Figure 2.5: Self-spinning of the mobile robot

Chapter 3

Cooperation between Multiple Robot Unit

Based on our previous work of the single mobile robot, we will move on to the study of multiple mobile robot system. In this task we have two *mofa* mobile robot with the same structure. The robot *mofa-1* is implemented with the close-loop motion control method which is introduced in chapter 2, while the planner node of the robot *mofa-2* is deactivated. That means we cannot directly control the motion of *mofa-2*. The goal of this task is to realize the distributed control of these two mobile robots. A simple example would be the relationship of leader and follower: let *mofa-2* follow the trajectory of *mofa-1* without motion planning. In order to realize this distributed control method, a trajectory message communication between the two mobile robots is needed.

Obviously, a follower always follows the same trajectory of the leader. The problem is that the initial positions and goal positions of the leader and follower are different. Therefore, our task can be concluded to the calculation of the trajectory of *mofa-2* based on the trajectory of *mofa-1*. For the calculation, besides the trajectory of *mofa-1*, we need to know the initial position offset between the two robots.

Figure 3.1 illustrates our trajectory communication method. The desired trajectory $x_{d1}(t)$ of *mofa-1* is designed by the planner node and published to the local area network in real-time. Besides, the initial positions of both two robots can be detected by the qualisys system. For the calculation of the desired trajectory $x_{d2}(t)$ of *mofa-2*, we need to send the information of $x_{d1}(t)$ and the initial position x_{01} of *mofa-1* to the control package of *mofa-2*. The trajectory $x_{d2}(t)$ can be calculated based on $x_{d1}(t)$ and the initial position offset:

$$x_{d2}(t) = x_{d1}(t) - (x_{01} - x_{02}) \tag{3.1}$$

It is worth noting that in our ROS program, if a message is lost or not publishing, the subscriber will receive numbers of 0. Therefore, if the nodes of *mofa-1* are

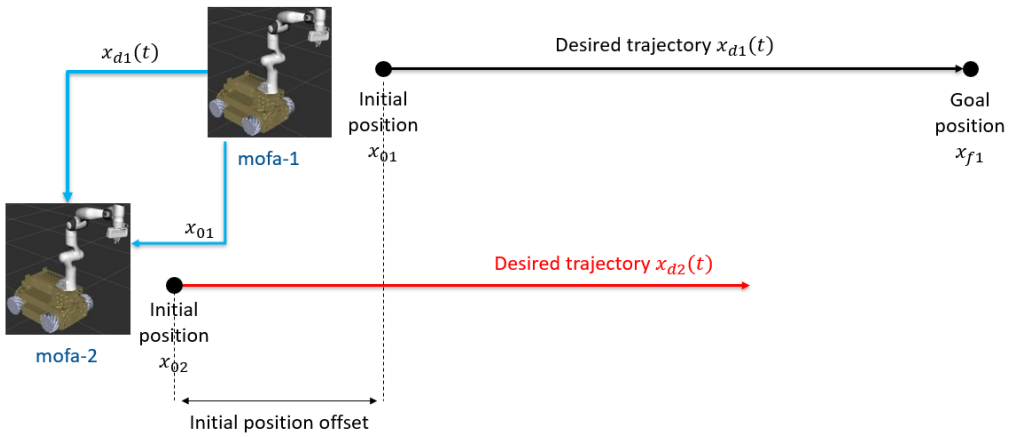
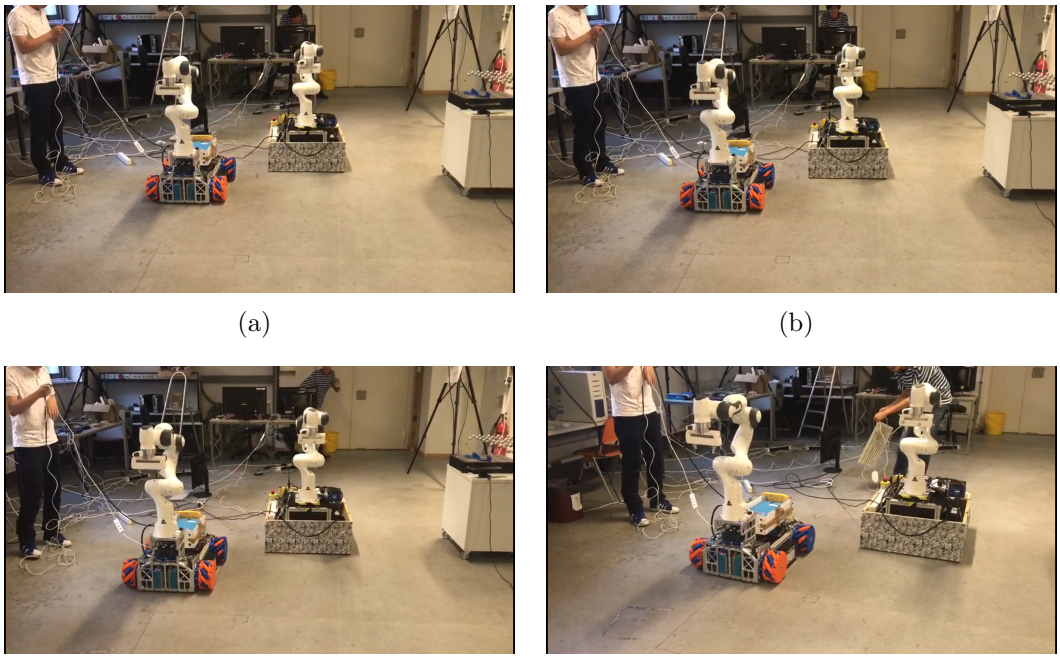


Figure 3.1: Trajectory communication between multiple mobile robots

initially not working, the result of equation (3.1) will become $x_{d2}(t) = x_{02}$. That means *mofa-2* will stay at its initial position. Therefore, the missing of *mofa-1* will not cause the calculation error of $x_{d2}(t)$ which could bring safety risks to the robots and users. For the same reason, each time when a new experiment is started, we should first launch *mofa-2*, and then *mofa-1* for the trajectory planning.

Figure 3.2 shows an example of the trajectory tracking experiment between two mobile robots. The robot in front is the leader *mofa-1* and behind is the follower *mofa-2*. In this experiment, *mofa-2* follows the trajectory of *mofa-1* and move 1m forward along the x-axis.



(a) (b)

(c) (d)

Figure 3.2: Trajectory tracking between two mobile robots

Chapter 4

Monitor part

4.1 Robot Model

To visualize the robot in the master controller , at first a virtual robot model is needed. In the ROS, the Unified Robot Description Format (URDF) is an xml specification to describe a robot. In this project we also use this xml file but a effective one “xacro”. With “xacro”, we can construct shorter and more readable xml files by using macros that expand to larger xml expressions.

Our robot Mofa consists of two parts: franka arm and the moving platform. In the beginning we should define them individually. A robot tree structure is needed to determine the important parts of the robot as links and joints. Figure 4.1 shows the moving platform structure.

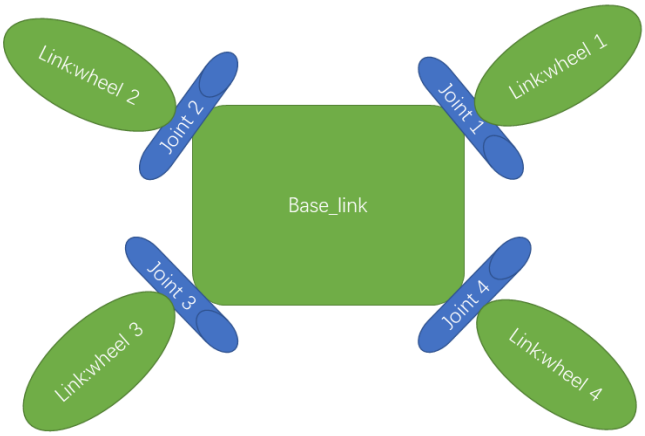


Figure 4.1: platform structure

The link element describes a rigid body with an inertia, visual features, and collision properties. Here for the moving platform the links are the basic body of the robot and the four wheels. The joint element describes the kinematics and dynamics of

the joint and also specifies the safety limits of the joint. Which means the joint describes the relationship between links. As the figure 4.2 shows below, the joint shows the transformation between parent link and child link.

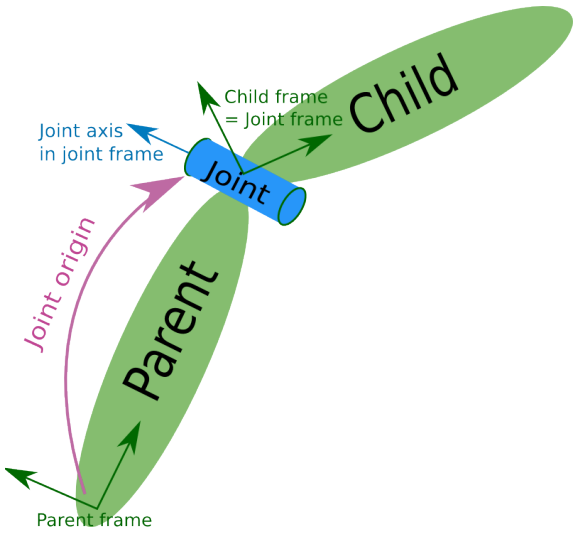
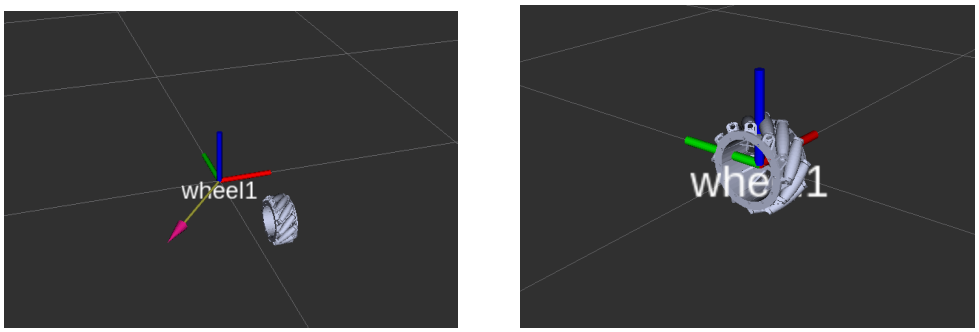


Figure 4.2: joint structure

After knowing the basic structure of generating a robot model, we should do the transformation between relative coordinate frames, as in the robot model the variable coordinate frame directly shows the changes when the robot moves. From the superviser we get the 3-D model of the moving platform. But for some reasons we should first normalize the coordinate frame for the wheels because like the following figure 4.3 shows the origin of the wheel is not the center of mass. To generate a kinematic model, the continuous joint(i.e joint of the wheels) must rotate around the axis through the origin of the wheel.

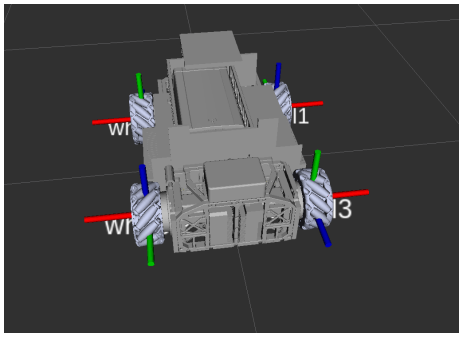


(a) wheel initial (b) wheel transformed

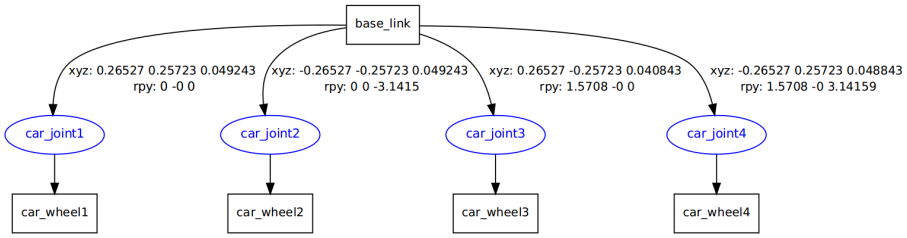
Figure 4.3: wheel part

Similarly we do the transformation of each wheel and connect the wheels with the basic part. And the TF structure shows the relationship between each link. Accord-

ing to figure 4.4, the whole part of the moving platform is shown with the frame structure.



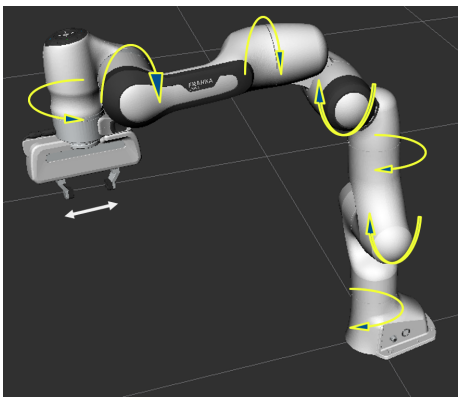
(a)



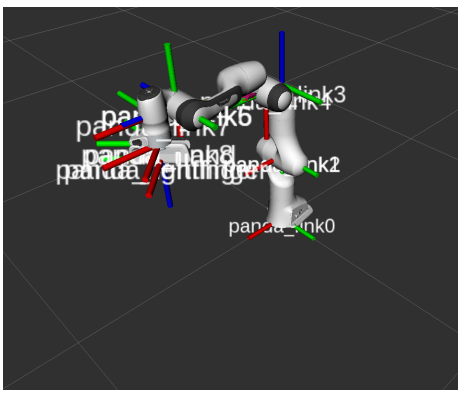
(b) platform structure

Figure 4.4: moving platform

Another robot part is the franka arm , similarly to describe the franka arm we just need to do the proper coordinate frame transformation between each basic link. Compared to the moving platform, the joint of each arm should be revolute , because it is like the arm of human , it should have a upper and lower limit, otherwise it will break down. The figure 4.5 shows the model and its coordinate frame.



(a)



(b)

Figure 4.5: franka arm

After generating these two parts, we should visualize them on our master controller, as we use the ROS, a visual tool called RViz is provided to help visualize the virtual robot model. In ROS, RViz is based on a important package TF, which maintains the relationship between coordinate frames in a tree structure buffered in time, and lets the user transform points, vectors, etc between any two coordinate frames at any desired point in time. At the same time RViz shows the changes visually. Therefore, we can conclude that in order to visualize the robot model we have to build a appropriate TF tree structure. Because our two individual robot are now independent with each other, there is no same reference coordinate frame for them. From the figure 4.6 we can observe, similar to the real robot, the franka arm is fixed on the front part of the basic body of the moving part. A fixed joint is created to connect these two parts.

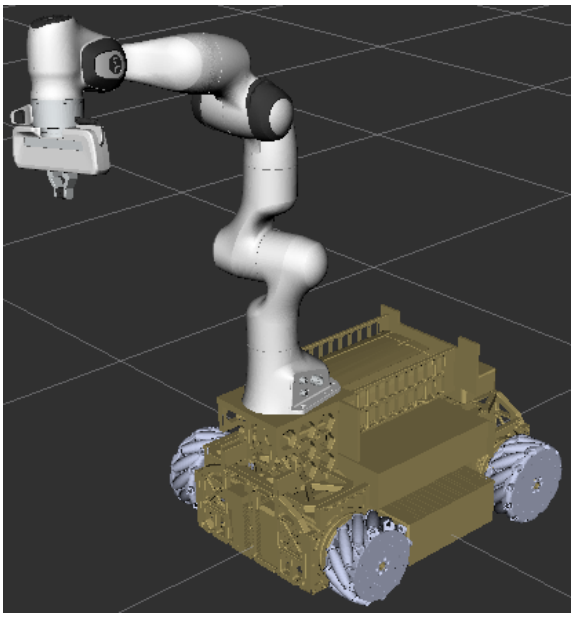


Figure 4.6: Mofa

4.2 Single robot monitor

As is mentioned before, RViz is based on the TF package i.e. the coordinate frame structure. In order to monitor the robot, it's also important to build the connection of the robot state with the real robot. In the previous chapter we have introduced that the robot has a marker to locate itself in the whole qualisys system. Since the network is already built, what we should do is to get the current pose from the real robot. Afterwards through the node that we generate ourselves we can finish the task to track the current pose of the robot, Upon the current poses is given, the current TF tree structure can be updated through the transformation in time. As

a result, we can directly observe the changes of the robot state on the computer in time as well.
Before connecting the virtual robot with the current poses. We need to test the node that we create. The figure 4.7 below shows how the each node work with each other for a basic monitor testing part. The node “joint_state_publisher ” and “robot_state_publisher” are the nodes that the ROS contains itself while the “state_publisher” is the node that we generate to determine the state message and send them to the TF tree.

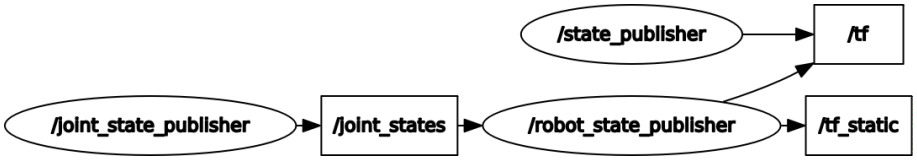
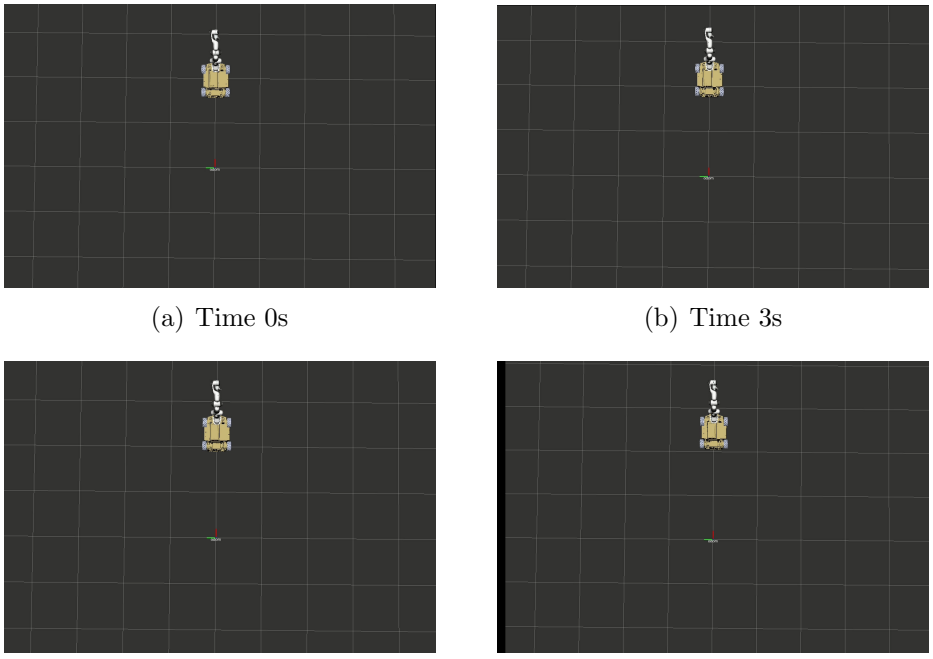


Figure 4.7: rosnode graph

Since our robot is based on the Mecanum wheels it can finish the job to move forward, self spin and move long the y axis. So when the position information is given it could be displayed on the computer to show the movement of these poses. After directly sending the position message through our node to the TF tree, we obtain the results. Figure 4.8 describes when mofa moves forward, then figure 4.9 shows when mofa moves along the y axis, at last according to the figure 4.10 we get the result when the robot spins itself.



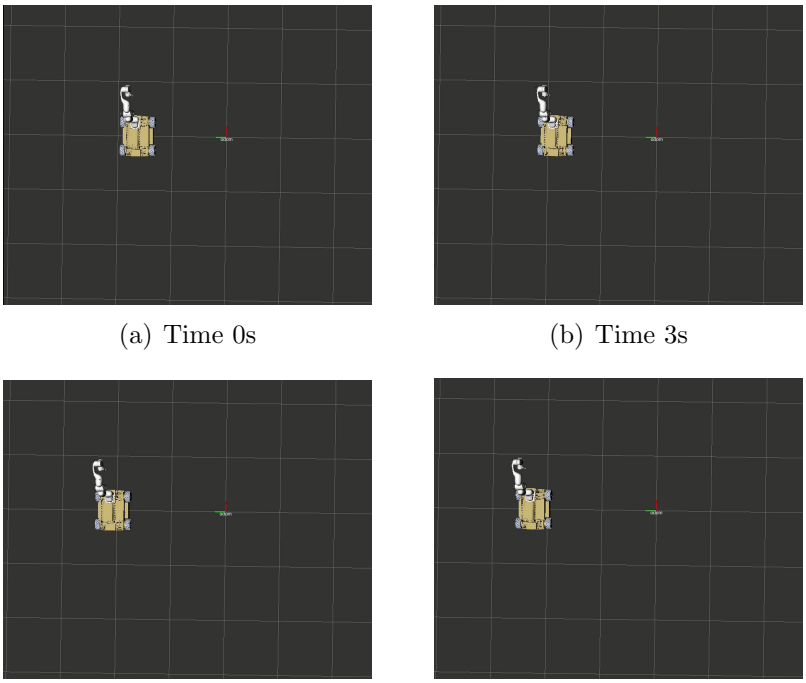
(a) Time 0s

(b) Time 3s

(c) Time 6s

(d) Time 9s

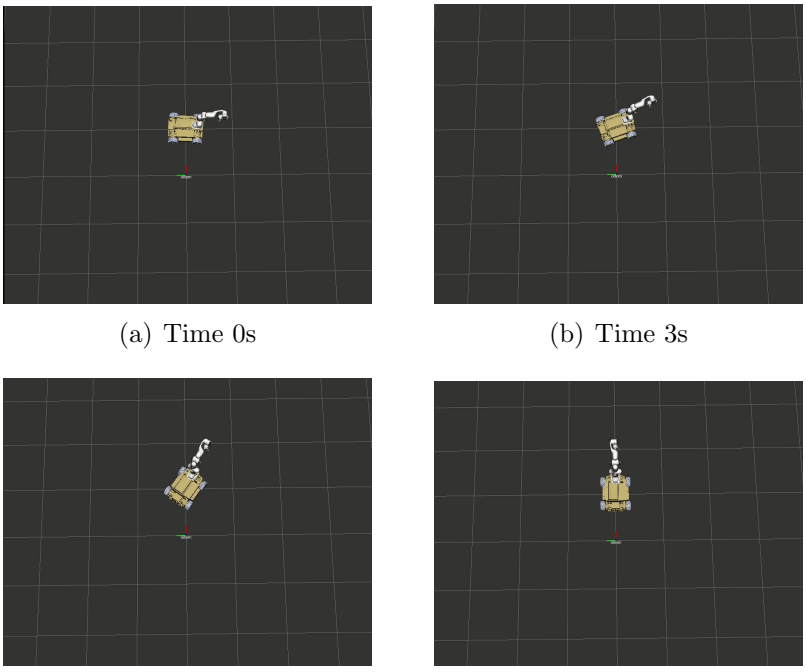
Figure 4.8: move forword



(a) Time 0s (b) Time 3s

(c) Time 6s (d) Time 9s

Figure 4.9: move along y axis

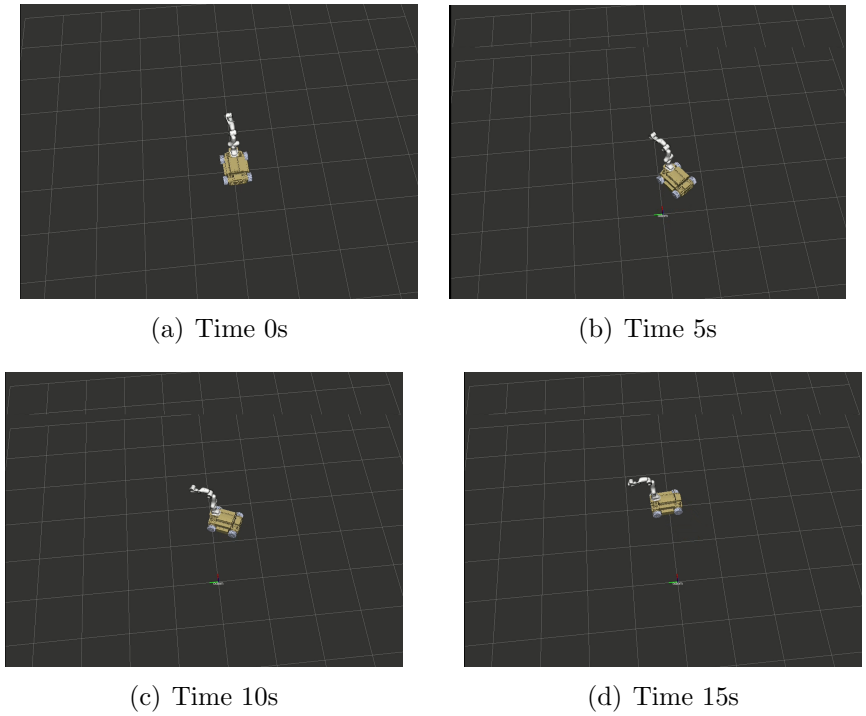


(a) Time 0s (b) Time 3s

(c) Time 6s (d) Time 9s

Figure 4.10: self-spinning

Furthermore when giving a random trajectory of position to TF, we can observe from the RViz that the robot move properly along the trajectory. The result can be seen from the figure 4.11.



(a) Time 0s (b) Time 5s

(c) Time 10s (d) Time 15s

Figure 4.11: move with trajectory

Finishing the test part, we build the connection with the real robot. As the geometrical messages from the qualisys contains position information of three axis and the orientation , we can use them directly to do the coordinate transformation between the qualisys and the robot. The following figure 4.12 shows how the node“state_publisher” works when connecting with the hardware.

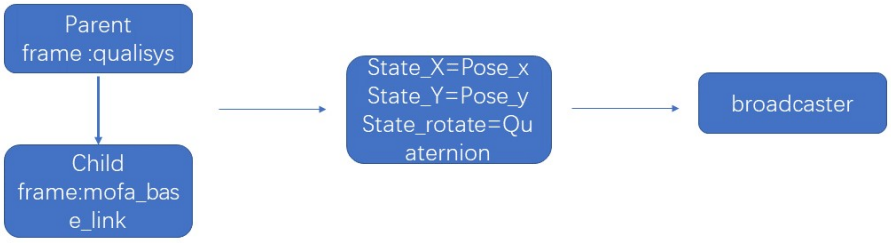


Figure 4.12: state publisher

What's more, through the kinematic analysis of Mecanum wheels mentioned before. The joint state can be also generated. With current messages subscribed from qualisys, we can easily calculate the speed of the robot, then with the help of Jacobian inverse matrix (Equation 1.3) we can obtain the angular speed of each wheel. Integrated with the time, the joint state of the angle the wheels rotate can be easily obtained.The figure 4.13 shows the when connecting with the hardware.

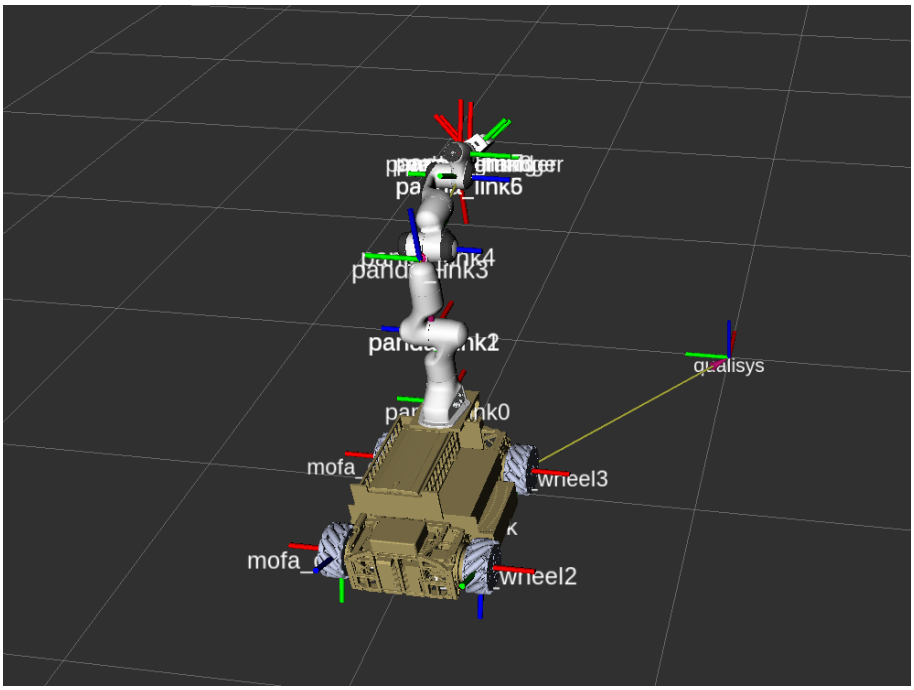


Figure 4.13: single robot with qualisys

4.3 Multi-robot monitor

When working with multi-robots , it not totally same with the single robots. One difficulty is to visualize the robots together. Because the robots are independent with each other, there are no reference coordinate frame for them. Although they are all based on the global coordinate frame, we can not describe the relationship when describe the robot as there is no link can be defined as a "coordinate frame". Moreover different from the relationship between franka arm and the moving platform. No joints could be connected directly between each individual robot.

An approach is to define a "dummy" link to help display the robots as we show in the figure 4.14. It is an empty link but all robots can connect with it, which means it establishes a "fake connection" between the robots. This approach is also based on the TF structure. In ROS the TF structure changes when the transformation happens. Hence, when the current state is updated the TF tree will be

reconstructed. At the same time the new relationship between the robots and the qualisys is established , the "dummy link " now makes no sense. Figure 4.15 and figure4.16 show the old and updated TF structure respectively.

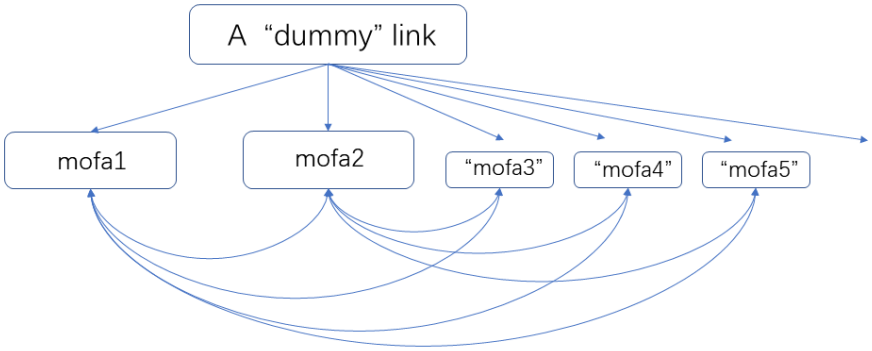


Figure 4.14: dummy link

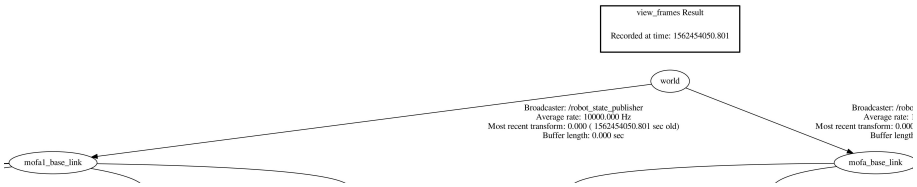


Figure 4.15: tf with dummy link"world"

Figure 4.16: reconstructured tf tree

Similarly , the total message transmission chart is shown below in the figure 4.17 when connecting with the hardware.

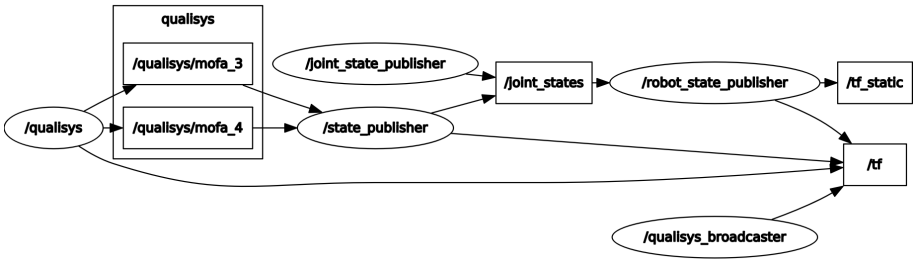


Figure 4.17: total node network

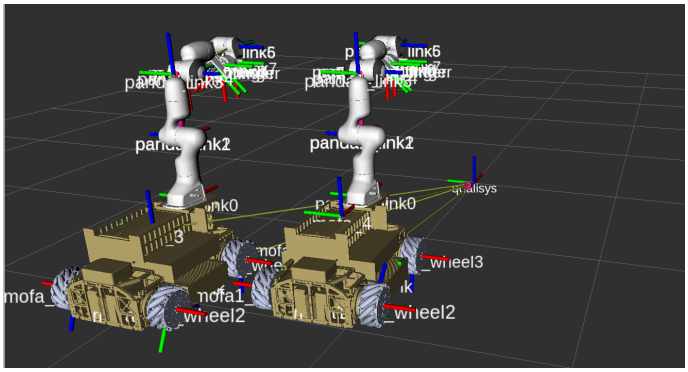
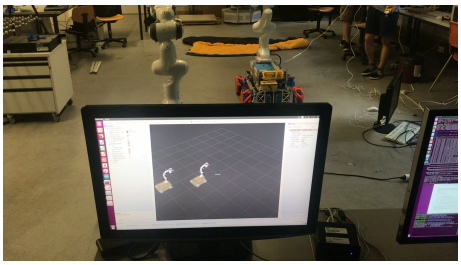
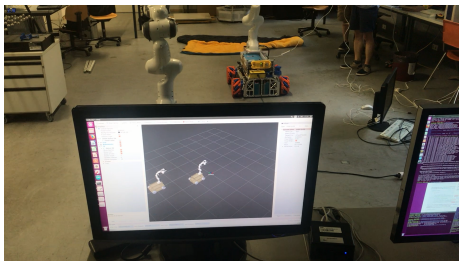


Figure 4.18: multi-robot with qualisys

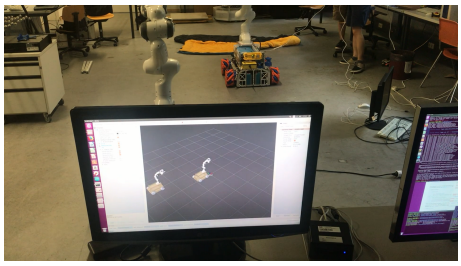
Finally, figure 4.19 shows the result how the virtual robot track the real robot.



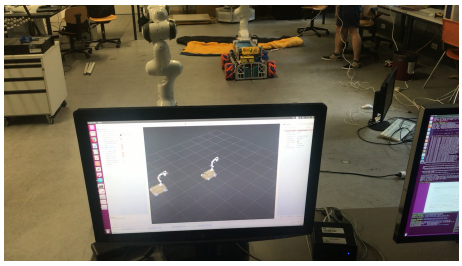
(a) Time 0s



(b) Time 3s



(c) Time 6s



(d) Time 9s

Figure 4.19: Monitor with real robots

Chapter 5

Discussion and Future Work

In this chapter, we will discuss some noteworthy factors that could influence the experimental results in previous tasks and conclude some possible future work about this topic.

5.1 Discussion

In our motion control method, we decided to use the cubic polynomial method which didn't consider the factor of the acceleration. In fact, the discontinuity of the acceleration could cause the robot shaking during the movement. Since the mobile robot moves in a very low speed in our experiment, the variation of the acceleration is hard to observe. But when the robot is moving faster, designing a trajectory function which contains smooth acceleration variation could be necessary. In this case, using the quintic polynomial method for motion planning would be a better solution.

Besides, since we are using the position data measured by the qualisys system for the close-loop motion control, the noise measured by qualisys could cause the robot shaking while the robot is launched or approaching the goal position. For noise cancelling, a digital filter can be adopted in the feedback of the position-loop.

For the monitor part, when we monitor the state of the real robot, the feedback of the joint state contains some noise caused by the qualisys, This kind of noise results in the unstable state of the virtual wheels. Same as the hardware part, a digital filter could be added to reduce the noise. Another problem is the sensor messages of the joint states may not transfer to the TF structure, when we transfer some the joint states using the namespace "joint state mofa". This will cause the virtual joint part stay static without moving. To solve this problem, one approach is to use the original message namespace(i.e. joint states)that the ROS already has, but it may cause problem if too many information of joints states are published at the same time. The other approach is to link the namespace to the TF structure.

5.2 Future Work

During the experiment, we discovered that the communication framework in multiple robot system relies on high real-time capability. For example, we have observed that the reading method of the initial position could cause the information of current position of the robot get lost in the first few loops when the node of *mofa* platform is launched. It is possibly because both of these two position informations come from the same qualisys message. This could cause delay when the program tries to read the initial position first and then the current position. If the information of current position is initially lost, the calculation result of the speed-loop would be extremely big($v = k_{pcl}(x_d(t) - x_{mofa})$ with $x_{mofa} = 0$). This could cause safety risks for both robots and users.

The same situation could happen when the qualisys system cannot detect the position of the mobile robot. In practical applications, the mobile robot might be occluded by obstacles or run out of the detectable area of the motion capture system. Therefore, we think that a possible direction of the future work could be the prevention of missing data.

Besides, since the mobile robot is able to move in a very high speed, the function of obstacle detection, which protects the safety of both robots and users, could also be an important task in the future. This function requires the robots implemented with sensors and the sensor feedback data properly integrated with the algorithm.

Appendix A

A.1 ROS part

A.1.1 Instruction

* SSH part:
  connection: *ssh user@ip-address*
  disconnection: *exit*
  shutdown: *sudo shutdown -h now*

* Catkin Workspace:
  source ~/test_ws/devel/setup.bash
  catkin_make clean
  export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:
  /user/workspace_name/package_name You can also add these commands into the /.bashrc file.
  locate the package: *rospack find package_name*
  change to the package library: *roscd package_name*

* ROS topic:
  rostopic list
  rostopic echo topic_name
  rostopic type topic_name
  rosmsg show message_name

* ROS launch:
  roslaunch mofa_platform_hw platform.launch
  roslaunch qualisys_hw_node qualisys.launch
  roslaunch planner qualisys_topic_sub.launch

* ROS data record:
  record data to rosbag file: *rosbag record -a*
  extract data to txt or csv: *rostopic echo -b file_name.bag -p /topic_name >txt_name.txt* or
  rostopic echo -b file_name.bag -p /topic_name >csv_name.csv

A.1.2 Common Errors

* catkin_make errors:
  close the terminal, delete the respective build files, run the command *catkin_make clean*, then *catkin_make*.

* Can't locate the package:
  Run the source command, if it doesn't work, add the command of the path to the ./bashrc file, as mentioned above.

A.2 Monitor part

* Launch monitor part

  command: cd manage_monitor

  "source ~/manage_monitor/devel/setup.bash"

  roslaunch rviz_visual dis.launch

  When the rviz is launched, kick the "add" button and add the " Robot Model". Then choose the proper fixed frame "qualisys" .

* When error "Resource not found" happens, first check if you source the "setup.bash" file in the workspace. Then check whether your directory is a package or not.

* Change the "xacro" file.

  If you want to add more virtual mofa in the system, first you should copy a "xacro " file called "mofa.xacro", "hand.xacro" and "panda_arm.xacro" and then change the namesapce "mofa" to "mofa(x)","hand" to "hand(x)"and "panda_arm" to "panda_arm(x)" . Then in another file called "mofa_panda" you should add some links as the annotation shows.

* Check the state message from qualisys

  command: rostopic echo /qualisys_mofa(x)

List of Figures

Bibliography

[CR17] Frank Kuenemund Christof Roehrig, Daniel Hess. Motion controller design for a mecanum wheeled mobile manipulator. *Proceedings of the 2017 IEEE Conference on Control Technology and Applications (CCTA 2017)*, pages 444–449, 08 2017.

[Par08] Lynne Parker. *Multiple Mobile Robot Systems*, pages 921–941. 01 2008. doi:10.1007/978-3-540-30301-5_41.

[SHV05] M. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. JOHN WILEY & SONS, INC., 1 edition, 2005. p.173-174.

[WLW13] H. Wu, N. Luo, and C. Wang. Study on control strategy of the rotary synchronous fixed-length cutting system. *Journal of Vibroengineering*, 15:713–725, 06 2013.

[Zha18] Xuhui Zhao. Derivation and relationship between the euler angle, quaternion number and rotation matrix. https://zhaoxuhui.top/blog/2018/03/13/RelationBetweenQ4&R&Euler.html, 05 2018.