DEGREE PROJECT IN ELECTRICAL ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Cooperative Transportation of Mobile Manipulators With Collision Avoidance

YU WANG

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Cooperative Transportation of Mobile Manipulators With Collision Avoidance

YU WANG

Master Program in Embedded Systems (120 Credits)
Date: May 22, 2018
Supervisor: Alexandros Nikou and Christos Verginis
Examiner: Dimos Dimarogonas
School of Electrical Engineering and Computer Science

Abstract

In this work, we propose two Nonlinear Model Predictive Control (NMPC) schemes, Decentralized Nonlinear Model Predictive Control (DNMPC) and Centralized Nonlinear Model Predictive Control (CNMPC), for cooperative transportation of an object, which is rigidly grasped by N agents. We use feedback linearization and model reduction to reduce overall complexity of the model. We also provide a mathematical proof for feasibility and convergence of the schemes. Finally, we use simulations and experiments to validate the robustness and efficiency of the proposed schemes.

Sammanfattning

I det här arbetet föreslår vi två system för icke-linjär model predictive control (NMPC), Decentraliserad icke-linjär MPC (DNMPC) och centraliserad icke-linjär MPC (CNMPC), för kooperativ transport av ett objekt, som är fast greppad av N agenter . Vi använder feedbacklinjärisering och modellreduktion för att minska modellens övergripande komplexitet. Vi tillhandahåller också ett matematiskt bevis för genomförbarhet och konvergens av systemen. Slutligen använder vi simuleringar och experiment för att validera de föreslagna systemens robusthet och effektivitet.

Acknowledgements

First of all, I would like to express my deepest gratitude to my supervisors, Alexandros Nikou and Christos Verginis as well as my examiner, Dimos Dimarogonas. Thank you for giving me such an interesting master thesis topic. I really appreciate your guidance, time and patience. I learn a lot in the process.

Furthermore, I would like to thank Pedro Roque for his assistance in experiments. Without your great help, I cannot finish experiments so successfully and quickly. I also want to express my gratitude to Matteo Mastellaro for your introduction of your previous work. Also, I want to thank my colleagues in Smart Mobility Lab Kuba Nowak, Philipp Ro, Olof Forsberg, Lukas Schönbächler, Stefano Mariuzzi, Umar Chughtai, Clara Cavaliere and Arianna Marangon for your companion and encouragement. I also would like to thank Tianlong Du, Beichuan Hong and Kai Chu for your encouragement.

I want to express my special gratitude to Xiao Chen and Frank Jiang. Your companion and encouragement give me a very happy time in SML and our discussions give me a lot of inspiration. Our friendship is precious.

I also would like to express my gratitude of Tengfan Lin to be my opponent.

I want to dedicate this thesis to my grandparents. You are my first teachers and my childhood was happy and colorful because of you. And I want to dedicate this work to my parents for supporting my master study. Your love is the strongest support in my life.

During my master study in KTH Royal Institute of Technology, I learn a lot. I would like to thank every professor and teaching assistant in my master study. Also, I would like to thank every friend in Sweden and my old friends in China. You make my master study more colorful. Especially, I would like to express my gratitude to Harold Rene Chamorro Vera 'Rapp', Rueben Laryea and Yuchao Li. You always encourage me to chase my dream. You are always willing to discuss dream, study and future plan with me. Also, I would like to express my gratitude to Prof. Zhinan Zhou, Prof. Xiaojun Ban, Prof. Xiaoping Shi, Luyan Gao and Xiaomeng Qin for your advice when I applied for the master program in KTH.

Last but not least, I would like to express my fiancée, Chenxi Feng. Thank you for your understanding and support. The most romantic

thing in my life is falling in love with you. I cannot imagine my life without you. I love you!

Yu Wang
May 17, 2018
In Stockholm.

Contents

List of Figures

List of Tables

Part I

The Problem

Chapter 1

Introduction

With the development of robotic technology, robots are widely used in many different fields, such as automation, logistics, space exploration, military operations, home care and health care etc.. Robots are expected to assist people in more areas, especially in some dangerous areas.

Manipulator is one of the major robots that are used in industries nowadays. Applications of manipulators has been proved successfully in various areas, such as logistics, health-care, aerospace industry etc. [29, 51, 41, 8, 33]. In 2016, a manipulator which consists of an industrial manipulator, a 3D camera and a gripper developed by Robotics Institute in Delft University of Technology won the Amazon Picking Challenge 2016 [29], due to its excellent performance in both speed and precision. By using deep learning and other artificial intelligent technology for object recognition, grasp planning and motion planning, the manipulator is proved to be reliable in both picking tasks and stowing tasks. By using manipulators in logistics, companies can greatly improve speed of service and reduce operating costs. Fig. 1.1a[1] shows team Delft's robot in the Amazon Picking Challenge 2016.

In health-care area, manipulators are majorly used in prosthesis [41, 51] and tumor diagnosis [8]. For disabled people, limb prosthesis can significantly improve their quality of life [51]. By combining manipulators with advanced image processing technique, diagnosis and early treatment of cancer become possible. Fig. 1.1b[2] is a prosthesis

[1] https://www.tudelft.nl/en/2016/tu-delft/team-delft-wins-amazon-picking-challenge/

[2] http://www.mobiusbionics.com/luke-arm/#section-two

called the LUKE arm, which consists of 10 powerful joints with different configurations. It can grasp different objects according to different needs of customers. Fig. 1.1c [3] is an Image-Guided Autonomous Robot (IGAR) manipulator developed by the Centre for Surgical Invention and Innovation (CSii) in Canada, which can provide highly accurate and minimally invasive procedure in early detection and treatment of breast cancer.

Manipulators also act an important role in space exploration and aerospace industry. With the help of mobile manipulators which are designed to hold and maneuver scientific instruments, NASA's Mars rover Curiosity is able to collect rocks and soil on Mars [58]. Fig. 1.1d[4] is a figure of the extended manipulator of NASA's Mars rover Curiosity captured by Navigation camera (Navcam) on August, 2012. Mobile manipulators are developed according to the needs in aerospace industry. Assemble tasks can be done by an aerial manipulator consisting of an unmanned aerial vehicle equipped with a robotic multi-link arm [33].

1.1 Motivation

Although single manipulators have been widely applied in several fields, they still have limitations when they are applied in real life.

One of the major disadvantages is lack of mobility. The combination of manipulators and mobile bases can improve mobility greatly. Mobile manipulators are manipulators combined with mobile bases, which can be divided into Mobile Manipulating Unmanned Aerial Vehicles (MM-UAV), Mobile Manipulating Unmanned Ground Vehicles (MM-UGV) and Mobile Manipulating Unmanned Underwater Vehicles (MM-UUV).

MM-UAV can be applied in both civilian and military operations. Due to its quickness and high efficiency when travelling long distance, MM-UAV can be used in disaster response, cargo resupply, material handling for infrastructure repair and construction etc.[12, 13]. Fig. 1.2a[5] is a small quadrotor UAV with mounted flexible manipulator in project called AVEMA in University of Luxembourg, which is a project focus-

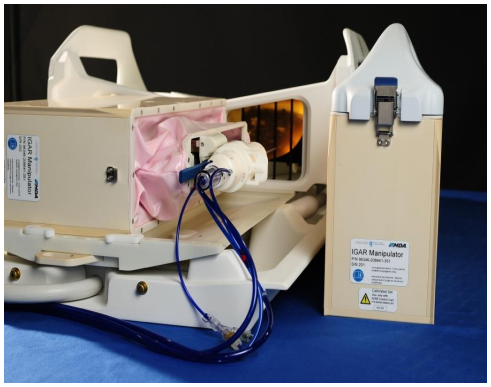
[3]https://www.nasa.gov/station/research/news/igar/

[4]https://mars.nasa.gov/msl/mission/rover/arm/

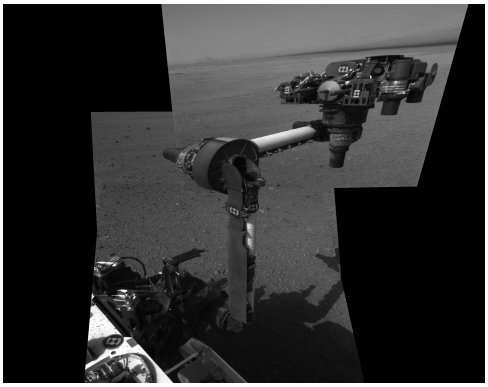
[5]https://wwwfr.uni.lu/snt/research/automation_robotics_research_group/projects/avema

(a) Team Delft's robot in the Amazon Picking Challenge 2016

(b) the LUKE arm



(c) IGAR manipulator



(d) the extended manipulator of NASA's Mars rover Curiosity

Figure 1.1: Applications of manipulators

ing on flying indoor manipulation systems. As is shown in Fig. 1.2b[6], researchers at the German Aerospace Center have integrated an seven-degree-of-freedom robotic gripper arm into an autonomous helicopter system, which can be applied to inspect and service robots on the pipelines without risk.

MM-UGV can be applied in security, defence, law enforcement, agriculture, logistics etc.. By employing UGV, manipulators can carry out various missions on different terrains. Fig. 1.3a[7] is a mobile manipulator mainly for indoor tasks. Fig. 1.3b[8] is a MM-UGV which is suitable for dangerous duty operations on battlefields. Fig. 1.3c[9] is a rescuer mobile manipulator which is able to conduct rescuer missions under the presence of toxic industrial agents such as nuclear, radiological, biological and chemical materials in the environment. The main challenge of MM-UGV is how to keeping balance on irregular landscapes. Nowadays, many researchers are dedicated to improving this. BostonDynamics has developed several UGV, which are able to keep balance on irregular landscapes. Fig. 1.3d[10] is Spotmini developed by BostonDynamics, which is able to keep balance on irregular landscapes and recover from failure.

MM-UUV have gained increasingly attention in recent years [37]. They can be operated remotely to inspect and repair underwater cables and pipes. They can also conduct some search and rescue missions. The main challenges of MM-UUV are limited communication with underwater systems and the harsh underwater environments such as invisibility, non-oxygen environment and high underwater pressure etc.. Fig. 1.4 are examples of MM-UUV. Fig. 1.4a[11] shows a autonomous light intervention vehicle called ALIVE developed by Cybernetix, which is able to perform light interventions on deep-water sub-sea facilities. Fig. 1.4b[12] is Seaeye Cougar-XT developed by SAAB group with

[6]https://informedinfrastructure.com/20562/flying-robots-inspect-and-maintain-inspection-robots/

[7]https://www.robotnik.eu/manipulators/x-wam/

[8]https://www.ecagroup.com/en/solutions/cameleon-e-ugv-unmanned-ground-vehicle

[9]https://www.robotnik.eu/services-robotic/mobile-robotics-applications/security-defense/

[10]https://www.bostondynamics.com/spot-mini

[11]http://auvac.org/configurations/view/15

[12]http://www.seaeye.com/cougar-XT.html

(a) A small quadrotor UAV with mounted flexible manipulator

(b) An autonomous helicopter system equipped with an seven-degree-of-freedom robotic gripper arm

Figure 1.2: Examples of MM-UAV

working depth 2000 meters. Fig. 1.4c[13] is a Girona 500 equipped with a 4 degree of freedom manipulator, which is able to conduct long-term missions. Researchers from Universitat Jaume have tested a MM-UUV, which can develop underwater intervention tasks as is shown in Fig. 1.4d[14]. It recovered an object which is similar to an aircraft black box successfully in the test.

Other drawbacks of single manipulators are their limitation on payload and lack of flexibility, which make single manipulators cannot fulfill needs of difficult tasks involving heavy payloads and challenging maneuvers. One possible solution is employment of multiple robots [45], i.e multi-agent systems.

In the past years, multi-agent systems have gained a great amount of attention from robotic community [15, 32, 46]. Multi-agent systems are defined as a collection of, possibly heterogeneous, computational entities, having their own problem-solving capabilities and they are able to interact in order to reach an overall goal [47]. The main challenge of cooperative control of multi-agent systems is how to derive desirable collective behaviours through the design of individual agent control algorithm [53].

Compared with single-agent systems, multi-agent systems has the following advantages in terms of transportation of robotic manipula-

[13]http://persistentautonomy.com/

[14]https://www.sciencedaily.com/releases/2011/05/110517074558.htm

(a) Mobile manipulator X-WAM

(b) Cameleon E

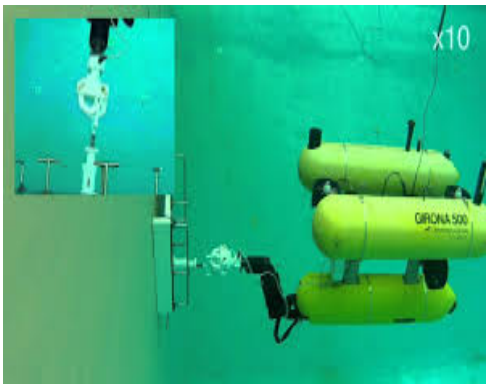
(c) Rescuer mobile manipulator

(d) Spotmini developed by BostonDynamics

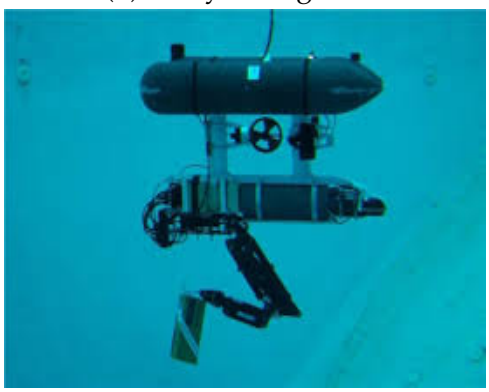
Figure 1.3: Examples of MM-UGV

(a) ALIVE MM-UUV

(b) Seaeye Cougar-XT

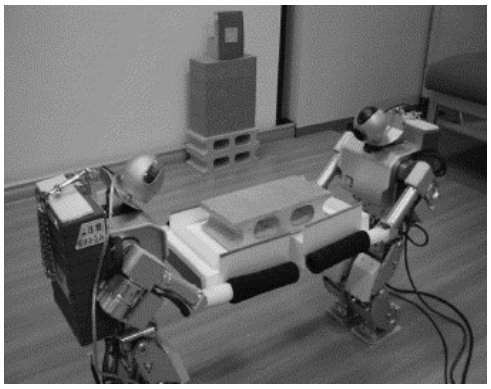


(c) Girona 500 equipped with a 4 degree of freedom manipulator

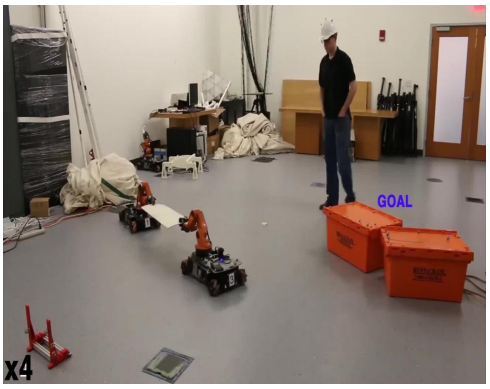


(d) A MM-UUV developed by researchers from Universitat Jaume

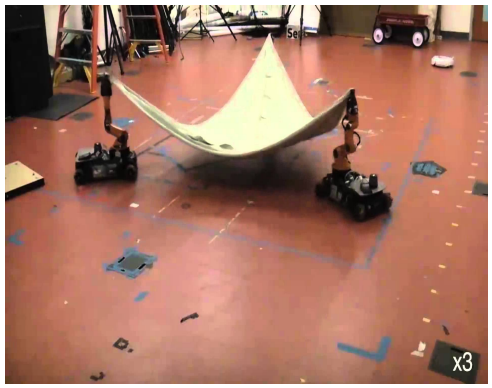
Figure 1.4: Examples of MM-UUV



(a) Cooperative transportation system for humanoid robots



(b) A team of Kuka Youbot mobile manipulators holding a rigid object



(c) A team of mobile manipulators holding a flexible object

Figure 1.5: Examples of cooperative manipulators

tors. First of all, with the employment of multiple robots, the payload can be shared, which makes carrying heavier payload possible. Secondly, the information of systems is distributed in each agent, which means multi-agent systems can provide opportunities for real-time adaptation. Furthermore, the employment of multiple mobile manipulators can improve robustness to dynamic uncertainties. Thus, cooperative transportation of manipulators has gained a great amount of attention [1, 31, 2]. Fig. 1.5a [31] depicts a cooperative transportation system for humanoid robots. Fig. 1.5b [1] shows a team of Kuka Youbot mobile manipulators which holds a rigid object. Fig. 1.5c [2] shows a team of mobile manipulators holding a flexible object.

This thesis focuses on cooperative transportation of manipulators on Unmanned Ground Vehicles.

1.2 Literature Review

This section reviews the previous works about cooperative manipulations, model predictive control and application of model predictive control in cooperative manipulations.

1.2.1 Cooperative Manipulations

In previous works [52, 38, 39, 63, 35, 5, 26], the control architectures of cooperative manipulators allow robot agents to communicate and share information with each other. Other control architectures of cooperative manipulators in earlier works are completely decentralized schemes, which means each agent can only uses local information and observers. Such completely decentralized schemes can avoid potential communication time delay between agents.

Authors in [52] design an object impedance controller for cooperative manipulation. The object impedance controller sends commands to each arm respectively and each arm will feed its states using its local sensors. It allows the arms to run in parallel and avoid time delay caused by communication between agents.

In [38], two decentralized cooperation controllers are proposed for trajectory tracking of two manipulators handling an object. One controller requires force sensing to sense the interaction force between the manipulator and the object, while the other one can track trajectory without force sensing. Both of the controllers have no requirements on the communication between manipulators. In [35], a decentralized control structure for cooperative manipulators is also designed, where each agent has real-time access only to its own state information.

Authors in [39] also propose decentralized controllers for cooperative manipulators. one adaptive and two non-adaptive decentralized controllers. The adaptive controller is robust under physical parameter uncertainties, while the non-adaptive decentralized controllers can guarantee convergence rate and transition performance. Adaptive control for multiple cooperative robot arms can also be found in [63].

In [26], a control law without velocity measurements is proposed, which can reduce the complexity of integrating a large amount of sensors.

Impedance, force and motion control are the most common techniques used in early works [52, 6, 28, 17, 18, 56, 59, 19, 48, 27]. How-

ever, most of them need to acquire contact manipulator-object forces or torques. The performance of the controller might be declined due to sensor noise. Fortunately, according to [10], a certain object can be grasped rigidly by manipulator grippers, which makes some usage of force or torque sensors unnecessary.

One important concern about cooperative manipulations is collision avoidance. It is very common that there are some obstacles on the path of robots. Most of the existing approaches to collision avoidance with obstacles are using over-actuated robots, i.e exploiting extra degree of freedom to avoid obstacles [44, 16, 9, 52]. However, when the agents are near obstacles, input control values are usually quite large, which means a risk of exceeding the limitations of actual motor input. Another effective way to avoid obstacles is using potential field-based methods [62, 57]. However, field-based methods might suffer from local minima.

Another importance concern about cooperative manipulators is singularity avoidance [11, 36, 61, 30]. Jacobian matrix maps the joint velocities of the agents into a six-dimension generalized velocities. Singularities of Jacobian matrix will make the joint velocities indeterminate. A kinematic singularity occurs where the robot's Jacobian matrix loses rank, which means robots will lose the ability to move in some directions. A representation singularity might occur when mapping from coordinate rates to angular velocities [45]. Apparently, singularity configurations should be treated as constraints and avoided.

1.2.2 Model Predictive Control

Model Predictive Control (MPC) is considered as the most successful control strategy in recent years, which has been adopted widely in both research and industries. MPC takes constraints into consideration explicitly. The control law is derived from solving optimal problem. MPC has been extended in nonlinear systems, hybrid systems etc. [4, 25].

Compared with classic control strategies, one of the advantages of MPC is that it can take constraints into consideration explicitly, which allows us to take collision and singularity avoidance into consideration as explicit constraints [50]. Another advantage of MPC compared with other traditional control strategies is it provides the optimal solution.

Due to the nature of nonlinearity of dynamic equation of manipulators [54], the systems of mobile manipulators are nonlinear systems, which means MPC for manipulator is Nonlinear Model Predictive Control (NMPC). However, compared with Linear Model Predictive Contorl (LMPC), NMPC is no longer convex optimization problem and computational burden increases greatly [25]. How to reduce computational burden is important for NMPC especially for real-time implementation.

1.2.3 Application of MPC on cooperative manipulations

MPC has been applied in cooperative manipulations in recent years [45, 20]. MPC shows good performance in cooperative manipulations.

In [45], the authors proposed a Centralized Nonlinear Model Predictive Control (CNMPC) scheme for cooperative manipulation with singularity and collision avoidance, which is suitable for the cooperative manipulation of an object rigidly grasped by several robotic agents. The proposed scheme is able to navigate the object to a final pose while avoiding singularity as well as collision with obstacles. The simulation results shows the effectiveness and convergence of the proposed scheme. However, the complexity of the proposed scheme needs to be reduced.

The authors in [20] proposed a Decentralized Nonlinear Model Predictive Control (DNMPC) scheme under the presence of disturbances and uncertainties. The scheme is able to avoid collision with obstacles while maintaining connectivity between robotic agents. The DNMPC scheme is proved to be robust under disturbances and uncertainties by simulation.

The previous works above show the possibility to apply MPC on cooperative manipulations. There are also possibilities to reduce complexity and computational burden of MPC.

1.3 Notations and Acronyms

This section introduces some notations and acronyms used in this thesis.

1.3.1 Notations

Table. 1.1 and Table. 1.2 shows some notations and constants used in this thesis. Other complicated notations are defined as follows.

Consider two sets S_1 and S_2. The set difference, which is defined by $S_1 \setminus S_2 = \{s : s \in S_1, s \notin S_2\}$, is denoted by $\setminus$. The Minkowski addition is denoted by $\oplus$, which is defined by $S_1 \oplus S_2 = \{s_1 + s_2 : s_1 \in S_1, s_2 \in S_2\}$.

The vector connecting the origins of coordinate frames $\{A\}$ and $\{B\}$ expressed in frame $\{C\}$ coordinates in 3D space is denoted as $p^C_{B/A} = [x_{B/A}, y_{B/A}, z_{B/A}]^\top \in \mathbb{R}^3$. $\eta_{B/A} = [\phi_{B/A}, \theta_{B/A}, \psi_{B/A}]^\top \in \mathbb{T}^3 \in \mathbb{R}^3$ denotes the x-y-z Euler angles representing the orientation of frame $\{A\}$ with respect to frame $\{B\}$, where $\phi_{B/A}, \theta_{B/A} \in [-\pi, \pi]$ and $\psi_{B/A} \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. The angular velocity of frame $\{B\}$ with respect to frame $\{A\}$ expressed in frame $\{C\}$, is denoted as $\omega^C_{B/A} \in \mathbb{R}^3$.

$R^B_A \in SO(3)$ is the rotation matrix from frame $\{B\}$ to frame $\{A\}$. If there is a vector $a \in \mathbb{R}^3$, $S(a)$ is the skew-symmetric matrix which is defined by $S(a)b = a \times b$. The rotation matrix and angular velocity have the following relationship $\dot{R}^B_A = S(\omega^A_{B/A})R^B_A$.

$\mathcal{N} = \{1, ..., N\}$ is the set of agents $1, .., N$ in a multi-agent system.

An ellipsoid in three-Dimensional space can be represented by $\mathcal{O}_z = \mathcal{O}(c_z, \beta_{1,z}, \beta_{2,z}, \beta_{3,z}) = \{p \in \mathbb{R}^n : (p - c_z)^\top P(p - c_z) \leq 1\}$, where $c_z \in \mathbb{R}^3$ is the center of the ellipsoid, $\beta_{1,z}, \beta_{2,z}, \beta_{3,z} \in \mathbb{R}^1_{>0}$ are the lengths of three semi-axes of the ellipsoid and $z \in \mathbb{N}$ is the index of ellipsoids.

To simplify notation, if a coordinate frame corresponds to an inertial frame of reference $\{I\}$, the explicit notation will be omitted.

1.3.2 Acronyms

For simplicity, the following abbreviations are used in this thesis. Table. 1.3 shows the abbreviations and their corresponding meanings.

1.4 Outlines

This thesis is divided into four parts. Part 1 describes the problem. Part 2 gives solution to the problem. Part 3 depicts results of simulations and experiments. Part 4 discusses future work and conclusion of this thesis project. The remainder of this thesis is organized as follows. Chapter 2 introduces modelling of mobile manipulations and propose

Table 1.1: Symbols and Notations

Symbol	Meaning
$\mathbb{N}$	Positive Integers
$\mathbb{R}^n$	Real n-coordinate Space, with $n \in \mathbb{N}$
$\mathbb{T}^3$	Three Dimensional Torus
$\mathbb{R}^n_{\geq 0}$	Real n-vectors with Nonnegative Elements, with $n \in \mathbb{N}$
$\mathbb{R}^n_{>0}$	Real n-vectors with Positive Elements, with $n \in \mathbb{N}$
$\mathbb{R}^{n\times n}_{\geq 0}$	Positive Semi-definite Matrices, with $n \in \mathbb{N}$
$\mathbb{R}^{n\times n}_{>0}$	Positive Definite Matrices, with $n \in \mathbb{N}$
I_n	$n \times n$ identity matrix, with $n \in \mathbb{N}$
$0_{n\times m}$	$n \times m$ matrix with zero entries, with $n, m \in \mathbb{N}$
$\parallel x \parallel$	Euclidean norm of a vector $x \in \mathbb{R}^n$
$\mid S \mid$	Cardinality of a Set S
S^N	Cartesian Product of a Set S
$\{A\}$	Coordinate Frame A
$\{I\}$	Inertial Frame
$SO(3)$	Three Dimensional Rotation Group

the problem of cooperative transportation with singularity avoidance and collision avoidance. Chapter 3 proposes DNMPC and CNMPC schemes with feedback linearization and model reduction. The proof of stability is also shown in chapter 3. Chapter 4 introduces the software tools that are used in this thesis. Chapter 5 provides the simulation results of DNMPC and CNMPC. Experimental results are shown in Chapter 6. Chapter 7 draws conclusion and proposes the future work.

Table 1.2: Constants

Symbol	Meaning	Value
g	Gravitational Acceleration	$9.8m/s^2$

Table 1.3: Abbreviations

Acronym	Meaning
MPC	Model Predictive Control
LMPC	Linear Model Predictive Control
NMPC	Nonlinear Model Predictive Control
CNMPC	Centralized Nonlinear Model Predictive Control
DNMPC	Decentralized Nonlinear Model Predictive Control
FHOCP	Finite Horizon Optimal Control Problem
UAV	Unmanned Aerial Vehicles
UGV	Unmanned Ground Vehicles
UUV	Unmanned Underwater Vehicles
MM-UAV	Mobile Manipulating Unmanned Aerial Vehicles
MM-UGV	Mobile Manipulating Unmanned Ground Vehicles
MM-UUV	Mobile Manipulating Unmanned Underwater Vehicles

Chapter 2

Modelling

As discussed in Chapter 1, this thesis discusses cooperative transportation of manipulators on UGV. This chapter introduces the mathematical model of cooperative transportation of manipulators on UGV. Section 2.1 gives some concepts [54] that are used in modelling. Section 2.2 introduces system model of cooperative transportation. Section 2.3 describes the problem formulation of singularity and collision avoidance.

2.1 Preliminaries

In this section, several concepts in [54] used in modelling are introduced. The concepts are described in detail by the authors of [54].

2.1.1 Rotation matrix and Euler angles

A rigid body in space is described by its *pose*, i.e its position and orientation, with respect to a reference frame. *Rotation matrix* and *Euler angles* [54] are needed to model pose of manipulators.

Suppose the reference frame $O - xyz$ is rotated by angle α around axis z. The rotated frame is denoted as $O - x'y'z'$. The rotation matrix of frame $O - x'y'z'$ with respect to frame $O - xyz$ is denoted as $R_z(\alpha)$. According to [54], $R_z(\alpha)$ can be expressed by (2.1) .

$$R_z(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

Similarly, the rotation matrix around axis y by angle β, $R_y(\beta)$, and the rotation matrix around axis x by angle γ, $R_x(\gamma)$, are expressed by (2.2) and (2.3), respectively.

$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \tag{2.2}$$

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & -\sin(\gamma) \\ 0 & \sin(\gamma) & \cos(\gamma) \end{bmatrix} \tag{2.3}$$

2.1.2 Joint space and operational space

A manipulator consists of several joints. Each joint corresponds to a *joint variable* and a degree of freedom. Usually, a task is conducted by the end-effector of a manipulator, whose pose is controlled by a group of joint variables. *Joint space* and *operational space* describe the space of joint variables and the space of end-effectors, respectively.

Definition 2.1.1 (joint space)**.** Joint space is a set of joint parameters which describes the overall configuration of a robotic manipulator. The joint space is also called configuration space.

Definition 2.1.2 (operational space)**.** Operational space is a set of all possible configurations of the end-effector of manipulators, which is also called task space.

Definition 2.1.3 (work space)**.** Workspace is a region described by the origin of the end-effector frame when all the joints execute all possible motions.

2.1.3 Kinematics and Jacobian matrix

The relationship between joint space and operational space is revealed by *forward kinematics*, *differential kinematics* and *inverse kinematics*. Forward kinematics equations, which is also called *direct kinematics* equations, can express the pose of end-effector, i.e position and orientation, as a function of joint variables, while *inverse kinematics* equations compute the joint variables using the pose of end-effector.

The mapping between the joint velocities and the corresponding end-effector linear and angular velocity is given by differential kinematics, which can be described by *geometric Jacobian* and *analytical Jacobian*.

Suppose the linear velocity of the end-effector with respect to the base frame is denoted by $\dot{p}_e$ and the end-effector angular velocity with respect to the base frame is denoted by ω_e. We define the end-effector velocity v_e in (2.4).

$$v_e = \begin{bmatrix} \dot{p}_e \\ \omega_e \end{bmatrix} \tag{2.4}$$

Then the differential kinematics equation can be expressed as (2.5), where q a $n \times 1$ vector of n joint variables and $J(q)$ is geometric Jacobian.

$$v_e = J(q)\dot{q} \tag{2.5}$$

Suppose the *rotational velocity* of the end-effector is denoted by $\dot{\phi}_e$. We define the end-effector velocity $\dot{x_e}$ in (2.6).

$$\dot{x_e} = \begin{bmatrix} \dot{p}_e \\ \dot{\phi}_e \end{bmatrix} \tag{2.6}$$

Similarly, analytical Jacobian J_A provides another way to calculate Jacobian matrix, which is shown in (2.7).

$$\dot{x_e} = J_A(q)\dot{q} \tag{2.7}$$

Normally, analytical Jacobian J_A is different from geometric Jacobian J. However, for *three link planar arm*, which will be used in our mobile manipulator, analytical Jacobian J_A equals to geometric Jacobian J. Both analytical Jacobian J_A and geometric Jacobian J are functions of joint variables.

2.1.4 Kinematic Redundancy

Kinematic redundancy refers to a situation where the number of degree of freedom is greater than the necessary number of joint variables to describe a given task. A manipulator is kinematically redundant when the the dimension of operational space is smaller than the dimension of joint space. In other word, kinematic redundancy means the number of columns is greater than the number of rows in a Jacobian matrix.

2.1.5 Singularity

Singularity is another important concept in manipulators. *Kinematic singularities* refer to the configurations that make Jacobian matrix rank-deficient. Kinematic singularities may cause the following problems. It may reduce the mobility of manipulators. For example, the end-effector of manipulator may be impossible to move to an arbitrary pose due to kinematic singularities. Another disadvantage of kinematic singularities is that it may cause infinite solutions to the inverse kinematics problem. Kinematic singularities may also cause large velocities in joint space which only corresponds to a small velocities in operational space. Thus, singularities should be avoided.

Kinematic singularities can be divided into *boundary singularities* and *internal singularities*. Boundary singularities will happen when the manipulator is out-stretched or retracted, which correspond to the boundary of the reachable workspace. Internal singularities will occur inside reachable workspace, which will cause serious problem when assigning a task in operational space.

2.2 System Model

In this section, system model of cooperative transportation used in this thesis is introduced. Kinematic and dynamic models of cooperative manipulators are described. Other techniques that might be used in modelling, such as inverse kinematic for redundant manipulator etc., are also introduced in this section.

Consider a multi-agent system with N MM-UGV agents rigidly grasping an object together in a bounded and convex workspace $\mathcal{W} \subseteq \mathbb{R}^3$. The rigidity means the agents can exert any force or torques along any direction to the object. Fig. 2.1 shows rigidly grasping an object. There are Z obstacles, which are described by the ellipsoids $\mathcal{O}_z$, where $z \in \mathcal{Z} = \{1, ..., Z\}$. A MM-UGV, which is fully actuated, consist of a UGV and a manipulator. The mobile base, i.e UGV, allows an agent to move around in the workspace $\mathcal{W}$. As is introduced in Chapter 1, the inertial reference frame is denoted by $\{I\}$, while the frames correspond to the object's center of mass and the i^{th} agent's end-effector are denoted by $\{O\}$ and $\{E_i\}$, respectively. Assume every agent only knows its own geometric parameters, position and velocity information of its own states as well as the object's geometric parameters. No

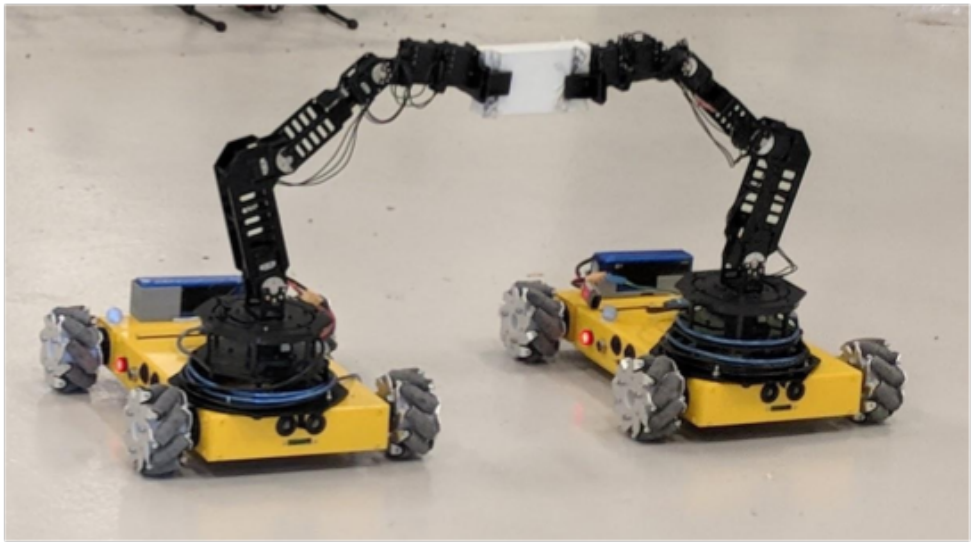


Figure 2.1: Tow robotic arms rigidly grasping an object

online communication and interaction force or torque measurements are allowed in CNMPC scheme, while communication between agents is allowed in DNMPC scheme.

The goal of the thesis is to avoid singularities and collision with obstacles when maintaining connections with the object. The free space, i.e the workspace without obstacle ellipsoids $\mathcal{O}_z$, is defined by $\mathcal{W}_{free} = \mathcal{W} \setminus \cup_{z \in \mathcal{Z}} \mathcal{O}_z$.

2.2.1 Kinematic Model

Without the loss of generalization, the mobile base is assumed to have six degree of freedom while the manipulator for agent i is assumed to have n_{α_i} degree of freedom, where $n_{\alpha_i} > 0$. The total degree of freedom of agent i is denoted by $n_i = n_{\alpha_i} + 6$.

The position of the agent's base, i.e the translation of the agent's base frame $\{B_i\}$ from the reference frame $\{I\}$ is denoted by $p_{B_i}(t) = [x_{B_i}, y_{B_i}, z_{B_i}]^\top : \mathbb{R}_{\geq 0} \to \mathbb{R}^3$, while the Euler-angle orientation of the agent's base, i.e the rotation of the agent's base frame $\{B_i\}$ from the reference frame $\{I\}$ is denoted by $\eta_{B_i}(t) = [\phi_{B_i}, \theta_{B_i}, \psi_{B_i}]^\top : \mathbb{R}_{\geq 0} \to \mathbb{T}^3 \subseteq \mathbb{R}^3$. $\alpha_i : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n_{\alpha_i}}$ is the degree of freedom of the robotic arm, i.e the joint variable of each degree of freedom, more specifically, α_i denotes joint angles for *rotational joints* and lengths for *prismatic joints*. The joint

space variables of agent $i \in \mathcal{N}$ can be denoted by $q_i : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n_i}$, where $q_i = \left[p_{B_i}^\top(t), \eta_{B_i}^\top(t), \alpha_i^\top(t)\right]^\top$. More specifically, the joint space variables of agent $i \in N$ can be expressed as (2.8).

$$q_i = \left[x_{B_i}, y_{B_i}, z_{B_i}, \phi_{B_i}, \theta_{B_i}, \psi_{B_i}, \alpha_{i_1}, \alpha_{i_2}, ..., \alpha_{i_{n_{\alpha_i}}}\right] \tag{2.8}$$

Then the overall joint space configuration vector can be denoted as $q = \left[q_1^\top, q_2^\top, ..., q_N^\top\right]^\top \in \mathbb{R}^n$, where $n = \sum_{i \in \mathcal{N}} n_i$.

The position and Euler-angle of agent i's end-effector are denoted by $p_{E_i} : \mathbb{R}^{n_i} \to \mathbb{R}^3$ and $\eta_{E_i} : \mathbb{R}^{n_i} \to \mathbb{T}^3 \subseteq \mathbb{R}^3$ respectively. The linear and angular velocity of agent's end-effector are denoted by $\dot{p}_{E_i} : \mathbb{R}^{n_i} \times \mathbb{R}^{n_i} \to \mathbb{R}^3$, $\omega_{E_i} : \mathbb{R}^{n_i} \times \mathbb{R}^{n_i} \to \mathbb{T}^3 \subseteq \mathbb{R}^3$ respectively, while $\dot{p}_{B_i} : \mathbb{R}^{n_i} \times \mathbb{R}^{n_i} \to \mathbb{R}^3$, $\omega_{B_i} : \mathbb{R}^{n_i} \times \mathbb{R}^{n_i} \to \mathbb{T}^3 \subseteq \mathbb{R}^3$ are the linear and angular velocity of agent's base respectively. The velocity of agent i's end-effector is denoted as $v_i(q_i, \dot{q}_i) : \mathbb{R}^{n_i} \times \mathbb{R}^{n_i} \to \mathbb{R}^6$, where $v_i(q_i, \dot{q}_i) = \left[\dot{p}_{E_i}^\top, \omega_{E_i}^\top\right]^\top$.

In CNMPC scheme, each agent only has access to its own state q_i, its linear and angular velocity with respect to the base frame $\dot{p}_{B_i}^{B_i}$, $\omega_{B_i}^{B_i}$ and its own degrees of freedom α_i by using local sensors. In DNMPC scheme, communication is allowed, which means each agent has access to the information of other agents.

Assume the rotation matrix of the agent i's base frame $\{B_i\}$ with respect to the reference frame $\{I\}$ is denoted by $R_{B_i}(\eta_{B_i}) : \mathbb{R} \to SO(3)$, where $R_{B_i}(\eta_{B_i})$ is a sub-matrix of the transformation matrix from frame $\{B_i\}$ to $\{I\}$, according to [54]. Then we can get the linear and angular velocity with respect to the reference frame $\{I\}$ via (2.9) and (2.10).

$$\dot{p}_{B_i} = R_{B_i}(\eta_{B_i})\dot{p}_{B_i}^{B_i} \tag{2.9}$$

$$\omega_{B_i} = R_{B_i}(\eta_{B_i})\omega_{B_i}^{B_i} \tag{2.10}$$

Furthermore, by using $J_{B_i}(\eta_{B_i}) : \mathbb{T}^3 \to \mathbb{R}^{3\times 3}$, the relationship between ω_{B_i} and $\dot{\eta}_{B_i}$ can be revealed by (2.11).

$$\omega_{B_i} = J_{B_i}(\eta_{B_i})\eta_{B_i} \tag{2.11}$$

where

$$J_{B_i}(\eta_{B_i}) = \begin{bmatrix} 1 & 0 & sin(\theta_{B_i}) \\ 0 & cos(\phi_{B_i}) & -cos(\theta_{B_i})sin(\phi_{B_i}) \\ 0 & sin(\phi_{B_i}) & cos(\theta_{B_i})cos(\phi_{B_i}) \end{bmatrix}.$$

Moreover, we define $k_{p_i} : \mathbb{R}^{n_{\alpha_i}} \to \mathbb{R}^3$ and $k_{\eta_i} : \mathbb{T}^3 \times \mathbb{R}^{n_{\alpha_i}} \to \mathbb{T}^3$ as the forward kinematics of the robotic arm with respect to the base frame

of the i^{th} agent $\{B_i\}$. The position of the i^{th} agent's end-effector can be computed by (2.12). The orientation of the i^{th} agent's end-effector can be computed by (2.13).

$$p_{E_i}(q_i) = p_{B_i} + R_{B_i}(\eta_{B_i})k_{p_i}(\alpha_i) \tag{2.12}$$

$$\eta_{E_i} = k_{\eta_i}(\eta_{B_i}, \alpha_i) \tag{2.13}$$

In the same way, ω_{E_i} can be computed by using frame transformation, which is shown in (2.14).

$$\omega_{E_i} = \omega_{B_i} + R_{B_i}\omega_{E_i}^{B_i} \tag{2.14}$$

The angular Jacobian of the robotic arm with respect to the agent's base frame $\{B_i\}$ is denoted by $J_{A_i} : \mathbb{R}^{n_{\alpha_i}} \to \mathbb{R}^{3\times n_{\alpha_i}}$. Then, $\omega_{E_i}^{B_i}$ can be obtained via differential kinematic equation (2.15) .

$$\omega_{E_i}^{B_i} = J_{A_i}\dot{\alpha}_i \tag{2.15}$$

By taking derivative with respect to (2.12), the overall differential kinematic for end-effector of i^{th} agent with respect to the base frame $\{B_i\}$ is shown in (2.16).

$$v_i(q_i, \dot{q}_i) = \begin{bmatrix} \dot{p}_{E_i}(q_i, \dot{q}_i) \\ \omega_{E_i}(q_i, \dot{q}_i) \end{bmatrix} = \begin{bmatrix} \dot{p}_{B_i} - S(R_{B_i}k_{p_i})\omega_{B_i} + R_{B_i}\frac{\partial k_{p_i}}{\partial \alpha_i} \\ \omega_{B_i} + R_{B_i}J_{A_i}\dot{\alpha}_i \end{bmatrix} \tag{2.16}$$

The i^{th} agent's Jacobian matrix is denoted by $J_i(q_i) : \mathbb{R}^{n_i} \to \mathbb{R}^{3\times n_i}$. According to the definition of Jacobian matrix, the overall differential kinematic equation for the i^{th} agent and agent i's variables can be related via $J_i(q_i)$ which is shown in (2.17).

$$v_i(q_i, \dot{q}_i) = J_i(q_i)\dot{q}_i, \tag{2.17}$$

where

$$J_i = \begin{bmatrix} I_3 & -S(R_{B_i}(\eta_{B_i})k_{p_i}(\alpha_i))J_{B_i}(\eta_{B_i}) & R_{B_i}(\eta_{B_i})\frac{\partial k_{p_i}(\alpha_i)}{\partial \alpha_i} \\ 0_{3\times 3} & J_{B_i}(\eta_{B_i}) & R_{B_i}(\eta_{B_i})J_{A_i}(q_i) \end{bmatrix}.$$

2.2.2 Inverse Kinematic for Redundant Manipulator

Inverse kinematic for manipulators is needed to compute dynamic equations.

Kinematic aims to compute the joint space variables $\dot{q}_i$ when the end-effector velocity v_i and Jacobian matrix J_i are given. In other word, the purpose of inverse kinematic is to find the solution $\dot{q}_i$ that satisfies (2.17).

For manipulator whose agent Jacobian J_i is a square matrix, i.e the number of columns is equal to the number of rows in J_i, the solution of the above equation can be obtained by left multiply the inverse of agent Jacobian $J_i^{-1}(q_i)$. More clearly, the solution is (2.18).

$$\dot{q}_i = J_i^{-1}(q_i)v_i \tag{2.18}$$

For redundant manipulators, since agent Jacobian J_i is not a square matrix, i.e J_i has more columns than rows, the inverse of J_i doesn't exist, which means we cannot get the solution in the same way. The solution of inverse kinematic for redundant manipulator can be obtained via solving an optimal problem. The optimal problem is to minimize the quadratic cost function of joint velocities (2.19), where $W \in \mathbb{R}^{n\times n}$ is a suitable symmetric positive weighting matrix, n is the number of columns of agent Jacobian J_i.

$$g(\dot{q}_i) = \frac{1}{2}\dot{q}_i^\top W \dot{q}_i \tag{2.19}$$

By employing Lagrange multipliers and the cost function (2.19), (2.20) can obtained, where λ is a vector of unknown multipliers that allows the cost function to take constraint $v_i = J_i(q_i)\dot{q}_i$ into consideration.

$$g(\dot{q}_i, \lambda) = \frac{1}{2}\dot{q}_i^\top W \dot{q}_i + \lambda^\top (v_i - J_i)\dot{q}_i \tag{2.20}$$

Then, the requested solution has to satisfy (2.21) and (2.22).

$$(\frac{\partial g}{\partial \dot{q}_i})^\top = 0 \tag{2.21}$$

$$(\frac{\partial g}{\partial \lambda})^\top = 0 \tag{2.22}$$

$W\dot{q}_i - \lambda^\top J = 0$ holds if and only if (2.21) holds. (2.23) can be derived from the first necessary condition (2.21).

$$\dot{q}_i = W^{-1} J_i^\top \lambda \tag{2.23}$$

From the second necessary condition (2.22), (2.24) can be obtained.

$$v_i = J_i W^{-1} J_i^\top \lambda \tag{2.24}$$

,where $J_i W^{-1} J^\top \in \mathbb{R}^{r \times r}$ is an invertible square matrix, r is the number of rows of agent Jacobian J_i . Since $J_i W^{-1} J^\top$ is invertible, λ can be solved, which is shown in (2.25).

$$\lambda = (J_i W^{-1} J_i^\top)^{-1} v_i \tag{2.25}$$

By substituting (2.25) into (2.21), the inverse kinematic for redundant manipulators can be solved. The solution is shown in (2.26).

$$\dot{q}_i = W^{-1} J_i^\top (J_i W^{-1} J_i^\top)^{-1} v_i \tag{2.26}$$

Usually, W is chosen as an identity matrix I, then the solution is simplified as (2.27).

$$\dot{q}_i = J_i^\top (J_i J_i^\top)^{-1} v_i \tag{2.27}$$

To simplify the notation, we define $J_i^+ = J_i^\top (J_i J_i^\top)^{-1}$. Then the velocities of joint variables can be expressed as (2.28) and time derivative of the velocities can be obtained (2.29).

$$\dot{q}_i = J_i^+ v_i \tag{2.28}$$

$$\ddot{q}_i = \dot{J}_i^+ v_i + J_i^+ \dot{v}_i \tag{2.29}$$

(2.28) and (2.29) can be used to derive dynamic of the system.

2.2.3 Dynamic Model

To get the dynamic equation in joint space, several concepts needs to be introduced first. The *joint space inertia matrix* is defined as $B_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i \times n_i}$, which is always a positive definite matrix. The *joint space Coriolis matrix* is $N_i : \mathbb{R}^{n_i} \times \mathbb{R}^{n_i} \to \mathbb{R}^{n_i \times n_i}$ and $g_{q_i} : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ is the *joint space gravity vector*. $\tau_i = \left[\lambda_{B_i}^\top, \tau_{\alpha_i}^\top\right]^\top$ is the generalized joint-space input vector, where $\lambda_{B_i} = \left[f_{B_i}^\top, \mu_{B_i}^\top\right]^\top \in \mathbb{R}^6 \in \mathbb{R}^{n_i}$ is the generalized force vector on the center of mass of the agent's base and $\tau_{\alpha_i} \in \mathbb{R}^{n_{\alpha_i}}$ is the

torque inputs of the robotic arms' joints. The generalized force vector which agent $i \in \mathcal{N}$ exterts on the object is denoted as $\lambda_i \in \mathbb{R}^6$.

By employing Lagrangian formulation and the mentioned concepts, the joint space dynamic equation for i^{th} agent can be expressed as (2.30).

$$B_i(q_i)\ddot{q}_i + N_i(q_i, \dot{q}_i)\dot{q}_i + g_{q_i}(q_i) = \tau_i - J_i^\top \lambda_i \tag{2.30}$$

The operational space dynamic equation can be derived easily by using (2.30) and (2.17). Thus, the operational space dynamic equation is shown in (2.31).

$$M_i(q_i)\dot{v}_i + C_i(q_i, \dot{q}_i)v_i + g_i(q_i) = u_i - \lambda_i, \tag{2.31}$$

where

$$\begin{aligned} M_i(q_i) &= (J_i(q_i)B_i^{-1}(q_i)J_i^\top(q_i))^{-1}, \\ C_i(q_i, \dot{q}_i)v_i &= M_i(q_i)J_i(q_i)B_i^{-1}(q_i)N_i(q_i, \dot{q}_i)\dot{q}_i - M_i(q_i)\dot{J}_i(q_i)\dot{q}_i, \\ g_i(q_i) &= M_i(q_i)J_i(q_i)B_i^{-1}(q_i)g_{q_i}(q_i), \end{aligned}$$

u_i is the task-space input, λ_i is the generalized force vector that i^{th} agent exerts on the object.

An alternative way is to express the operational space dynamic via joint variables, which is more suitable for experiments.

Recall that end-effector velocity $v_i(q_i, \dot{q}_i) = J_i(q_i)\dot{q}_i$. By differentiating (2.17), (2.32) can be obtained.

$$\dot{v}_i(q_i, \dot{q}_i, \ddot{q}_i) = \dot{J}_i(q_i)\dot{q}_i + J_i(q_i)\ddot{q}_i \tag{2.32}$$

By substituting the relationship between end-effector velocity and joint variables, (2.17) and (2.32), into operational space dynamic equation (2.31), (2.33) can be derived. Then, in (2.33), every term can be expressed using joint variables and the differentiation of joint variables of i^{th} agent.

$$\begin{aligned} M_i(q_i)J_i(q_i)\ddot{q}_i + M_i(q_i)J_i(q_i)B_i^{-1}(q_i)N_i(q_i, \dot{q}_i)\dot{q}_i + M_i(q_i)J_i(q_i)B_i^{-1}(q_i)g_{q_i} \\ = u_i - \lambda_i \end{aligned} \tag{2.33}$$

To get the coupled dynamic equation for cooperative transportation systems, the dynamic equation of the object is needed. The position of the center of the mass of the object is denoted as $p_o(t) =$

$[x_o(t), y_o(t), z_o(t)]^\top : \mathbb{R}_{\geq 0} \to \mathbb{R}^3$ and the orientation of the center of mass of the object is denoted as $v_o(t) = \left[\dot{p}_o^\top, \omega_o^\top\right]^\top$. $\eta_o(t) = [\phi_o(t), \theta_o(t), \psi_o(t)]^\top : \mathbb{R}_{\geq 0} \to \mathbb{T}^3 \subseteq \mathbb{R}^3$. The pose of the object is denoted as $x_o : \mathbb{R}_{\geq 0} \to \mathbb{M}$, where $x_o = \left[p_o^\top, \eta_o^\top\right]^\top$. $\dot{p}_o : \mathbb{R}^{n_i} \times \mathbb{R}^{n_i} \to \mathbb{R}^3$, $\omega_o : \mathbb{R}^{n_i} \times \mathbb{R}^{n_i} \to \mathbb{T}^3 \subseteq \mathbb{R}^3$ are the linear and angular velocity of the center of mass of the object. Then the velocity of the object's center of mass is denoted as $v_o(t) = [\dot{p}_o^\top, \omega_o^{top}]^\top : \mathbb{R}_{\geq 0} \to \mathbb{R}^3$. The inertia matrix of the object is denoted as $M_o : \mathbb{M} \to \mathbb{R}^{6\times 6}$. $C_o : \mathbb{M} \times \mathbb{R}^6 \to \mathbb{R}^{6\times 6}$ is the Coriolis matrix of the object and $g_o : \mathbb{M} \to \mathbb{R}^6$ is the gravity vector of the object. The object representation Jacobian is denoted as $J_{o_r} : \mathbb{M} \to \mathbb{R}^{6\times 6}$, which is a function of the pose of the object as shown in (2.34).

$$J_{o_r}(x_o) = \begin{bmatrix} I_3 & 0 \\ 0 & J_{o_{r,\theta}}(x_o) \end{bmatrix}, \tag{2.34}$$

where

$$J_{o_{r,\theta}}(x_o) = \begin{bmatrix} 1 & 0 & sin(\theta_o) \\ 0 & cos(\phi_o) & -cos(\theta_o)sin(\phi_o) \\ 0 & sin(\phi_o) & cos(\theta_o)cos(\phi_o) \end{bmatrix}.$$

Then, the dynamic equations of the object are described by (2.35) and (2.36).

$$\dot{x}_o(t) = J_{o_r}^{-1}(x_o)v_o(t) \tag{2.35}$$

$$\lambda_o = M_o(x_o)\dot{v}_o(t) + C_o(x_o, v_o)v_o(t) + g_o(x_o) \tag{2.36}$$

The pose of the object can be expressed as a function of q_i and $\dot{q}_i$ as (2.37) and (2.38), where $i \in \mathcal{N}$.

$$p_{E_i}(q_i(t)) = p_o(t) + p_{E_i/o}(q_i) = p_o(t) + R_{E_i}(t)p_{E_i/o}^{E_i}, \tag{2.37}$$

$$\eta_{E_i}(q_i(t)) = \eta_o(t) + \eta_{E_i/o}, \tag{2.38}$$

where $p_{E_i/o}^{E_i}$ is the constant distance and $\eta_{E_i/o}$ is the relative orientation offset between the i^{th} agent's end-effector and the object's center of mass, which are known.

From (2.37) and (2.38), the forward kinematic of the object can be derived as (2.39).

$$x_o(t) = k_i(q_i(t)) = \begin{bmatrix} p_{E_i}(q_i(t)) + p_{O/E_i}(q_i(t)) \\ \eta_{E_i}(q_i(t)) + \eta_{O/E_i} \end{bmatrix} = \begin{bmatrix} p_{B_i} + R_{B_i}k_{p_i} + R_{E_i}p_{O/E_i}^{E_i} \\ k_{\eta_i} + \eta_{O/E_i} \end{bmatrix} \quad (2.39)$$

Furthermore, the velocity and acceleration of the object can be described as (2.40) and (2.41), respectively.

$$v_i(q_i, \dot{q}_i) = J_{o_i}(q_i)v_o(t), \quad (2.40)$$

$$\dot{v}_i(t) = J_{o_i}(q_i)\dot{v}_o(t) + \dot{J}_{o_i}(q_i)v_o(t), \quad (2.41)$$

where $J_{o_i}(q_i) : \mathbb{R}^n \to \mathbb{R}^{6\times 6}$ is a representing Jacobian matrix which is a mapping from the object to the i^{th} agent's end-effector, with $J_{o_i}(q_i) = \begin{bmatrix} I_3 & S(p_{O/E_i}(q_i)) \\ 0_{3\times 3} & I_3 \end{bmatrix}$.

Note that the angular velocity of the end-effector equals to the angular velocity of the object due to the grasp rigidity, i.e $\omega_{E_i} = \omega_o$, $\forall i \in \mathcal{N}$. The grasp rigidity also implies that $J_{o_i}(q_i)$ is full rank.

The force exerting on the center of mass of the object is denoted as $\lambda_o \in \mathbb{R}^6$ and the generalized forces exerted by the agent at the contact points is denoted as λ_i, $\forall i \in \mathcal{N}$. The relationship between λ_o and λ_i is shown in (2.42), if the object is gripped rigidly.

$$\lambda_o = G^\top(q)\lambda, \quad (2.42)$$

where $\lambda = \left[\lambda_1^\top, \cdots, \lambda_N^\top\right]^\top \in \mathbb{R}^6$ is a vector of the generalized forces exerted by the agent at the contact points and the grasp matrix $G(q) : \mathbb{R}^n \to \mathbb{R}^{6N\times 6}$ is defined by $G(q) = \left[J_{o_1}^\top, \cdots, J_{o_N}^\top\right]^\top$. And λ in (2.42) can be obtained by (2.43).

$$\lambda = u - \bar{M}(q)\dot{v} - \bar{C}(q)v - \bar{g}(q), \quad (2.43)$$

where $\bar{M} = diag\{[M_i]\}, \forall i \in \mathcal{N}$, $\bar{C} = diag\{[C_i]\}, \forall i \in \mathcal{N}$, $v = \left[v_1^\top, \cdots, v_N^\top\right]^\top$, $\bar{g} = \left[g_1^\top, \cdots, g_N^\top\right]^\top$ and $u = \left[u_1^\top, \cdots, u_N^\top\right]^\top$.

Since there are N agents in the system, the agents will share the load of the object. The dynamic equation with sharing load coefficient can be expressed in (2.44).

$$\sum_{i \in N} c_i \left\{ M_o \dot{v}_o + C_o v_o + g_o \right\} = G^\top \lambda, \tag{2.44}$$

where c_i is a sharing coefficient, with $c_i \in [0, 1]$ and $\sum_{i \in N} c_i = 1$.

By substituting (2.40) and (2.41) into (2.44), (2.45) is obtained, where the dynamic equation with sharing load coefficient is expressed by the end-effector velocities $v_i, \forall i \in \mathcal{N}$.

$$\sum_{i \in N} c_i \left\{ M_o J_{i_o} \dot{v}_i + (M_o \dot{J}_{i_o} + C_o J_{i_o}) v_i + g_o \right\} = G^\top \lambda \tag{2.45}$$

Furthermore, we can express the coupled dynamics in joint space, i.e the dynamic equations are described by joint variables, which will be used in DNMPC method later. By substituting (2.33) into (2.45), the coupled dynamics expressed in joint space can be obtained as (2.46).

$$\sum_{i \in N} \widetilde{M}_i \ddot{q}_i + \widetilde{C}_i \dot{q}_i + \widetilde{g}_i = \sum_{i \in \mathcal{N}} J_{o_i}^\top u_i, \tag{2.46}$$

where

$$\begin{aligned}
\widetilde{M}_i(x_o, q_i) &= c_i M_o J_{i_o} J_i + J_{o_i}^\top M_i J_i, \\
\widetilde{C}_i(\dot{x}_o, x_o, q_i, \dot{q}_i) &= c_i (C_o J_{i_o} J_i + M_o \dot{J}_{i_o} J_i + M_o J_{i_o} \dot{J}_i) + J_{o_i}^\top M_i J_i B_i^{-1} N_i, \\
\widetilde{g}_i(x_o, q_i) &= c_i g_o + J_{o_i}^\top g_i.
\end{aligned}$$

The dynamic system which is suitable for CNMPC method is introduced hereafter. By substituting (2.40) and (2.41) into (2.31), (2.47) can be obtained.

$$\lambda = u - \bar{M}(q) G(q) \dot{v}_o - (\bar{M}(q) \dot{G}(q, \dot{q}) + \bar{C}(q, \dot{q}) G(q)) v_o - \bar{g}(q)) \tag{2.47}$$

Then by substituting (2.47), (2.35) and (2.36) into (2.42), the coupled dynamics (2.48), which is suitable for CNMPC method, can be derived.

$$\widetilde{M}(q) \dot{v}_o + \widetilde{C}(q, \dot{q}) v_o + \widetilde{g}(q) = G^\top(q) u, \tag{2.48}$$

where

$$\begin{aligned}
\widetilde{M}(q) &= M_o(q) + G^\top(q) \bar{M}(q) G(q), \\
\widetilde{C}(q, \dot{q}) &= C_o(q) + G^\top(q) \bar{M}(q) \dot{G}(q, \dot{q}) + G^\top(q) \bar{C}(q) G(q), \\
\widetilde{g}(q) &= g_o(q) + G^\top(q) \bar{g}(q).
\end{aligned}$$

2.3 Problem Formulation

In this thesis, we focus on cooperative manipulation using DNMPC method and CNMPC method. The transportation goal is to move the object from starting point to destination while avoiding collision with obstacles and singularity of the manipulators.

The following reasonable assumption is needed to solve the problem.

Assumption 2.3.1. Assume that N robotic agents grasp an object rigidly and the distance between any two obstacles are large enough such that the system can be navigated by controllers without collision with obstacles.

2.3.1 Collision Avoidance

Assume the volume of the i^{th} agent can be bounded by an ellipsoid $\mathcal{A}_i(q_i) \triangleq \mathcal{O}_i, \forall i \in \mathcal{N}$, whose center is c_i and semi axes are $\beta_{i,1}$, $\beta_{i,2}$ and $\beta_{i,3}$, $\forall i \in \mathcal{N}$. $\mathcal{A}_i(q_i)$ represents the workspace of agent i, which includes the manipulator and the mobile base. Obviously, $\mathcal{A}_i(q_i)$ is only determined by the states of agent i, q_i. The object can be bounded by an ellipsoid $\mathcal{C}_o(x_o) \triangleq \mathcal{O}_o$, which can be expressed by the position of the object, x_o.

Collision avoidance can be described by (2.49), (2.50) and (2.51).

$$\mathcal{A}_i(q_i) \cap \mathcal{O}_z = \emptyset, \forall i \in \mathcal{N}, z \in \mathcal{Z} \tag{2.49}$$

$$\mathcal{C}_o(x_o) \cap \mathcal{O}_z = \emptyset, \forall z \in \mathcal{Z} \tag{2.50}$$

$$\mathcal{A}_i(q_i) \cap \mathcal{A}_j(q_j) = \emptyset, \forall i, j \in \mathcal{N}, i \neq j \tag{2.51}$$

Collision avoidance between agents and obstacles can be described by (2.49). (2.50) describes collision avoidance between the object and obstacles. (2.51) describes the condition of collision avoidance between agents.

For convenience, we can define the sets, which are related to the collision avoidance properties.

$$S_{i,A}(q) = \{q_i \in \mathbb{R}^{n_i} : \mathcal{A}_i(q_i) \cap \mathcal{O}_z \neq \emptyset, \forall i \in \mathcal{N}, z \in \mathcal{Z}\}$$

$$S_o(x_o) = \{x_o \in \mathbb{M} : \mathcal{C}_o(x_o) \cap \mathcal{O}_z \neq \emptyset, \forall z \in \mathcal{Z}\}$$

$$S_{i,A}(q) = \{q_i \in \mathbb{R}^{n_i} : \mathcal{A}_i(q_i) \cap \mathcal{A}_j(q_j) \neq \emptyset, \forall i, j \in \mathcal{N}, i \neq j\}$$

2.3.2 Singularity Avoidance

In the aforementioned introduction, singularity will happen when a Jacobian matrix is rank deficient, which should be avoided by control algorithms. There are three Jacobian matrices in cooperative manipulation problem, which are the Jacobian matrix of the mobile base J_{B_i}, Jacobian matrix of agent i J_i and the object representation Jacobian matrix J_{o_r}.

J_{B_i} becomes singular at representation singularities when $\theta_{B_i} = \pm\frac{\pi}{2}$. Thus, the singularity of J_{B_i} is avoided when the following equation is satisfied.

$$-\frac{\pi}{2} < -\bar{\theta} \leq \theta_{B_i} \leq \bar{\theta} < \frac{\pi}{2}, \forall i \in \mathcal{N}, \tag{2.52}$$

where $0 < \bar{\theta} < \frac{\pi}{2}$.

J_{o_r} becomes singular at representation singularities when $\theta_o = \pm\frac{\pi}{2}$. Thus, the singularity of J_{o_r} is avoided when the following equation is satisfied.

$$-\frac{\pi}{2} < -\bar{\theta} \leq \theta_o \leq \bar{\theta} < \frac{\pi}{2}, \tag{2.53}$$

where $0 < \bar{\theta} < \frac{\pi}{2}$.

J_i becomes singular at kinematic singularities when the states of agent i q_i are in the set Q_i, which is defined by (2.54).

$$Q_i = \{q_i \in \mathbb{R}^{n_i} : det(J_i^\top J_i) = 0\}, \forall i \in \mathcal{N} \tag{2.54}$$

To avoid kinematic singularity in J_i, q_i should be within set $\tilde{Q}_i$, which is defined by (2.55).

$$\widetilde{Q_i} = \{q_i \in \mathbb{R}^{n_i} : det(J_i^\top J_i) \geq \varepsilon > 0\}, \tag{2.55}$$

where ε is an arbitrary small positive constant number.

2.4 Summary

In this chapter, the kinematic model and dynamic model of cooperative manipulations are introduced, which will be used to develop DNMPC controller and CNMPC controller in the following chapters. The goal of cooperative manipulation is to transport an object to destination with collision and singularity avoidance. The collision avoidance

and singularity avoidance are formulated in a mathematical manner, which will be used in the controller design.

Part II

Solution

Chapter 3

Model Predictive Control

Model Predictive Control is an effective control strategy which has been widely used in both industrial use and academic research. MPC, which is also known as receding horizon control, is an online optimal control strategy. MPC controller optimizes a finite time horizon using a predicted model, implements only the current optimal solution and repeats this process. One of the advantages of MPC is its real-time optimization with predictive ability. NMPC is suitable for nonlinear systems, state constraints and input constraints [42, 21, 22, 7].

However, MPC with nonlinear system and nonlinear constraints has a huge computation expense compared with linear MPC [25, 55]. The drawback of NMPC restricts the use in nonlinear systems. One way to reduce the computational expense caused by nonlinear systems and nonlinear constraints is feedback linearization [55]. Nonlinear terms in differential equation can be cancelled out by feedback linearization, which will reduce computational burden greatly. Another solution to reduce the computational burden is model reduction. Obviously, the second order system, i.e dynamic system, is more complex than the first order system, i.e kinematic system. By reducing the second order system to the first order system, the computational burden will be reduced greatly, which is more suitable for experiments. In this thesis, both feedback linearization and model reduction are employed.

In this chapter, two NMPC strategies, CNMPC and DNMPC, are introduced. CNMPC scheme of a nonlinear dynamic system of cooperative transportation has been introduced in [45], while DNMPC scheme of a nonlinear dynamic system of cooperative transportation has been introduced in [60]. In this thesis, we introduce feedback lin-

earzation and model reduction in both CNMPC scheme and DNMPC scheme to reduce computational burden.

Before we introduce CNMPC and DNMPC schemes, $\mathcal{K}$ function and a lemma needs to be introduced.

Definition 3.0.1 ($\mathcal{K}$ function [25])**.** A continuous function $f : [0, \alpha] \to \mathbb{R}_{\geq 0}$, $\alpha \in \mathbb{R}_{>0}$ is said to belong to class $\mathcal{K}$, if it is strictly increasing and $f(0) = 0$.

Lemma 3.0.1 ([34])**.** Let γ be a continuous, positive definite function and x be an absolutely continuous function on $\mathbb{R}$. If the following conditions hold:

$$\|x(\cdot)\| < \infty, \|\dot{x}(\cdot)\| < \infty,$$

$$\lim_{t\to\infty} \int_0^t \gamma(x(s))ds < \infty.$$

Then $lim_{t\to\infty}\|x(t)\| = 0$.

The rest of this chapter are organized as follow. In section 3.1, CNMPC controllers for dynamic system with feedback linearization and kinematic system are introduced. In section 3.2, DNMPC controllers for dynamic system with feedback linearization and kinematic system are introduced. In section 3.3, stability of the proposed controllers are proved.

3.1 Centralized NMPC

The transportation of the object can be implemented with singularity avoidance and collision avoidance via a centralized NMPC controller, which takes all the agents into consideration. In this section, the CNMPC scheme for dynamic system and kinematic system will be introduced respectively.

3.1.1 Dynamic System

According to Chapter 2, the coupled agents-object dynamic system with feedback linearization is governed by (3.1).

$$\dot{x} = f(x, u) = \begin{bmatrix} f_1(x, u) \\ f_2(x, u) \\ f_3(x, u) \end{bmatrix}, x(0) = x_0 \tag{3.1}$$

, where $x = \left[x_o^\top, v_o^\top, q^\top\right]^\top \in \mathbb{R}^{n+12}$, $u \in \mathbb{R}^6$ and

$$f_1(x,u) = J_{o_r}^{-1}(x_o)v_o, \tag{3.2}$$

$$f_2(x,u) = \widetilde{M}^{-1}(q)\left[G^\top(q)\widetilde{u} - \widetilde{C}(q,\dot{q})v_o - \widetilde{g}(q)\right], \tag{3.3}$$

$$f_3(x,u) = \hat{J}(q)J_o(q)\widetilde{I}v_o, \tag{3.4}$$

where

$$\widetilde{u} = G(q)(G^\top(q)G(q))^{-1}(\widetilde{C}(q,\dot{q})v_o + \widetilde{g}(q) + \widetilde{M}(q)u), \tag{3.5}$$

$$\hat{J}(q) = diag\{\left[J_i^\top(J_iJ_i^\top)^{-1}\right]_{i\in\mathcal{N}}\} \in \mathbb{R}^{n\times 6N}, \tag{3.6}$$

$$J_o(q) = diag\{[J_{o_i}]_{i\in\mathcal{N}}\} \in \mathbb{R}^{6N\times 6N}, \tag{3.7}$$

$$\widetilde{I} = [I_6, ..., I_6]^\top \in \mathbb{R}^{6N\times 6}. \tag{3.8}$$

Feedback linearization is implemented by (3.5). Then, $f_2(x,u)$ is linearized, which will reduce computational burden of NMPC solver greatly. $f(x,u)$ is continuously differentiable in its domain, thus it is locally Lipschitz continuous.

Furthermore, the error dynamics can be derived from (3.1), which will make the CNMPC scheme easier be described.

First of all, the destinations of the states are needed to be defined. $q_{des} = [q_{1,des}, ..., q_{N,des}]^\top$ is chosen according to (2.37) and (2.38), such that $x_o(t) = x_{o,des}$ and $\dot{x}_{o,des} = \dot{q}_{des} = 0$.

Then, the errors can be defined as (3.9).

$$e(t) = x(t) - x_{des} = \begin{bmatrix} x_o(t) \\ v_o(t) \\ q(t) \end{bmatrix} - \begin{bmatrix} x_{o,des} \\ \dot{x}_{o,des} \\ q_{des} \end{bmatrix} = \begin{bmatrix} x_o(t) - x_{o,des} \\ v_o(t) \\ q(t) - q_{des} \end{bmatrix} \in \mathbb{R}^{n+12} \tag{3.9}$$

The error dynamics can be transformed from (3.1), which is described by (3.10).

$$\dot{e}(t) = f_e(e(t), u(t)) = f(e(t) + x_{des}, u(t)), e(0) = x(0) - x_{des}. \tag{3.10}$$

Then the input constraints needs to be defined. There is a constraint for the input and velocity magnitude, which is described in (3.11).

$$|\dot{q}_{i_k}| \leq \bar{\dot{q}}_i, \forall k \in \{1, .., n_i\}, i \in \mathcal{N}, \tag{3.11}$$

where $\bar{\dot{q}}_i$ is is a positive constant. The joint space inputs of manipulators are defined as $\tau_i = J_i^\top(q_i)u_i \in \mathbb{R}^{n_i}, \forall i \in \mathbb{N}$, if we ignore over-actuated inputs. Assume the joint space input is bounded by (3.12).

$$|\tau_{i_k}| \leq \bar{\tau}_i, \forall k \in \{1, .., n_i\}, i \in \mathcal{N}, \tag{3.12}$$

where $\bar{\tau}_i$ is is a positive constant.

(3.12) can be translated into (3.14) by using the property $\sigma(J_i^\top)\|u_i\| \leq \|J_i^\top u_i\|$, where $\sigma(J_i^\top)$ is the minimum singular value of $J_i^\top$. Note that $\sigma(J_i^\top)$ is strictly positive if $q_i \in \widetilde{Q}_i$ is satisfied.

$$\|\tau_i\| \leq \bar{\tau}_i \iff \sigma_{min}(J_i^\top)\|u_i\| \leq \bar{\tau}_i, \forall i \in \mathcal{N} \tag{3.13}$$

Then, the input constraints in joint space (3.12) is equivalent to (3.14).

$$\|u_i\| \leq \frac{\bar{\tau}_i}{\sigma_{min}(J_i^\top)}, \forall i \in \mathcal{N} \tag{3.14}$$

Then the input constraints of the error dynamic system (3.10) can be described by set $U \subseteq \mathbb{R}^6$, which is defined as (3.15).

$$U = \left\{ u \subseteq \mathbb{R}^6 : \|u_i\| \leq \frac{\bar{\tau}_i}{\sigma_{min}(J_i^\top)}, \forall i \in \mathcal{N} \right\} \tag{3.15}$$

The state constraints of the dynamic system (3.1) can be described by set $X \subseteq \mathbb{R}^{n+12}$, which is defined as (3.16).

$$\begin{aligned} X = \{x \subseteq \mathbb{R}^{n+12} : \theta_o(t) \in \left[-\bar{\theta}, \bar{\theta}\right], \theta_{B_i}(t) \in \left[-\bar{\theta}, \bar{\theta}\right], |\dot{q}_{i_k}| \leq \bar{\dot{q}}_i, \\ q_i \in \widetilde{Q}_i \setminus (S_{i,o}(q_i)) \cup S_{i,A}(q_i), x_o \in \mathbb{R}^3 \setminus S_o(x_o), \forall t \in \mathbb{R}_{\geq 0}\} \end{aligned} \tag{3.16}$$

The state constraints of the error dynamic system (3.10) can be described by set $E \subseteq \mathbb{R}^{n+12}$, which is defined as (3.17).

$$E = \{e \in \mathbb{R}^{n+12} : e \in X \oplus (-x_{des})\} \tag{3.17}$$

CNMPC scheme is able to find a control input $u(t) \in U$ such that $lim_{t\to\infty}\|e(t)\| = 0$ and $e(t) \in E, \forall t \in \mathbb{R}_{\geq 0}$.

The CNMPC scheme has a constant sampling time $0 < h < T_p$, where T_p is the prediction horizon of CNMPC. Then the sequence of sampling time $\{t_i\}_{t_i \geq 0}$ satisfies (3.18).

$$t_{i+1} = t_i + h, \forall t \geq 0 \tag{3.18}$$

At sampling time instants t_i, a finite-horizon open-loop optimal problem, which is define by (3.19), (3.20), (3.21), and (3.22), is solved according to the current state error $e(t_i)$. The solution is an optimal control signal $\hat{u}(t)$ in time interval $t \in [t_i, t_i + T_p]$, which will be implemented in between the sampling instants. For convenience, hat $\hat{}$

denotes the predicted variables. $\hat{e}(\cdot)$ is the predicted solution of (3.20) driven by $\hat{u} : [t_i, t_i + T_p] \to U$ under initial condition $e(t_i)$.

$$\min_{\hat{u}(\cdot)} J(e(t_i)) = \min_{\hat{u}(\cdot)} \{V(\hat{e}(t_i + T_p)) + \int_{t_i}^{t_i+T_p} [F(\hat{e}(s), \hat{u}(s))]\, ds\} \quad (3.19)$$

subject to:

$$\dot{\hat{e}}(s) = f_e(\hat{e}(s), \hat{u}(s)), \hat{e}(t_i) = e(t_i), \quad (3.20)$$

$$\hat{e}(s) \in E, \hat{u} \in U, s \in [t_i, t_i + T_p], \quad (3.21)$$

$$\hat{e}(t_i + T_p) \in \mathcal{E}_f, \quad (3.22)$$

where $F : E \times U \to \mathbb{R}_{\geq 0}$ is the running cost term, which is defined as (3.23), $V : E \to \mathbb{R}_{>0}$ is the terminal penalty cost, which is given by (3.24), and $\mathcal{E}_f$ is the terminal set. V and $\mathcal{E}_f$ are able to enforce the stability of the system. The terms Q, P, R are chosen such that $Q = diag\{\widetilde{q}_1, ..., \widetilde{q}_{n+12}\} \in \mathbb{R}_{\geq 0}^{(n+12)\times(n+12)}$, $P = diag\{\widetilde{p}_1, ..., \widetilde{p}_{n+12}\} \in \mathbb{R}_{>0}^{(n+12)\times(n+12)}$, and $R = diag\{\widetilde{r}_1, ..., \widetilde{r}_6\} \in \mathbb{R}_{>0}^{6\times 6}$, where $\widetilde{q}_i \in \mathbb{R}_{\geq 0}$, $\widetilde{p}_i \in \mathbb{R}_{>0}$, $\forall i \in \{1, ..., n + 12\}$ and $\widetilde{r}_j \in \mathbb{R}_{>0}$, $\forall j \in \{1, ..., 6\}$.

$$F(e(t), u(t)) = e(t)^\top Q e(t) + u(t)^\top R u(t) \quad (3.23)$$

$$V(e(t)) = e(t)^\top P e(t) \quad (3.24)$$

Note that the running cost function F holds that $F(0, 0) = 0$ and (3.25).

$$m\|e\|^2 \leq m \left\| \begin{matrix} e \\ u \end{matrix} \right\|^2 \leq F(e, u) \leq M \left\| \begin{matrix} e \\ u \end{matrix} \right\|^2 \leq M\|e\|^2, \quad (3.25)$$

where $m = min\{\widetilde{q}_1, ..., \widetilde{q}_{n+12}, \widetilde{r}_1, ..., \widetilde{r}_6\}$ and $M = max\{\widetilde{q}_1, ..., \widetilde{q}_{n+12}, \widetilde{r}_1, ..., \widetilde{r}_6\}$. According to Definition 3.0.1, $m\|e\|^2$ and $M\|e\|^2$ are $\mathcal{K}$ functions.

An optimal input for $t \in [t_i, t_i + T_p]$, which is denoted by $\hat{u}^*(t; e(t_i))$, is calculated by solving the optimal control problem, which is define by (3.19), (3.20), (3.21), and (3.22), at time t_i. $\hat{u}^*(t; e(t_i))$ is applied to the system until the next sampling time t_{i+1}, which can be described in (3.26) and the corresponding optimal value function is given by (3.27).

$$u(t; e(t_i)) = \hat{u}^*(t; e(t_i)), t \in [t_i, t_i + 1) \quad (3.26)$$

$$J^*(e(t_i)) = J^*(e(t_i)), \hat{u}^*(\cdot; e(t_i)) \quad (3.27)$$

At each sampling time instants, the control input $u(t; e(t_i))$ is recalculated using new state information. Thus, the control input $u(t; e(t_i))$ is a feedback.

3.1.2 Kinematic System

According to Chapter 2, the coupled agents-object kinematic system is governed by (3.28).

$$\dot{x} = f(x,u) = \begin{bmatrix} f_1(x,u) \\ f_2(x,u) \end{bmatrix}, x(0) = x_0, \tag{3.28}$$

where $x = \left[x_o^\top, q^\top\right]^\top \in \mathbb{R}^{n+6}$, $u \in \mathbb{R}^6$ and

$$f_1(x,u) = J_{o_r}^{-1}(x_o)u, \tag{3.29}$$

$$f_2(x,u) = \hat{J}(q)J_o(q)\widetilde{I}v_o, \tag{3.30}$$

where

$$\hat{J}(q) = diag\{\left[J_i^\top(J_iJ_i^\top)^{-1}\right]_{i\in\mathcal{N}}\} \in \mathbb{R}^{n\times 6N}, \tag{3.31}$$

$$J_o(q) = diag\{[J_{o_i}]_{i\in\mathcal{N}}\} \in \mathbb{R}^{6N\times 6N}, \tag{3.32}$$

$$\widetilde{I} = [I_6, ..., I_6]^\top \in \mathbb{R}^{6N\times 6}. \tag{3.33}$$

The input of the kinematic system is the velocity of object, i.e v_o. $f(x,u)$ is continuously differentiable in its domain, thus it is locally Lipschitz continuous.

Furthermore, the error kinematics can be derived from (3.28), which will make the CNMPC scheme easier be described.

First of all, the destinations of the states are needed to be defined. $q_{des} = [q_{1,des}, ..., q_{N,des}]^\top$ is chosen according to (2.37) and (2.38), such that $x_o(t) = x_{o,des}$ and $\dot{x}_{o,des} = \dot{q}_{des} = 0$.

Then, the errors can be defined as (3.34).

$$e(t) = x(t) - x_{des} = \begin{bmatrix} x_o(t) \\ q(t) \end{bmatrix} - \begin{bmatrix} x_{o,des} \\ q_{des} \end{bmatrix} = \begin{bmatrix} x_o(t) - x_{o,des} \\ q(t) - q_{des} \end{bmatrix} \in \mathbb{R}^{n+6} \tag{3.34}$$

The error dynamics can be transformed from (3.28), which is described by (3.35).

$$\dot{e}(t) = f_e(e(t), u(t)) = f(e(t) + x_{des}, u(t)), e(0) = x(0) - x_{des}. \tag{3.35}$$

Then the input constraints needs to be defined. There is a constraint for the velocity magnitude, which is described in (3.36).

$$|q_{i_k}| \leq \bar{q}_i, \forall k \in \{1, .., n_i\}, i \in \mathcal{N}, \tag{3.36}$$

where $\bar{q}_i$ is is a positive constant.

Then the input constraints of the error kinematic system (3.10) can be described by set $U \subseteq \mathbb{R}^6$, which is defined as (3.37).

$$U = \{u \subseteq \mathbb{R}^6 : |q_{i_k}| \leq \bar{q}_i, \forall k \in \{1, .., n_i\}, i \in \mathcal{N}\} \tag{3.37}$$

The state constraints of the dynamic system (3.28) can be described by set $X \subseteq \mathbb{R}^{n+6}$, which is defined as (3.38).

$$\begin{aligned} X = \{x \subseteq \mathbb{R}^{n+6} : \theta_o(t) \in \left[-\bar{\theta}, \bar{\theta}\right], \theta_{B_i}(t) \in \left[-\bar{\theta}, \bar{\theta}\right], \\ q_i \in \widetilde{Q}_i \setminus (S_{i,o}(q_i)) \cup S_{i,A}(q_i), x_o \in \mathbb{R}^3 \setminus S_o(x_o), \forall t \in \mathbb{R}_{\geq 0}\} \end{aligned} \tag{3.38}$$

The state constraints of the error kinematic system (3.35) can be described by set $E \subseteq \mathbb{R}^{n+6}$, which is defined as (3.39).

$$E = \{e \in \mathbb{R}^{n+6} : e \in X \oplus (-x_{des})\} \tag{3.39}$$

CNMPC scheme is able to find a control input $u(t) \in U$ such that $lim_{t\to\infty}\|e(t)\| = 0$ and $e(t) \in E, \forall t \in \mathbb{R}_{\geq 0}$.

The CNMPC scheme has a constant sampling time $0 < h < T_p$, where T_p is the prediction horizon of CNMPC. Then the sequence of sampling time $\{t_i\}_{t_i \geq 0}$ satisfies (3.40).

$$t_{i+1} = t_i + h, \forall t \geq 0 \tag{3.40}$$

At sampling time instants t_i, a finite-horizon open-loop optimal problem, which is define by (3.41), (3.42), (3.43), and (3.44), is solved according to the current state error $e(t_i)$. The solution is an optimal control signal $\hat{u}(t)$ in time interval $t \in [t_i, t_i + T_p]$, which will be implemented in between the sampling instants. For convenience, hat $\hat{\cdot}$ denotes the predicted variables. $\hat{e}(\cdot)$ is the predicted solution of (3.42) driven by $\hat{u} : [t_i, t_i + T_p] \to U$ under initial condition $e(t_i)$.

$$\min_{\hat{u}(\cdot)} J(e(t_i)) \;=\; \min_{\hat{u}(\cdot)}\{V(\hat{e}(t_i + T_p)) + \int_{t_i}^{t_i+T_p} [F(\hat{e}(s), \hat{u}(s))]\, ds\} \tag{3.41}$$

subject to:

$$\dot{\hat{e}}(s) = f_e(\hat{e}(s), \hat{u}(s)), \hat{e}(t_i) = e(t_i), \tag{3.42}$$

$$\hat{e}(s) \in E, \hat{u} \in U, s \in [t_i, t_i + T_p], \tag{3.43}$$

$$\hat{e}(t_i + T_p) \in \mathcal{E}_f, \tag{3.44}$$

where $F : E \times U \to \mathbb{R}_{\geq 0}$ is the running cost term, which is defined as (3.45), $V : E \to \mathbb{R}_{>0}$ is the terminal penalty cost, which is given by (3.46), and $\mathcal{E}_f$ is the terminal set. V and $\mathcal{E}_f$ are able to enforce the stability of the system. The terms Q, P, R are chosen such that $Q = diag\{\widetilde{q}_1, ..., \widetilde{q}_{n+6}\} \in \mathbb{R}_{\geq 0}^{(n+6)\times(n+6)}$, $P = diag\{\widetilde{p}_1, ..., \widetilde{p}_{n+6}\} \in \mathbb{R}_{>0}^{(n+6)\times(n+6)}$, and $R = diag\{\widetilde{r}_1, ..., \widetilde{r}_6\} \in \mathbb{R}_{>0}^{6\times 6}$, where $\widetilde{q}_i \in \mathbb{R}_{\geq 0}$, $\widetilde{p}_i \in \mathbb{R}_{>0}$, $\forall i \in \{1, ..., n+6\}$ and $\widetilde{r}_j \in \mathbb{R}_{>0}$, $\forall j \in \{1, ..., 6\}$.

$$F(e(t), u(t)) = e(t)^\top Q e(t) + u(t)^\top R u(t) \tag{3.45}$$

$$V(e(t)) = e(t)^\top P e(t) \tag{3.46}$$

Note that the running cost function F holds that $F(0, 0) = 0$ and (3.47).

$$m\|e\|^2 \leq m \left\| \begin{matrix} e \\ u \end{matrix} \right\|^2 \leq F(e, u) \leq M \left\| \begin{matrix} e \\ u \end{matrix} \right\|^2 \leq M\|e\|^2, \tag{3.47}$$

where $m = min\{\widetilde{q}_1, ..., \widetilde{q}_{n+12}, \widetilde{r}_1, ..., \widetilde{r}_6\}$ and $M = max\{\widetilde{q}_1, ..., \widetilde{q}_{n+12}, \widetilde{r}_1, ..., \widetilde{r}_6\}$. According to Definition 3.0.1, $m\|e\|^2$ and $M\|e\|^2$ are $\mathcal{K}$ functions.

An optimal input for $t \in [t_i, t_i + T_p]$, which is denoted by $\hat{u}^*(t; e(t_i))$, is calculated by solving the optimal control problem, which is define by (3.41), (3.42), (3.43), and (3.44), at time t_i. $\hat{u}^*(t; e(t_i))$ is applied to the system until the next sampling time t_{i+1}, which can be described in (3.48) and the corresponding optimal value function is given by (3.49).

$$u(t; e(t_i)) = \hat{u}^*(t; e(t_i)), t \in [t_i, t_i + 1) \tag{3.48}$$

$$J^*(e(t_i)) = J^*(e(t_i)), \hat{u}^*(\cdot; e(t_i)) \tag{3.49}$$

At each sampling time instants, the control input $u(t; e(t_i))$ is recalculated using new state information. Thus, the control input $u(t; e(t_i))$ is a feedback.

3.2 Decentralized NMPC

In this section, DNMPC schemes for dynamic system and kinematic system will be introduced. The design is based on NMPC. Since there is no centralized computer in DNMPC scheme, the agents are allowed to communicate with each other and get the information of other agents.

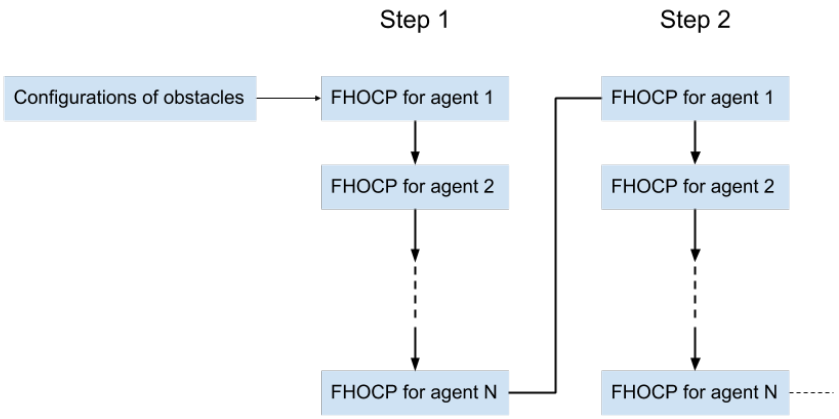


Figure 3.1: DNMPC structure

Define the priority variables $\delta_i, i \in \mathcal{N}$, which represents the priority of agents. Assume the priority variables are $\delta_1 > \delta_2 > ... > \delta_N$. Then the structure of DNMPC can be shown in 3.1, which will be explained later.

3.2.1 Dynamic System

At each sampling time, the DNMPC for dynamic system scheme will solve the problem for each agent in sequel. After the agent with a higher priority solve the optimization problem, the predicted states will be transmitted to the next agent in the priority order. For convenience, we assume that the leader is agent $i = 1$ and the rest agents are the followers without loss of generality. At each sampling time, agent 1 will solve the FHOPC problem subject to (2.46) when $i = 1$, with constraints about collision avoidance with obstacles and singularity avoidance. The solution of agent 1, i.e the control input sequence, will give a set of predicted variables for $q_1, \dot{q}_1$, which are denoted as $\hat{q}_1, \hat{\dot{q}}_1$, respectively. By employing forward kinematic equations and $\hat{q}_1, \hat{\dot{q}}_1$,

the pose and velocity of object in control horizon can be calculated via agent 1, which are denoted as $x_{o_1}(\hat{q_1})$ and $v_{o_1}(\hat{\dot{q}_1})$, respectively. After solving the FHOCP problem for agent 1, agent 1 will transmit $x_{o_1}(\hat{q_1})$ and $v_{o_1}(\hat{\dot{q}_1})$ to the other agents $\{2, ..., N\}$ according to the priority variables. The agents $\{2, ..., N\}$ will solve the FHOCP problem in sequel subject to (2.46) when $i = \{2, ..., N\}$, with state equality constraints about connection maintenance and inequality constraints about singularity avoidance and inter-agent collision avoidance. Every agent will transmit the predicted variables to the agents with lower priority after solving the FHOCP problem. The leader, i.e agent 1, will determine the path of the object without collision, while the other agents $\{2, ..., N\}$ will only follow the path provided by the leader and transport the object.

According to Chapter 2, the coupled agents-object dynamic system with feedback linearization is governed by (3.50).

$$\dot{x_i} = g_i(x, u_{i,r}) = \begin{bmatrix} g_{i_1}(x_i) \\ g_{i_2}(x_i, u_{i,r}) \end{bmatrix}, x_i(0) = x_0, \tag{3.50}$$

where $x_i = \left[x_{i_1}^\top, x_{i_2}^\top\right]^\top = \left[q_i^\top, \dot{q}_i^\top\right]^\top \in \mathbb{R}^{2n_i}$, $u_{i,r} \in \mathbb{R}^6$ and

$$g_{i_1}(x_i) = x_{i_2}, \tag{3.51}$$

$$g_{i_2}(x_i, u_{i,r}) = \widetilde{M}_i(q_i)(\widetilde{M}_i(q_i)\widetilde{M}_i(q_i)^\top)^{-1}\left[J_{o_i}^\top(q)\widetilde{u}_i - \widetilde{C}_i(q, \dot{q})\dot{q}_i - \widetilde{g}_i(q)\right], \tag{3.52}$$

where

$$\widetilde{u}_i = J_{o_i}^{-1}[(\widetilde{C}_i(q, \dot{q})\dot{q}_i + \widetilde{g}_i(q) + \widetilde{M}_i(q_i)u_{i,r}] \tag{3.53}$$

Feedback linearization is implemented by (3.53). Then, $g_{i_2}(x_i, u_{i,r})$ is linearized, which will reduce computational burden of NMPC solver greatly. $g_i(x, u_{i,r})$ is continuously differentiable in its domain, thus it is locally Lipschitz continuous.

Assumption 3.2.1. Assume that each agent has access to the measurements of $x_i \forall i \in \mathcal{N}$ for all times and the agents can communicate with each other without any time delays.

Furthermore, the error dynamics can be derived from (3.50), which will make the DNMPC scheme easier be described.

First of all, the destinations of the states are needed to be defined. $q_{des} = [q_{1,des}, ..., q_{N,des}]^\top$ is chosen according to (2.37) and (2.38), such that $x_o(t) = x_{o,des}$ and $\dot{x}_{o,des} = \dot{q}_{des} = 0$.

Then, the errors can be defined as (3.54).

$$e_i(t) = x_i(t) - x_{i,des} = \begin{bmatrix} q_i(t) \\ \dot{q}_i(t) \end{bmatrix} - \begin{bmatrix} q_{i,des} \\ \dot{q}_{i,des}(t) \end{bmatrix} = \begin{bmatrix} q_i(t) - q_{i,des} \\ \dot{q}_i(t) \end{bmatrix} \in \mathbb{R}^{2n_i} \tag{3.54}$$

The error dynamics can be transformed from (3.1), which is described by (3.55).

$$\dot{e}_i(t) = g_{e_i}(e_i(t), u_{i,r}(t)) = g_i(e_i(t) + x_{i,des}, u_{i,r}(t)), e_i(0) = x_i(0) - x_{i,des}. \tag{3.55}$$

Then the input constraints needs to be defined. There is a constraint for the input and velocity magnitude, which is described in (3.56).

$$|\dot{q}_{i_k}| \leq \bar{\dot{q}}_i, \forall k \in \{1, .., n_i\}, i \in \mathcal{N}, \tag{3.56}$$

where $\bar{\dot{q}}_i$ is is a positive constant. The joint space inputs of manipulators are defined as $\tau_i = J_i^\top(q_i)u_i \in \mathbb{R}^{n_i}, \forall i \in \mathbb{N}$, if we ignore over-actuated inputs. Assume the joint space input is bounded by (3.57).

$$|\tau_{i_k}| \leq \bar{\tau}_i, \forall k \in \{1, .., n_i\}, i \in \mathcal{N}, \tag{3.57}$$

where $\bar{\tau}_i$ is is a positive constant.

(3.57) can be translated into (3.59) by using the property $\sigma(J_i^\top)\|u_i\| \leq \|J_i^\top u_i\|$, where $\sigma(J_i^\top)$ is the minimum singular value of $J_i^\top$. Note that $\sigma(J_i^\top)$ is strictly positive if $q_i \in \widetilde{Q}_i$ is satisfied.

$$\|\tau_i\| \leq \bar{\tau}_i \iff \sigma_{min}(J_i^\top)\|u_i\| \leq \bar{\tau}_i, \forall i \in \mathcal{N} \tag{3.58}$$

Then, the input constraints in joint space (3.57) is equivalent to (3.59).

$$\|u_i\| \leq \frac{\bar{\tau}_i}{\sigma_{min}(J_i^\top)}, \forall i \in \mathcal{N} \tag{3.59}$$

Then the input constraints of the error dynamic system (3.55) can be described by set $U_i \subseteq \mathbb{R}^6$, which is defined as (3.60).

$$U_i = \left\{ u_{i,r} \subseteq \mathbb{R}^6 : \|u_i\| \leq \frac{\bar{\tau}_i}{\sigma_{min}(J_i^\top)}, \forall i \in \mathcal{N} \right\} \tag{3.60}$$

For convenience, we define $[q_l]_{l \in \mathcal{N} \setminus \{i\}}$ as the stack vector of all $q_l, l \in \mathcal{N} \setminus \{i\}$ and the set for collision avoidance.

$$S_{o_i}(x_{o_i}) = \{x_{o_i} \in \mathbb{M} : \mathcal{C}_{o_i}(x_{o_i}) \cap \mathcal{O}_z \neq \emptyset, \forall z \in \mathcal{Z}\}$$

The state constraints of the dynamic system (3.50) for the leader agent, i.e agent 1, can be described by set $X_1 \subseteq \mathbb{R}^{2n_1}$, which is defined as (3.61). The state constraints of the dynamic system (3.50) for the follower agents, i.e agent $\{2, ..., N\}$, can be described by set $X_i \subseteq \mathbb{R}^{2n_i}$, which is defined as (3.62).

$$X_1([q_l]_{l\in\{2,...,N\}}) = \{x \subseteq \mathbb{R}^{2n_1} : \theta_o(t) \in \left[-\bar{\theta}, \bar{\theta}\right], \theta_{B_1}(t) \in \left[-\bar{\theta}, \bar{\theta}\right], |\dot{q}_{1_k}| \leq \bar{\dot{q}}_1,$$
$$q_1 \in \widetilde{Q}_1 \setminus (S_{1,o}(q_1)) \cup S_{1,A}([q_l]_{l\in\{2,...,N\}}), x_{o_1} \in \mathbb{R}^3 \setminus S_{o_1}(x_{o_1}), \forall t \in \mathbb{R}_{\geq 0}\} \tag{3.61}$$

$$X_i([q_l]_{l\in\mathcal{N}\setminus\{i\}}) = \{x \subseteq \mathbb{R}^{2n_i} : \theta_{B_i}(t) \in \left[-\bar{\theta}, \bar{\theta}\right], |\dot{q}_{i_k}| \leq \bar{\dot{q}}_i,$$
$$q_i \in \widetilde{Q}_i \setminus (S_{i,o}(q_i)) \cup S_{i,A}([q_l]_{l\in\mathcal{N}\setminus\{i\}}), \forall t \in \mathbb{R}_{\geq 0}\} \tag{3.62}$$

The state constraints of the error dynamic system (3.10) of the leader, i.e agent 1, can be described by set $E_1 \subseteq \mathbb{R}^{2n_1}$, which is defined as (3.63).

$$E_1 = \{e_1 \in \mathbb{R}^{2n_1} : e_1 \in X_1 \oplus (-x_{1,des})\} \tag{3.63}$$

DNMPC scheme is able to find a control input of the leader,i.e the control input sequence of agent 1, $u_{1,r}(t) \in U_1$ such that $lim_{t\to\infty}\|e_1(t)\| = 0$ and $e_1(t) \in E_1, \forall t \in \mathbb{R}_{\geq 0}$, while ensuring singularity avoidance and collision avoidance. The followers, i.e agent $\{2, ..., N\}$, contribute to the transportation of object according to load-sharing coefficients $c_2, ..., c_N$ and follow the trajectory of the object calculated by the leader.

The DNMPC scheme has a constant sampling time $0 < h < T_p$, where T_p is the prediction horizon of DNMPC. Then the sequence of sampling time $\{t_i\}_{t_i\geq 0}$ satisfies (3.64).

$$t_{i+1} = t_i + h, \forall t \geq 0 \tag{3.64}$$

At sampling time instants t_i, a finite-horizon open-loop optimal problem of the leader, which is define by (3.65), (3.66), (3.67), and (3.68), is solved according to the current state error $e_1(t_i)$. The solution is an optimal control signal $\hat{u}_{1,r}(t)$ in time interval $t \in [t_i, t_i + T_p]$, which will be implemented in between the sampling instants. For convenience, hat $\hat{\cdot}$ denotes the predicted variables. $\hat{e}_1(\cdot)$ is the predicted solution of (3.66) driven by $\hat{u}_{1,r} : [t_i, t_i + T_p] \to U_1$ under initial condition $e_1(t_i)$.

$$\min_{\hat{u}_{1,r}(\cdot)} J_1(e_1(t_i)) = \min_{\hat{u}_{1,r}(\cdot)} \{V_1(\hat{e}_1(t_i+T_p)) + \int_{t_i}^{t_i+T_p} [F_1(\hat{e}_1(s), \hat{u}_{1,r}(s))]\, ds\} \tag{3.65}$$

subject to:

$$\dot{\hat{e}}_1(s) = g_{i,e}(\hat{e}_1(s), \hat{u}_1(s)), \hat{e}_1(t_i) = e_1(t_i), \tag{3.66}$$

$$\hat{e}_1(s) \in E_1, \hat{u}_{1,r} \in U_1, s \in [t_i, t_i + T_p], \tag{3.67}$$

$$\hat{e}_1(t_i + T_p) \in \mathcal{E}_{1,f}, \tag{3.68}$$

where $F_1 : E_1 \times U_1 \to \mathbb{R}_{\geq 0}$ is the running cost term, which is defined as (3.69), $V_1 : E_1 \to \mathbb{R}_{>0}$ is the terminal penalty cost, which is given by (3.70), and $\mathcal{E}_{1,f}$ is the terminal set. V_1 and $\mathcal{E}_{1,f}$ are able to enforce the stability of the system. The terms Q_1, P_1, R_1 are chosen such that $Q_1 = diag\{\widetilde{q}_{1_1}, ..., \widetilde{q}_{1_{2n_1}}\} \in \mathbb{R}_{\geq 0}^{(2n_1)\times(2n_1)}$, $P_1 = diag\{\widetilde{p}_{1_1}, ..., \widetilde{p}_{1_{2n_1}}\} \in \mathbb{R}_{>0}^{(2n_1)\times(2n_1)}$, and $R_1 = diag\{\widetilde{r}_{1_1}, ..., \widetilde{r}_{1_6}\} \in \mathbb{R}_{>0}^{6\times 6}$, where $\widetilde{q}_{1_i} \in \mathbb{R}_{\geq 0}$, $\widetilde{p}_{1_i} \in \mathbb{R}_{>0}$, $\forall i \in \{1, ..., 2n_1\}$ and $\widetilde{r}_{1_j} \in \mathbb{R}_{>0}$, $\forall j \in \{1, ..., 6\}$.

$$F_1(e_1(t), u_1(t)) = e_1(t)^\top Q_1 e_1(t) + u_{1,r}(t)^\top R_1 u_{1,r}(t) \tag{3.69}$$

$$V_1(e(t)) = e_1(t)^\top P_1 e_1(t) \tag{3.70}$$

Note that the running cost function F_1 holds that $F_1(0, 0) = 0$ and (3.71).

$$m\|e_1\|^2 \leq m \left\| \begin{matrix} e_1 \\ u_{1,r} \end{matrix} \right\|^2 \leq F_1(e_1, u_{1,r}) \leq M \left\| \begin{matrix} e_1 \\ u_{1,r} \end{matrix} \right\|^2 \leq M\|e_1\|^2 \tag{3.71}$$

, where $m = min\{\widetilde{q}_{1_1}, ..., \widetilde{q}_{1_{2n_i}}, \widetilde{r}_{1_1}, ..., \widetilde{r}_{1_6}\}$ and $M = max\{\widetilde{q}_{1_1}, ..., \widetilde{q}_{1_{2n_i}}, \widetilde{r_{1_1}}, ..., \widetilde{r}_{1_6}\}$. According to Definition 3.0.1, $m\|e_1\|^2$ and $M\|e_1\|^2$ are $\mathcal{K}$ functions.

An optimal input for $t \in [t_i, t_i + T_p]$, which is denoted by $\hat{u}^*_{1,r}(t; e_1(t_i))$, is calculated by solving the optimal control problem, which is define by (3.65), (3.66), (3.67), and (3.68), at time t_i. $\hat{u}^*_{1,r}(t; e_1(t_i))$ is applied to the system until the next sampling time t_{i+1}, which can be described in (3.72) and the corresponding optimal value function is given by (3.73).

$$u_{1,r}(t; e_1(t_i)) = \hat{u}^*_{1,r}(t; e_1(t_i)), t \in [t_i, t_i + 1) \tag{3.72}$$

$$J_1{}^*(e_1(t_i)) = J_1^*(e_1(t_i)), \hat{u}^*_{1,r}(\cdot; e_1(t_i)) \tag{3.73}$$

At each sampling time instants, the control input $u_{1,r}(t; e_1(t_i))$ is re-calculated using new state information. Thus, the control input $u_{1,r}(t; e_1(t_i))$ is a feedback.

After the optimization problem of the leader is solved, the pose and velocity of object, x_{o_1} and v_{o_1}, will be calculated by the predicted variables of agent 1, q_1 and $\dot{q_1}$. x_{o_1} and v_{o_1} will be transmitted to the followers, i.e agent $\{2, ..., N\}$.

Then the followers will solve a finite-horizon open-loop optimal problem, which is define by (3.74), (3.75), (3.76), (3.77), and (3.78), is solved according to the current state error $e_i(t_i)$. The solution is an optimal control signal $\hat{u}_{i,r}(t)$ in time interval $t \in [t_i, t_i + T_p]$, which will be implemented in between the sampling instants. For convenience, hat $\hat{\cdot}$ denotes the predicted variables. $\hat{e}_i(\cdot)$ is the predicted solution of (3.75) driven by $\hat{u}_{i,r} : [t_i, t_i + T_p] \to U_i$ under initial condition $e_i(t_i)$.

$$\min_{\hat{u}_{i,r}(\cdot)} J_1(e_i(t_i)) = \min_{\hat{u}_{i,r}(\cdot)} \{V_i(\hat{e}_i(t_i + T_p)) + \int_{t_i}^{t_i+T_p} [F_i(\hat{e}_i(s), \hat{u}_{i,r}(s))]\, ds\} \tag{3.74}$$

subject to:

$$\dot{\hat{e}}_i(s) = g_{i,e}(\hat{e}_i(s), \hat{u}_{i,r}(s)), \hat{e}_i(t_i) = e_i(t_i), \tag{3.75}$$

$$\dot{\hat{x}}_{o_i}(\hat{q}_i(s)) = x_{o_1}(\hat{q}_1(s; \cdot)), \dot{\hat{v}}_{o_i}(\hat{q}_i(s), \dot{\hat{q}}_i(s)) = v_{o_1}(\hat{q}_1(s; \cdot), \dot{\hat{q}}_1(s; \cdot)) \tag{3.76}$$

$$\hat{e}_i(s) \in E_i, \hat{u}_{i,r} \in U_i, s \in [t_i, t_i + T_p], \tag{3.77}$$

$$\hat{e}_i(t_i + T_p) \in \mathcal{E}_{i,f}, \tag{3.78}$$

where $F_i : E_i \times U_i \to \mathbb{R}_{\geq 0}$ is the running cost term, which is defined as (3.69), $V_i : E_i \to \mathbb{R}_{>0}$ is the terminal penalty cost, which is given by (3.70), and $\mathcal{E}_{i,f}$ is the terminal set. V_i and $\mathcal{E}_{i,f}$ are able to enforce the stability of the system. The terms Q_i, P_i, R_i are chosen such that $Q_i = diag\{\widetilde{q}_{i_1}, ..., \widetilde{q}_{i_{2n_i}}\} \in \mathbb{R}_{\geq 0}^{(2n_i)\times(2n_i)}$, $P_i = diag\{\widetilde{p}_{i_1}, ..., \widetilde{p}_{i_{2n_i}}\} \in \mathbb{R}_{>0}^{(2n_i)\times(2n_i)}$, and $R_i = diag\{\widetilde{r}_{i_1}, ..., \widetilde{r}_{i_6}\} \in \mathbb{R}_{>0}^{6\times 6}$, where $\widetilde{q}_{i_k} \in \mathbb{R}_{\geq 0}$, $\widetilde{p}_{i_k} \in \mathbb{R}_{>0}$, $\forall k \in \{1, ..., 2n_i\}$ and $\widetilde{r}_{i_j} \in \mathbb{R}_{>0}$, $\forall j \in \{1, ..., 6\}$.

$$F_i(e_i(t), u_{i,r}(t)) = e_i(t)^\top Q_i e_i(t) + u_{i,r}(t)^\top R_i u_{i,r}(t) \tag{3.79}$$

$$V_i(e(t)) = e_i(t)^\top P_i e_i(t) \tag{3.80}$$

Note that the running cost function F_i holds that $F_i(0, 0) = 0$ and (3.81).

$$m\|e_i\|^2 \leq m \left\| \begin{matrix} e_i \\ u_{i,r} \end{matrix} \right\|^2 \leq F_i(e_i, u_{i,r}) \leq M \left\| \begin{matrix} e_i \\ u_{i,r} \end{matrix} \right\|^2 \leq M\|e_i\|^2, \tag{3.81}$$

where $m = min\{\widetilde{q}_{i_1}, ..., \widetilde{q}_{i_{2n_i}}, \widetilde{r}_{i_1}, ..., \widetilde{r}_{i_6}\}$ and $M = max\{\widetilde{q}_{i_1}, ..., \widetilde{q}_{i_{2n_i}}, \widetilde{r_{i_1}}, ..., \widetilde{r}_{i_6}\}$. According to Definition 3.0.1, $m\|e_i\|^2$ and $M\|e_i\|^2$ are $\mathcal{K}$ functions.

An optimal input for $t \in [t_i, t_i + T_p]$, which is denoted by $\hat{u}^*_{i,r}(t; e_i(t_i))$, is calculated by solving the optimal control problem, which is define by (3.74), (3.75), (3.76), (3.77), and (3.78), at time t_i. $\hat{u}^*_{i,r}(t; e_i(t_i))$ is applied to the system until the next sampling time t_{i+1}, which can be described in (3.82) and the corresponding optimal value function is given by (3.83).

$$u_{i,r}(t; e_i(t_i)) = \hat{u}^*_{i,r}(t; e_i(t_i)), t \in [t_i, t_i + 1) \tag{3.82}$$

$$J_i{}^*(e_i(t_i)) = J^*_i(e_i(t_i)), \hat{u}^*_{i,r}(\cdot; e_i(t_i)) \tag{3.83}$$

At each sampling time instants, the control input $u_{i,r}(t; e_i(t_i))$ is recalculated using new state information. Thus, the control input $u_{i,r}(t; e_i(t_i))$ is a feedback.

3.2.2 Kinematic System

At each sampling time, the DNMPC for kinematic system scheme will solve the problem for each agent in sequel. After the agent with a higher priority solve the optimization problem, the predicted states will be transmitted to the next agent in the priority order. For convenience, we assume that the leader is agent $i = 1$ and the rest agents are the followers without loss of generality. At each sampling time, agent 1 will solve the FHOPC problem subject to (3.84) when $i = 1$, with constraints about collision avoidance with obstacles and singularity avoidance. The solution of agent 1, i.e the control input sequence, will give a set of predicted variables for q_1, which are denoted as $\hat{q_1}$, respectively. By employing forward kinematic equations and $\hat{q_1}$, the pose of object in control horizon can be calculated via agent 1, which are denoted as $x_{o_1}(\hat{q_1}$, respectively. After solving the FHOPC problem for agent 1, agent 1 will transmit $x_{o_1}(\hat{q_1})$ to the other agents $\{2, ..., N\}$ according to the priority variables. The agents $\{2, ..., N\}$ will solve the FHOPC problem in sequel subject to (3.84) when $i = \{2, ..., N\}$,

with state equality constraints about connection maintenance and inequality constraints about singularity avoidance and inter-agent collision avoidance. Every agent will transmit the predicted variables to the agents with lower priority after solving the FHOPC problem. The leader, i.e agent 1, will determine the path of the object without collision, while the other agents $\{2, ..., N\}$ will only follow the path provided by the leader and transport the object.

According to Chapter 2, the coupled agents-object kinematic system is governed by (3.84).

$$\dot{x_i} = g_i(x_i, u_i) = I_{n_i}, x_i(0) = x_0, \tag{3.84}$$

where $x_i = \left[x_{i_1}^\top\right]^\top = \left[q_i^\top\right]^\top \in \mathbb{R}^{n_i}$, $u_i \in \mathbb{R}^{n_i}$.

$g_i(x_i, u_i)$ is continuously differentiable in its domain, thus it is locally Lipschitz continuous.

Assumption 3.2.2. Assume that each agent has access to the measurements of $x_i \forall i \in \mathcal{N}$ for all times and the agents can communicate with each other without any time delays.

Furthermore, the error dynamics can be derived from (3.84), which will make the DNMPC scheme easier be described.

First of all, the destinations of the states are needed to be defined. $q_{des} = [q_{1,des}, ..., q_{N,des}]^\top$ is chosen according to (2.37) and (2.38), such that $x_o(t) = x_{o,des}$ and $\dot{x}_{o,des} = \dot{q}_{des} = 0$.

Then, the errors can be defined as (3.85).

$$e_i(t) = x_i(t) - x_{i,des} = q_i(t) - q_{i,des} \in \mathbb{R}^{2n_i} \tag{3.85}$$

The error dynamics can be transformed from (3.28), which is described by (3.86).

$$\dot{e_i}(t) = g_{e_i}(e_i(t), u_i(t)) = g(e_i(t) + x_{i,des}, u_i(t)), e_i(0) = x_i(0) - x_{i,des}. \tag{3.86}$$

Then the input constraints needs to be defined. There is a constraint for the input and velocity magnitude, which is described in (3.87).

$$|\dot{q}_{i_k}| \leq \bar{\dot{q}}_i, \forall k \in \{1, .., n_i\}, i \in \mathcal{N} \tag{3.87}$$

, where $\bar{\dot{q}}_i$ is is a positive constant.

Then the input constraints of the error kinematic system (3.86) can be described by set $U_i \subseteq \mathbb{R}^{n_i}$, which is defined as (3.88).

$$U_i = \{u_i \subseteq \mathbb{R}^{n_i} : |\dot{q}_{i_k}| \leq \bar{\dot{q}}_i, \forall k \in \{1, .., n_i\}, i \in \mathcal{N}\} \tag{3.88}$$

For convenience, we define $[q_l]_{l \in \mathcal{N} \setminus \{i\}}$ as the stack vector of all $q_l, l \in \mathcal{N} \setminus \{i\}$ and the set for collision avoidance.

$$S_{o_i}(x_{o_i}) = \{x_{o_i} \in \mathbb{M} : \mathcal{C}_{o_i}(x_{o_i}) \cap \mathcal{O}_z \neq \emptyset, \forall z \in \mathcal{Z}\}$$

The state constraints of the dynamic system (3.84) for the leader agent, i.e agent 1, can be described by set $X_1 \subseteq \mathbb{R}^{n_1}$, which is defined as (3.89). The state constraints of the dynamic system (3.84) for the follower agents, i.e agent $\{2, ..., N\}$, can be described by set $X_i \subseteq \mathbb{R}^{n_i}$, which is defined as (3.90).

$$\begin{aligned} X_1([q_l]_{l \in \{2,...,N\}}) = \{x \subseteq \mathbb{R}^{2n_1} : \theta_o(t) \in \left[-\bar{\theta}, \bar{\theta}\right], \theta_{B_1}(t) \in \left[-\bar{\theta}, \bar{\theta}\right], |\dot{q}_{1_k}| \leq \bar{\dot{q}}_1, \\ q_1 \in \widetilde{Q}_1 \setminus (S_{1,o}(q_1)) \cup S_{1,A}([q_l]_{l \in \{2,...,N\}}), x_{o_1} \in \mathbb{R}^3 \setminus S_{o_1}(x_{o_1}), \forall t \in \mathbb{R}_{\geq 0}\} \end{aligned} \tag{3.89}$$

$$\begin{aligned} X_i([q_l]_{l \in \mathcal{N} \setminus \{i\}}) = \{x \subseteq \mathbb{R}^{2n_i} : \theta_{B_i}(t) \in \left[-\bar{\theta}, \bar{\theta}\right], |\dot{q}_{i_k}| \leq \bar{\dot{q}}_i, \\ q_i \in \widetilde{Q}_i \setminus (S_{i,o}(q_i)) \cup S_{i,A}([q_l]_{l \in \mathcal{N} \setminus \{i\}}), \forall t \in \mathbb{R}_{\geq 0}\} \end{aligned} \tag{3.90}$$

The state constraints of the error kinematic system (3.35) of the leader, i.e agent 1, can be described by set $E_1 \subseteq \mathbb{R}^{n_1}$, which is defined as (3.91).

$$E_1 = \{e_1 \in \mathbb{R}^{n_1} : e_1 \in X_1 \oplus (-x_{1,des})\} \tag{3.91}$$

DNMPC scheme is able to find a control input of the leader,i.e the control input sequence of agent 1, $u_1(t) \in U_1$ such that $lim_{t \to \infty} \|e_1(t)\| = 0$ and $e_1(t) \in E_1, \forall t \in \mathbb{R}_{\geq 0}$, while ensuring singularity avoidance and collision avoidance. The followers, i.e agent $\{2, ..., N\}$, contribute to the transportation of object according to load-sharing coefficients $c_2, ..., c_N$ and follow the trajectory of the object calculated by the leader.

The DNMPC scheme has a constant sampling time $0 < h < T_p$, where T_p is the prediction horizon of DNMPC. Then the sequence of sampling time $\{t_i\}_{t_i \geq 0}$ satisfies (3.92).

$$t_{i+1} = t_i + h, \forall t \geq 0 \tag{3.92}$$

At sampling time instants t_i, a finite-horizon open-loop optimal problem of the leader, which is define by (3.93), (3.94), (3.95), and

(3.96), is solved according to the current state error $e_1(t_i)$. The solution is an optimal control signal $\hat{u}_1(t)$ in time interval $t \in [t_i, t_i + T_p]$, which will be implemented in between the sampling instants. For convenience, hat $\hat{\cdot}$ denotes the predicted variables. $\hat{e}_1(\cdot)$ is the predicted solution of (3.94) driven by $\hat{u}_1 : [t_i, t_i + T_p] \to U_1$ under initial condition $e_1(t_i)$.

$$\min_{\hat{u}_1(\cdot)} J_1(e_1(t_i)) \;=\; \min_{\hat{u_1}(\cdot)} \{V_1(\hat{e}_1(t_i + T_p)) + \int_{t_i}^{t_i+T_p} [F_1(\hat{e}_1(s), \hat{u}_1(s))]\, ds\} \tag{3.93}$$

subject to:

$$\dot{\hat{e}}_1(s) = g_{i,e}(\hat{e}_1(s), \hat{u}_1(s)), \hat{e}_1(t_i) = e_1(t_i), \tag{3.94}$$

$$\hat{e}_1(s) \in E_1, \hat{u}_1 \in U_1, s \in [t_i, t_i + T_p]\,, \tag{3.95}$$

$$\hat{e}_1(t_i + T_p) \in \mathcal{E}_{1,f}, \tag{3.96}$$

where $F_1 : E_1 \times U_1 \to \mathbb{R}_{\geq 0}$ is the running cost term, which is defined as (3.97), $V_1 : E_1 \to \mathbb{R}_{>0}$ is the terminal penalty cost, which is given by (3.98), and $\mathcal{E}_{1,f}$ is the terminal set. V_1 and $\mathcal{E}_{1,f}$ are able to enforce the stability of the system. The terms Q_1, P_1, R_1 are chosen such that $Q_1 = diag\{\widetilde{q}_{1_1}, ..., \widetilde{q}_{1_{n_1}}\} \in \mathbb{R}_{\geq 0}^{(n_1)\times(n_1)}$, $P_1 = diag\{\widetilde{p}_{1_1}, ..., \widetilde{p}_{1_{n_1}}\} \in \mathbb{R}_{>0}^{(n_1)\times(n_1)}$, and $R_1 = diag\{\widetilde{r}_{1_1}, ..., \widetilde{r}_{1_{n_1}}\} \in \mathbb{R}_{>0}^{n_1\times n_1}$, where $\widetilde{q}_{1_i} \in \mathbb{R}_{\geq 0}$, $\widetilde{p}_{1_i} \in \mathbb{R}_{>0}$, $\forall i \in \{1, ..., n_1\}$ and $\widetilde{r}_{1_j} \in \mathbb{R}_{>0}$, $\forall j \in \{1, ..., n_1\}$.

$$F_1(e_1(t), u_1(t)) = e_1(t)^\top Q_1 e_1(t) + u_1(t)^\top R_1 u_1(t) \tag{3.97}$$

$$V_1(e(t)) = e_1(t)^\top P_1 e_1(t) \tag{3.98}$$

Note that the running cost function F_1 holds that $F_1(0, 0) = 0$ and (3.99).

$$m\|e_1\|^2 \leq m \left\| \begin{matrix} e_1 \\ u_1 \end{matrix} \right\|^2 \leq F_1(e_1, u_1) \leq M \left\| \begin{matrix} e_1 \\ u_1 \end{matrix} \right\|^2 \leq M\|e_1\|^2, \tag{3.99}$$

where $m = min\{\widetilde{q}_{1_1}, ..., \widetilde{q}_{1_{n_i}}, \widetilde{r}_{1_1}, ..., \widetilde{r}_{1_6}\}$ and $M = max\{\widetilde{q}_{1_1}, ..., \widetilde{q}_{1_{n_i}}, \widetilde{r_{1_1}}, ..., \widetilde{r}_{1_{n_i}}\}$. According to Definition 3.0.1, $m\|e_1\|^2$ and $M\|e_1\|^2$ are $\mathcal{K}$ functions.

An optimal input for $t \in [t_i, t_i + T_p]$, which is denoted by $\hat{u}_1^*(t; e_1(t_i))$, is calculated by solving the optimal control problem, which is define by (3.93), (3.94), (3.95), and (3.96), at time t_i. $\hat{u}_1^*(t; e_1(t_i))$ is applied to

the system until the next sampling time t_{i+1}, which can be described in (3.100) and the corresponding optimal value function is given by (3.101).

$$u_1(t; e_1(t_i)) = \hat{u}_1^*(t; e_1(t_i)), t \in [t_i, t_i + 1) \tag{3.100}$$

$$J_1{}^*(e_1(t_i)) = J_1^*(e_1(t_i)), \hat{u}_1^*(\cdot; e_1(t_i)) \tag{3.101}$$

At each sampling time instants, the control input $u_1(t; e_1(t_i))$ is re-calculated using new state information. Thus, the control input $u_1(t; e_1(t_i))$ is a feedback.

After the optimization problem of the leader is solved, the pose of object, x_{o_1}, will be calculated by the predicted variables of agent 1. x_{o_1} will be transmitted to the followers, i.e agent $\{2, ..., N\}$.

Then the followers will solve a finite-horizon open-loop optimal problem, which is define by (3.102), (3.103), (3.104), (3.105), and (3.106), is solved according to the current state error $e_i(t_i)$. The solution is an optimal control signal $\hat{u}_i(t)$ in time interval $t \in [t_i, t_i + T_p]$, which will be implemented in between the sampling instants. For convenience, hat $\hat{\cdot}$ denotes the predicted variables. $\hat{e}_i(\cdot)$ is the predicted solution of (3.75) driven by $\hat{u}_i : [t_i, t_i + T_p] \to U_i$ under initial condition $e_i(t_i)$.

$$\min_{\hat{u}_i(\cdot)} J_1(e_i(t_i)) \;=\; \min_{\hat{u}_i(\cdot)} \{V_i(\hat{e}_i(t_i + T_p)) + \int_{t_i}^{t_i+T_p} [F_i(\hat{e}_i(s), \hat{u}_i(s))]\, ds\} \tag{3.102}$$

subject to:

$$\dot{\hat{e}}_i(s) = g_{i,e}(\hat{e}_i(s), \hat{u}_i(s)), \hat{e}_i(t_i) = e_i(t_i), \tag{3.103}$$

$$\dot{\hat{x}}_{o_i}(\hat{q}_i(s)) = x_{o_1}(\hat{q}_1(s; \cdot)), \dot{\hat{v}}_{o_i}(\hat{q}_i(s), \dot{\hat{q}}_i(s)) = v_{o_1}(\hat{q}_1(s; \cdot), \dot{\hat{q}}_1(s; \cdot)) \tag{3.104}$$

$$\hat{e}_i(s) \in E_i, \hat{u}_i \in U_i, s \in [t_i, t_i + T_p], \tag{3.105}$$

$$\hat{e}_i(t_i + T_p) \in \mathcal{E}_{i,f}, \tag{3.106}$$

where $F_i : E_i \times U_i \to \mathbb{R}_{\geq 0}$ is the running cost term, which is defined as (3.97), $V_i : E_i \to \mathbb{R}_{>0}$ is the terminal penalty cost, which is given by (3.98), and $\mathcal{E}_{i,f}$ is the terminal set. V_i and $\mathcal{E}_{i,f}$ are able to enforce the stability of the system. The terms Q_i, P_i, R_i are chosen such that $Q_i = diag\{\widetilde{q}_{i_1}, ..., \widetilde{q}_{i_{n_i}}\} \in \mathbb{R}_{\geq 0}^{(n_i)\times(n_i)}$, $P_i = diag\{\widetilde{p}_{i_1}, ..., \widetilde{p}_{i_{n_i}}\} \in \mathbb{R}_{>0}^{(n_i)\times(n_i)}$,

and $R_i = diag\{\widetilde{r}_{i_1}, ..., \widetilde{r}_{i_{n_i}}\} \in \mathbb{R}_{>0}^{n_i \times n_i}$, where $\widetilde{q}_{i_k} \in \mathbb{R}_{\geq 0}$, $\widetilde{p}_{i_k} \in \mathbb{R}_{>0}$, $\forall k \in \{1, ..., n_i\}$ and $\widetilde{r}_{i_j} \in \mathbb{R}_{>0}$, $\forall j \in \{1, ..., n_i\}$.

$$F_i(e_i(t), u_i(t)) = e_i(t)^\top Q_i e_i(t) + u_i(t)^\top R_i u_i(t) \tag{3.107}$$

$$V_i(e(t)) = e_i(t)^\top P_i e_i(t) \tag{3.108}$$

Note that the running cost function F_i holds that $F_i(0, 0) = 0$ and (3.109).

$$m\|e_i\|^2 \leq m \left\| \begin{matrix} e_i \\ u_i \end{matrix} \right\|^2 \leq F_i(e_i, u_i) \leq M \left\| \begin{matrix} e_i \\ u_i \end{matrix} \right\|^2 \leq M\|e_i\|^2, \tag{3.109}$$

where $m = min\{\widetilde{q}_{i_1}, ..., \widetilde{q}_{i_{n_i}}, \widetilde{r}_{i_1}, ..., \widetilde{r}_{i_{n_i}}\}$ and $M = max\{\widetilde{q}_{i_1}, ..., \widetilde{q}_{i_{n_i}}, \widetilde{r_{i_1}}, ..., \widetilde{r}_{i_{n_i}}\}$. According to Definition 3.0.1, $m\|e_i\|^2$ and $M\|e_i\|^2$ are $\mathcal{K}$ functions.

An optimal input for $t \in [t_i, t_i + T_p]$, which is denoted by $\hat{u}_i^*(t; e_i(t_i))$, is calculated by solving the optimal control problem, which is define by (3.102), (3.103), (3.104), (3.105), and (3.106), at time t_i. $\hat{u}_i^*(t; e_i(t_i))$ is applied to the system until the next sampling time t_{i+1}, which can be described in (3.110) and the corresponding optimal value function is given by (3.111).

$$u_i(t; e_i(t_i)) = \hat{u}_i^*(t; e_i(t_i)), t \in [t_i, t_i + 1) \tag{3.110}$$

$$J_i{}^*(e_i(t_i)) = J_i^*(e_i(t_i)), \hat{u}_i^*(\cdot; e_i(t_i)) \tag{3.111}$$

At each sampling time instants, the control input $u_i(t; e_i(t_i))$ is re-calculated using new state information. Thus, the control input $u_i(t; e_i(t_i))$ is a feedback.

3.3 Stability

The stability proof follows [45]. The proof of CNMPC for kinematic system is almost the same as CNMPC for dynamic system, the only difference is the system. The proof of DNMPC is also similar as CNMPC case. Thus the proof of DCNMPC and proof of CNMPC for dynamic system are omitted.

The proof for CNMPC of dynamic system is presented hereafter.

$e(s; u(\cdot, e(t_1))), s \in [t_1, t_2]$ denotes the solution of (3.10), starting at time t_1 with initial condition $e(t_1)$, applying a control input $u : [t_1, t_2] \rightarrow$

U. $\hat{e}(t_i + s; u(\cdot), e(t_i))$ is the predicted state of the system (3.10) at time $t_i + s, s > 0$, which is based on the measurement of the state $e(t_i)$ at time t_i when a control input $u(\cdot; e(t_i))$ is applied for the time period $[t_i, t_i + s]$. Obviously, (3.112) holds.

$$e(t_i) = \hat{e}(t_i; u(\cdot), e(t_i)) \tag{3.112}$$

Definition 3.3.1 (Admissible control input [25])**.** A control input $u : [0, T_p] \to \mathbb{R}^6$ for a state e_0 is admissible, if the following hold:

- $u(\cdot)$ is piecewise continuous;
- $u(s) \in U, \forall s \in [0, T_p]$;
- $e(s; u(\cdot), e_0) \in E, \forall s \in [0, T_p]$;
- $e(T_p; u(\cdot), e_0)) \in \mathcal{E}_f$;

Lemma 3.3.1. The terminal penalty function $V(\cdot)$ is Lipschitz continues in $\mathcal{E}_f$, with Lipschitz constant $L_v = 2\epsilon_0\sigma_{max}(P), \forall e(t) \in \mathcal{E}_f$, where $\epsilon = \sup_{e\in E} \|e\|$.

Proof. For every $e(t) \in \mathcal{E}_f$, the following holds:

$$\begin{aligned} |V(e_1) - V(e_2)| &= |e_1^\top P e_1 - e_2^\top P e_2| = |e_1^\top P e_1 + e_1^\top P e_2 - e_1^\top P e_2 e_2^\top P e_2| \\ &= |e_1^\top P(e_1 - e_2) - e_2^\top P(e_1 - e_2)| \leq |e_1^\top P(e_1 - e_2)| + |e_2^\top P(e_1 - e_2)| \end{aligned} \tag{3.113}$$

By using the property that: $|x^\top A y| \leq \sigma_{max}(A)\|x\|\|y\|, \forall x, y \in \mathbb{R}^n, A \in \mathbb{R}^{n\times n}$, (3.113) can be written as:

$$|V(e_1) - V(e_2)| \leq \sigma_{max}(P)\|e_1\|\|e_1 - e_2\| + \sigma_{max}(P)\|e_2\|\|e_1 - e_2\| =$$
$$\sigma_{max}(P)(\|e_1\|+\|e_2\|)\|e_1-e_2\| \leq \sigma_{max}(P)(\epsilon_0+\epsilon_0)\|e_1-e_2\| = 2\epsilon_0\sigma_{max}(P)$$

The proof is done. □

The stability of CNMPC of dynamic system can be guaranteed by Theorem (3.3.1).

Theorem 3.3.1. Suppose that:

- The optimal control problem, which is define by (3.19), (3.20), (3.21), and (3.22) is feasible for the initial time $t = 0$;

- The terminal set $\mathcal{E}_f$ is closed, with $0_{n+12} \in \mathcal{E}_f$;
- The terminal set $\mathcal{E}_f$ is chosen such that there exists an admissible control input $u_f : [0, h] \to U$ such that for all $e(s) \in \mathcal{E}_f$ it holds that:
  - $e(s) \in \mathcal{E}_f, \forall s \in [0, h]$;
  - $\frac{\partial V}{\partial e} f_e(e(s), u_f(s)) + F(e(s), u_f(s)) \leq 0, \forall s \in [0, h]$;

Then the closed loop system (3.10) converges to the set $\mathcal{E}_f$ for $t \to \infty$ under the control input (3.26) and Assumption 2.3.1.

Proof. The proof consists of two parts: feasibility analysis and convergence analysis.

Feasibility Analysis

Feasibility analysis is to prove that initial feasibility implies feasibility afterwards.

Consider any sampling time instant t_i for which a solution exists. In between t_i and t_{i+1}, the optimal control input $\hat{u}^*(s; e(t_i)), s \in [t_i, t_{i+1}]$ is applied. According to (3.112), it holds that:

$$e(t_{i+1}) = \hat{e}(t_{i+1}; \hat{u}^*(\cdot; e(t_i)), e(t_i)).$$

The rest piece of the optimal control input $\hat{u}^*(s; e(t_i)), s \in [t_{i+1}, t_i + T_p]$ satisfies the state constraints E and input constraints U, respectively. According to $\hat{e}(t_i + T_p; \hat{u}^*(\cdot; e(t_i)), e(t_i))$ and and the assumptions of Theorem (3.3.1), we know that there exists at least one control input $u_f(\cdot)$ that renders that the set $\mathcal{E}_f$ is invariant over sampling time interval h for all $e(t) \in \mathcal{E}_f$. A feasible control input $\bar{u}(\cdot; e(t_{i+1}))$ at time instant t_{t+1}, may be (3.114).

$$\bar{u}(\cdot; e(t_{i+1})) = \begin{cases} \hat{u}^*(s; e(t_i)), & s \in [t_{i+1}, t_i + T_p], \\ u_f(\hat{e}(t_i + T_p; u^*(\cdot), e(t_i))), & s \in [t_i + T_p, t_{i+1} + T_p]. \end{cases} \tag{3.114}$$

Thus, from feasibility of $\hat{h}^*(s, e(t_i))$ and the fact that $u_f(e(t)) \in U$, for all $e(t) \in \mathcal{E}_f$, it follows that:

$$\bar{u}(\cdot; e(t_{i+1})) \in U, \forall s \in [t_{i+1}, t_i + T_p].$$

Hence, the feasibility at time t_i implies feasibility at time t_{t+1}. Thus, if the optimal control problem, which is define by (3.19), (3.20), (3.21),

and (3.22), is feasible for the initial time $t = 0$, it remains feasible $\forall t \geq 0$.

Convergence Analysis

Convergence analysis is to prove the error $e(t)$ converges to the terminal set $\mathcal{E}_f$. In order to prove convergence, a value function must be proven to be decreasing along the solution trajectories starting at a sampling time t_i. Consider the optimal value function $J^*(e(t_i))$ given by (3.26). The cost function of the feasible control input is denoted as $\bar{J}(e(t_{i+1}))$, which is defined by (3.115).

$$\bar{J}(e(t_{i+1})) = \bar{J}(e(t_{i+1}), \bar{u}(\cdot; e(t_{i+1}))), \tag{3.115}$$

where $t_{i+1} = t_i + h$.

For convenience, the predicted state e at time s based on the measurement of the state e at time t_{i+1} is denoted as $e_1(s)$, while using the feasible control input $\bar{u}(s; e(t_{i+1}))$, which is denoted as $u_1(s)$. $e_1(s)$ and $u_1(s)$ are defined by (3.116) and (3.117).

$$u_1(s) = \bar{u}(s; e(t_{i+1})) \tag{3.116}$$

$$e_1(s) = \bar{e}(s; u_1(s), e(t_{i+1})), s > t_{i+1} \tag{3.117}$$

Also define $u_2(s)$ and $e_2(s)$ for convenience.

$$u_2(s) = \hat{u}^*(s; e(t_i)) \tag{3.118}$$

$$e_2(s) = \hat{e}(s; u_2(s), e(t_i)), s > t_{i+1} \tag{3.119}$$

The difference between the optimal and feasible cost is given by (3.120), by using (3.19), (3.27) and (3.115).

$$\begin{aligned}
&\bar{J}(e(t_{i+1})) - J^*(e(t_i)) = \\
&\qquad V(e_1(t_{i+1} + T_p)) + \int_{t_{i+1}}^{t_{i+1}+T_p} [F(e_1(s), u_1(s))]\, ds \\
&\qquad - V(e_2(t_i + T_p)) - \int_{t_i}^{t_i+T_p} [F(e_2(s), u_2(s))]\, ds \\
&= V(e_1(t_{i+1}+T_p)) + \int_{t_{i+1}}^{t_i+T_p} [F(e_1(s), u_1(s))]\, ds + \int_{t_i+T_p}^{t_{i+1}+T_p} [F(e_1(s), u_1(s))]\, ds \\
&- V(e_2(t_i+T_p)) - \int_{t_i}^{t_{i+1}} [F(e_2(s), u_2(s))]\, ds - \int_{t_{i+1}}^{t_i+T_p} [F(e_2(s), u_2(s))]\, ds
\end{aligned} \tag{3.120}$$

From (3.114), (3.121) holds.

$$\bar{u}(s; e(t_{i+1})) = \hat{u}^*(s; e(t_i)), \forall s \in [t_{i+1}, t_i + T_p] \tag{3.121}$$

By using (3.116), (3.117), (3.118), (3.119) and (3.121), (3.122) holds, which implies that (3.123).

$$u_1(s) = u_2(s) = \bar{u}(s), \forall s \in [t_{i+1}, t_i + T_p] \tag{3.122}$$

$$e_1(s) = e_2(s), \forall s \in [t_{i+1}, t_i + T_p] \tag{3.123}$$

The combination of (3.122) and (3.123) implies that:

$$F(e_1(s), u_1(s)) = F(e_2(s), u_2(s)), \forall s \in [t_{i+1}, t_i + T_p], \tag{3.124}$$

which implies that:

$$\int_{t_{i+1}}^{t_i+T_p} [F(e_1(s), u_1(s))]\, ds = \int_{t_{i+1}}^{t_i+T_p} [F(e_2(s), u_2(s))]\, ds, \forall s \in [t_{i+1}, t_i+T_p] \tag{3.125}$$

By employing (3.125), (3.120) becomes:

$$\begin{aligned} \bar{J}(e(t_{i+1})) - J^*(e(t_i)) = V(e_1(t_{i+1} + T_p)) + \int_{t_i+T_p}^{t_{i+1}+T_p} [F(e_1(s), u_1(s))]\, ds \\ - V(e_2(t_i + T_p)) - \int_{t_i}^{t_{i+1}} [F(e_2(s), u_2(s))]\, ds \end{aligned} \tag{3.126}$$

Note that $t_{i+1} + T_p - (t_i + T_p) = t_{i+1} - t_i = h$ and the inequality in Theorem (3.3.1) holds for one sampling period h. By integrating the inequality in Theorem (3.3.1) from $t_i + T_p$ to $t_{i+1} + T_p$, we can get the following:

$$\int_{t_i+T_p}^{t_{i+1}+T_p} \left[\frac{\partial V}{\partial e} f_e(e_1(s), u_1(s)) + F(e_1(s), u_1(s))\right] ds \leq 0$$

$$\Leftrightarrow \int_{t_i+T_p}^{t_{i+1}+T_p} \left[\dot{V}(e_1(s))\right] ds + \int_{t_i+T_p}^{t_{i+1}+T_p} [F(e_1(s), u_1(s))]\, ds \leq 0$$

$$\Leftrightarrow V(e_1(t_{i+1} + T_p)) - V(e_1(t_i + T_p)) + \int_{t_i+T_p}^{t_{i+1}+T_p} [F(e_1(s), u_1(s))]\, ds \leq 0$$

$$V(e_1(t_{i+1} + T_p)) - V(e_1(t_i + T_p)) + \int_{t_i+T_p}^{t_{i+1}+T_p} [F(e_1(s), u_1(s))]\, ds \leq V(e_2(t_i + T_p)) - V(e_2(t_i + T_p))$$

$$\Leftrightarrow V(e_1(t_{i+1} + T_p)) + \int_{t_i+T_p}^{t_{i+1}+T_p} [F(e_1(s), u_1(s))]\, ds - V(e_2(t_i + T_p)) \leq V(e_1(t_i + T_p)) - V(e_2(t_i + T_p))$$

By employing the property $y \leq |y|, \forall y \in \mathbb{R}$, we have:

$$V(e_1(t_{i+1}+T_p)) + \int_{t_i+T_p}^{t_{i+1}+T_p} [F(e_1(s),u_1(s))]\,ds - V(e_2(t_i+T_p)) \leq |V(e_1(t_i+T_p)) - V(e_2(t_i+T_p))| \quad (3.127)$$

By employing Lemma 3.3.1, we have that:

$$|V(e_1(t_{i+1}+T_p)) - V(e_2(t_{i+1}+T_p))| \leq L_V \|e_1(t_{i+1}+T_p) - e_2(t_{i+1}+T_p)\| \quad (3.128)$$

By combining (3.127) and (3.128), we have:

$$V(e_1(t_{i+1}+T_p)) + \int_{t_i+T_p}^{t_{i+1}+T_p} [F(e_1(s),u_1(s))]\,ds - V(e_2(t_i+T_p)) \leq L_V \|e_1(t_{i+1}+T_p) - e_2(t_{i+1}+T_p)\| \quad (3.129)$$

For $s = t_i + T_p$, (3.123) gives:

$$e_1(t_i+T_p) = e_2(t_i+T_p) \quad (3.130)$$

By using (3.129) and (3.130), we get:

$$V(e_1(t_{i+1}+T_p)) + \int_{t_i+T_p}^{t_{i+1}+T_p} [F(e_1(s),u_1(s))]\,ds - V(e_2(t_i+T_p)) \leq 0 \quad (3.131)$$

By combining (3.126) and (3.131), we have:

$$\bar{J}(e(t_{i+1})) - J^*(e(t_i)) \leq -\int_{t_i}^{t_{i+1}} [F(e_2(s),u_2(s))]\,ds \quad (3.132)$$

By substituting $e = e_2(s), u = u_2(s)$ in (3.25), we have:

$$F(e_2(s),u_2(s)) \geq m\|e_2(s)\|^2 \quad (3.133)$$

By integrating (3.133) from t_i to t_{i+1}, we get:

$$\int_{t_i}^{t_{i+1}} [F(e_2(s),u_2(s))]\,ds \geq m\int_{t_i}^{t_{i+1}} \|e_2(s)\|^2 ds \Leftrightarrow -\int_{t_i}^{t_{i+1}} [F(e_2(s),u_2(s))]\,ds \leq -m\int_{t_i}^{t_{i+1}} \|e_2(s)\|^2 ds \quad (3.134)$$

By combining (3.134) and (3.132), we get:

$$\bar{J}(e(t_{i+1})) - J^*(e(t_i)) \leq -m \int_{t_i}^{t_{i+1}} \|e_2(s)\|^2 ds \tag{3.135}$$

Obviously, the optimal solution at time t_{i+1}, i.e $J^*(e(t_{i+1}))$ will not be worse than the feasible solution at the same time, i.e $\bar{J}(e(t_{i+1}))$. Therefore, (3.135) implies:

$$J^*(e(t_{i+1})) - J^*(e(t_i)) \leq -m \int_{t_i}^{t_{i+1}} \|e_2(s)\|^2 ds \leq 0 \tag{3.136}$$

By using the face that $\int_{t_0}^{t_1} \|e_2(s)\|^2 ds = \sum_{j=0}^{i-1} \int_{t_j}^{t_{j+1}} \|e_2(s)\|^2 ds$, we get:

$$J^*(e(t_{i+1})) - J^*(e(t_i)) \leq -m \int_{t_0}^{t_{i+1}} \|e_2(s)\|^2 ds + \sum_{j=0}^{i-1} \int_{t_j}^{t_{j+1}} \|e_2(s)\|^2 ds \tag{3.137}$$

By using $t_i = h \cdot i, t_{i+1} = h \cdot (i+1), \forall i \geq 0$ and induction, (3.137) can be written as:

$$J^*(e(t_i)) - J^*(e(t_0)) \leq -m \int_{t_0}^{t_i} \|e_2(s)\|^2 ds \tag{3.138}$$

When $t_0 = 0$, we obtain:

$$J^*(e(t_i)) \leq J^*(e(0)) - m \int_0^{t_i} \|e_2(s)\|^2 ds, \tag{3.139}$$

which implies:

$$J^*(e(t_i)) \leq J^*(e(0)) \tag{3.140}$$

By combining (3.136) and (3.140), we get:

$$J^*(e(t_{i+1})) \leq J^*(e(t_i)) \leq J^*(e(0)), \forall t_i = i \cdot h, i \geq 0 \tag{3.141}$$

Thus, the value function $J^*(e(t_i))$ is proved to be non-increasing for all sampling times. Define the function:

$$V(e(t)) = J^*(e(s)) \leq J^*(e(0)), t \in \mathbb{R}_{\geq 0}, \tag{3.142}$$

where$s = max\{t_i : t_i \leq t\}$. (3.142) implies $V(e(t))$ is bounded, since $J^*(e(0))$ is bounded. Since $e(t) \in E$ and $u(t) \in U$, which means $e(t)$ and $u(t)$ are bounded, $\dot{e}(t)$ is bounded. From (3.139), we have:

$$V(e(t)) = J^*(e(s)) \leq J^*(e(0)) - m \int_0^s \|e_2(s)\|^2 ds,$$

which is equivalent to:

$$V(e(t)) \leq J^*(e(0)) - m \int_0^t \|e_2(s)\|^2 ds, \forall t \in \mathbb{R}_{\geq 0} \tag{3.143}$$

From (3.143), we have:

$$\int_0^t \|e_2(s)\|^2 ds \leq \frac{1}{m}[J^*(e(0)) - V(e(t))], \forall t \in \mathbb{R}_{\geq 0} \tag{3.144}$$

Since $J^*(e(0))$ and $V(e(t))$ are bounded, $\int_0^t \|e_2(s)\|^2 ds$ is bounded. Thus, by using 3.0.1, we have that $\|e_2(t)\| \to 0$ as $t \to \infty$, which implies:

$$\lim_{t\to\infty} \|e(t)\| = 0 \Rightarrow \lim_{t\to\infty} \|e(t)\| = \mathcal{E}_f \tag{3.145}$$

The proof is done. □

Chapter 4

Software Tools

There are several software tools, which can be used for simulation and experiment of model predictive control. With appropriate formulation of problems, the software tools are able to solve optimization problems using powerful solvers. Users can solve optimization problems easily by using appropriate software tools and solvers. Users need to choose software tools carefully, since different tools are suitable for different applications and problems.

In experiments, one important issue is how to develop a fast prototype. Robot Operating System (ROS) [49] provides a framework that allows users to finish a prototype of research easily.

In this chapter, several software tools and ROS are introduced.

4.1 Optimization tools

There are several online open source optimization toolboxes available for Matlab, Python and C++. Different toolboxes have different features and suitable for different problem formulations.

CVX [23, 24] is a Matlab-based modelling toolbox focusing on convex optimization problems, which can be used to solve linear MPC problems. CVX also supports geometric programming (GP) and mixed integer disciplined convex programming (MIDCP). Similarly, CVXPY [14] is a Python-based modelling language of convex optimization, which supports several open source solvers, such as ECOS, CVXOPT, and SCS. YALMIP [40] is also a Matlab-based toolbox, which supports around 20 kinds of solvers. The adequate solvers of YALMIP allow YALMIP to solve many optimization problems, such as linear

programming (LP), quadratic programming (QP), second order cone programming (SOCP), semidefinite programming, determinant maximization, mixed integer programming. CVXGEN [43] allows users automatically generate C code for embedded convex optimization described by linear programming (LP) and quadratic programming (QP). CasADi [3] is an open source symbolic tool for nonlinear optimization, which supports C++, Python and Octave. Authors in [25] also provide a Matlab-based toolbox, which is suitable for nonlinear optimization problems.

In this thesis, the MPC problem is not a convex problem, since it is a nonlinear optimization problem. Thus, CasADi is used for experiments and simulations of kinematic system and the toolbox provided by [25] is used for Matlab simulations of dynamic system in this thesis.

4.2 ROS

With the development of robotic industries, robotic systems are becoming more complex and the scale of robotic systems are continuously growing. Different robotic systems has different hardware structure and a single researcher or developer only has knowledge in a specific area. All of these challenges make the development of general software tools of robotic systems difficult for researchers.

Currently, other frameworks only focus on some specific robotic systems. However, ROS is a general robotic operating system, which can satisfy the requirements of large-scale integrative robotics research with different robotic systems. Compared with traditional robotic frameworks, which only focus on process management and scheduling, ROS also provides structured communications layer above the host operating systems. It provides more convenience for robotic researchers in both academia and industry. Thus, ROS is a convenient tool for robotic researchers.

4.2.1 Features

One of the features of ROS is its peer-to-peer connectivity. Usually, a traditional framework of robotic systems has a central server, which may cause unnecessary traffic flowing across wireless connection when different devices are connected in a a heterogenous network. The issue can be avoided by using peer-to-peer connectivity, especially when a

robotic system has a number of devices, processes and computation-intensive tasks.

Another advantage of ROS is its multi-lingual support. Instead of only supporting one specific language, ROS currently supports four different languages: C++, Python, Octave, and LISP. The multi-lingual feature allows users to choose programming languages according to their preferences, which provides the probability of saving programming and debugging time, and improving runtime efficiency. Thanks to the multi-lingual feature, different languages can be mixed in a message processing scheme, which provides more flexibility.

The tool-based structure provides improvement of stability and complexity management. Instead of using a monolithic development environment, ROS provides a number of small tools in separate modules. The tools provides a number of services, such as getting and setting configuration parameters, visualizing the peer-to-peer topology, graphically plotting message data, recording message data etc..

The codes in traditional frameworks have a difficulty to be reused due to their dependency on the frameworks. Compared with other frameworks, the codes in ROS are reusable due to its thin ideology. Users are encouraged to develop algorithms and drivers using standalone libraries that have no dependencies on ROS, which makes codes reusable in ROS.

ROS is an open-source and free tool under the terms of the BSD license, which allows both commercial usage and non-commercial usage. Due to the open-source and free feature, ROS has a good community where developers can provide their feedbacks and make ROS better.

4.2.2 Concepts

Nodes, messages, topics and services are the fundamental concepts of ROS.

Nodes

Nodes are computational possesses in ROS. A system may consist of many nodes, which can communicate with each other. The nodes are the smallest modules in ROS, which are suitable for peer-to-peer connectivity and large-scale robotic systems. In ROS, it is convenient to figure out the peer-to-peer connectivity by rendering a graph.

Messages

Nodes can communicate with each other via messages. Each message has a strictly specific type of data structure, which can be standard data structure in ROS or a customized data structure.

Topics

Messages can be published or subscribed by a node using topics. Multiple concurrent publishers and subscribers for a single topic may exist. Similarly, a node can publish and subscribe multiple topics. Each publisher and each subscriber are independent and will not affect each other when publishing or subscribing messages.

Services

Compared with publishers and subscribers, services are more suitable for synchronous transactions. A service has a request and a response, which are defined by a pair of strictly defined messages. Services are similar with functions or web services. Any node can call for a service at any time. Once the service receives the request, it will send out a response with a strict data type.

4.3 Summary

In this chapter, several software tools used in this thesis are introduced. Since the mobile manipulation problem is a nonlinear optimal problem, CasADi and a toolbox in [25] are used for simulations and experiments. In experiments, ROS is an useful tool, which will help users develop a robotic prototype easily and quickly.

Part III

Simulations and Experiments

Chapter 5

Simulations

In this chapter, simulations of dynamic system and kinematic system are presented. CasADi is used for simulations of kinematic system and the toolbox provided by [25] is used for Matlab simulations of dynamic system. The simulation of dynamic system is conducted on a desktop computer with 8 cores, 3.10GHz Intel ® Core™ i7-4770S CPU, 16GB of RAM. The simulation of kinematic system is conducted on a desktop computer with 8 cores, 3.40GHz Intel ® Core™ i7-6700 CPU, 32GB of RAM.

5.1 Dynamic System

In simulation of dynamic system, a two-agent system, which consists of two ground vehicles equipped with 3 dof planar manipulators, rigidly grasping an object together is considered. The ground vehicle, which has three degrees of freedom, can move along x-axis, y-axis and rotate around z-axis. Obviously, for each agent, the degree of freedom is $n_1 = n_2 = 6$. The total degree of freedom of the system is $n = n_1 + n_2 = 12$. Both the manipulators and ground vehicles are able to control force or torque.

For convenience, the distance between the first joint and the second joint is denoted as $lq1$. The distance between the second joint and the third joint is denoted as $lq2$ and the length of the end-effector is denoted as $l3$. In simulation, we choose $lq1 = 0.2$, $lq2 = 1$ and $l3 = 1$.

The simulations are conducted using the Matlab toolbox provided by [25].

5.1.1 Centralized MPC

According to the configuration, we have that $x = [x_o^\top, v_o^\top, q]^\top \in \mathbb{R}^{20}$, with $x_o = [p_o^\top, \phi_o]^\top \in \mathbb{R}^4$, $v_o = [\dot{p}_o^\top, \omega_{x_o}]^\top \in \mathbb{R}^4$, $p_o = [x_o, y_o, z_o]^\top \in \mathbb{R}^3$, $q = [q_1^\top, q_2^\top] \in \mathbb{R}^{12}$, $q_i = [p_{B_i}^\top, \alpha_i^\top] \in \mathbb{R}^6$, $p_{B_i} = [x_{B_i}, y_{B_i}, \psi_{B_i}]^\top \in \mathbb{R}^3$, $\alpha_i = [\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_3}]^\top \in \mathbb{R}^3$, $i \in \{1, 2\}$.

The ground vehicle can rotate around z-axis clockwise or counter-clockwise. Thus, there is a constraint for the ground vehicle:

$$-2\pi \leq \psi_{B_i} \leq 2\pi,$$

where $i \in \{1, 2\}$.

The determinant of Jacobian matrix of manipulator is

$$det(J_i) = lq1 * lq2 * sin(\alpha_{i_1} + \alpha_{i_1})cos(\alpha_{i_1}) + lq1 * lq2 * cos(\alpha_{i_1} + \alpha_{i_1})sin(\alpha_{i_1})$$

Thus, the manipulators become singular when $tan(\alpha_{i_1} + \alpha_{i_2}) = tan(\alpha_{i_1})$, i.e $\alpha_{i_2} = 0$, $i \in \{1, 2\}$. Then, the constraints for the manipulators are set to:

$$-\frac{\pi}{2} < \alpha_{1_1} < \frac{\pi}{2}, \epsilon < \alpha_{1_2} < \frac{\pi}{2}, -\frac{\pi}{2} < \alpha_{1_3} < \frac{\pi}{2}$$

$$-\frac{\pi}{2} < \alpha_{2_1} < \frac{\pi}{2}, -\frac{\pi}{2} < \alpha_{2_2} < -\epsilon, -\frac{\pi}{2} < \alpha_{2_3} < \frac{\pi}{2},$$

where ϵ is an arbitrary small value, which is larger than zero.

The input constraints are

$$-5 \leq u_i(t) \leq 5,$$

where $i \in \{1, 2, ..., 6\}$.

The initial conditions are set to:

$$\begin{aligned}
x_o &= \left[\ 0 \quad , -2.2071, \quad 0.9071, \quad \tfrac{\pi}{2}\ \right]^\top, \\
v_o &= \left[\ 0 \quad , 0, \quad 0, \quad 0\ \right]^\top, \\
q_1 &= \left[\ 0, \quad 0, \quad 0, \quad 0, \quad \tfrac{\pi}{4}, \quad \tfrac{\pi}{4}\ \right]^\top, \\
q_2 &= \left[\ 0, \quad -4.4142, \quad 0, \quad 0, \quad -\tfrac{\pi}{4}, \quad -\tfrac{\pi}{4}\ \right]^\top.
\end{aligned}$$

The desired goal states are set to:

$$
\begin{aligned}
x_o &= \left[\begin{array}{cccc} 5 & ,-2.2071, & 0.9071, & \frac{\pi}{2} \end{array}\right]^\top, \\
v_o &= \left[\begin{array}{cccc} 0 & ,0, & 0, & 0 \end{array}\right]^\top, \\
q_1 &= \left[\begin{array}{cccccc} 5, & 0, & 0, & 0, & \frac{\pi}{4}, & \frac{\pi}{4} \end{array}\right]^\top, \\
q_2 &= \left[\begin{array}{cccccc} 5, & -4.4142, & 0, & 0, & -\frac{\pi}{4}, & -\frac{\pi}{4} \end{array}\right]^\top.
\end{aligned}
$$

The obstacle is set between the initial and the desired position of the object. The obstacle is a sphere whose center is $\left[\begin{array}{ccc} 2.5 & ,-2.2071, & 1.2 \end{array}\right]$ and radius is 0.7071. The sampling time is 0.1 seconds. The horizon is 4 and the total simulation time is 40 seconds. The matrices P, Q, R are set to :

$$P = 10I_{20\times 20}, Q = 10I_{20\times 20}, R = 2I_{4\times 4}$$

Fig. 5.1a shows the error between states of agent 1 and the destination of agent 1. Similarly, Fig. 5.1b and Fig. 5.1c are the error of agent 2 and the error of object, respectively.

Fig. 5.2 shows the states of agent 1, the states of agent 2 and the states of the object.

Fig. 5.3 shows the velocity of the object using CNMPC. Fig. 5.4 shows the control input of CNMPC and Fig. 5.5 shows the obstacle function of CNMPC.

The simulation results show that the states of the agents and the object converge to the goal states and the obstacle is avoided. The simulation takes 6218.762939 seconds, while the simulation for original nonlinear system is intractable.

5.1.2 Decentralized MPC

For agent i, we have that $x_i = [q_i, \dot{q}_i]^\top \in \mathbb{R}^{12}$, with $q_i = [p_{B_i}^\top, \alpha_i^\top]^\top \in \mathbb{R}^6$, $p_{B_i} = [x_{B_i}, y_{B_i}, \psi_{B_i}]^\top \in \mathbb{R}^3$, $\alpha_i = [\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_3}]^\top \in \mathbb{R}^3$, $i \in \{1, 2\}$.

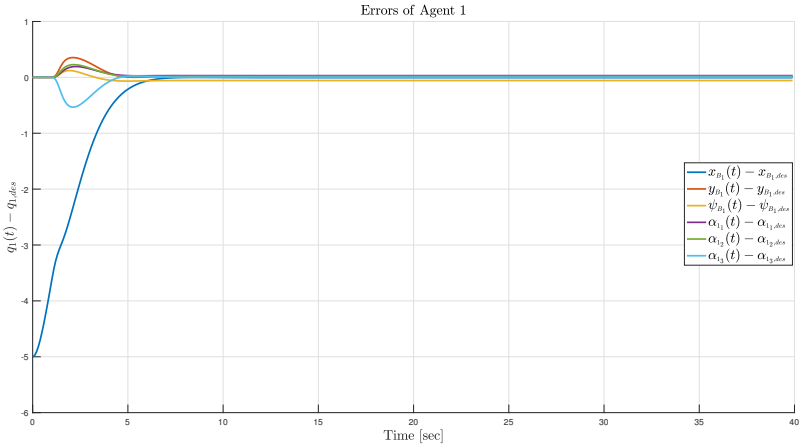
The ground vehicle can rotate around z-axis clockwise or counterclockwise. Thus, there is a constraint for the ground vehicle:

$$-2\pi \leq \psi_{B_i} \leq 2\pi,$$

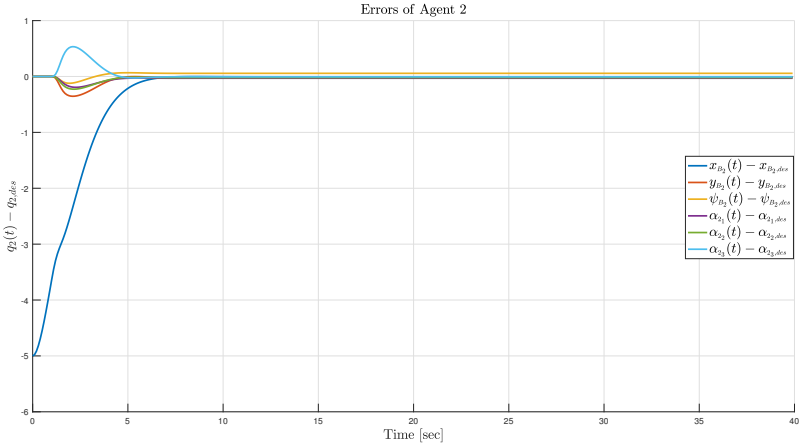
where $i \in \{1, 2\}$.

The determinant of Jacobian matrix of manipulator is

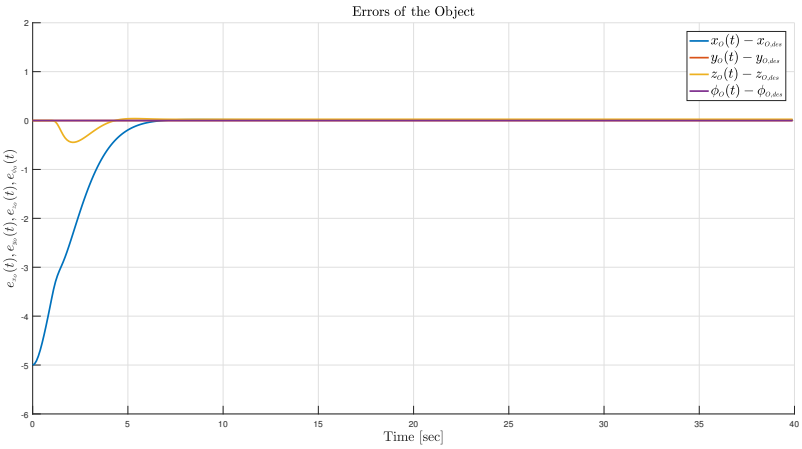
$$det(J_i) = lq1 * lq2 * sin(\alpha_{i_1} + \alpha_{i_1})cos(\alpha_{i_1}) + lq1 * lq2 * cos(\alpha_{i_1} + \alpha_{i_1})sin(\alpha_{i_1})$$



(a) Error of agent 1 using CNMPC

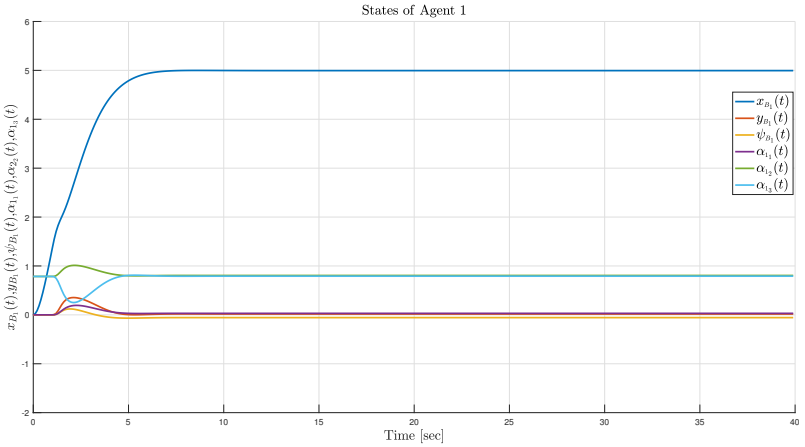


(b) Error of agent 2 using CNMPC

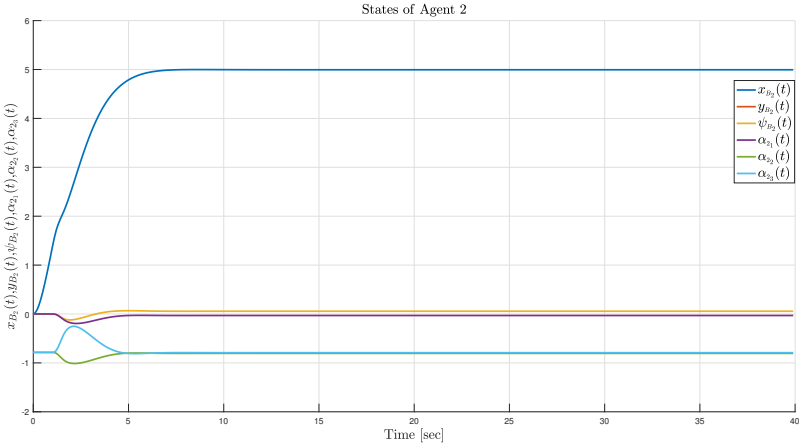


(c) Error of object using CNMPC

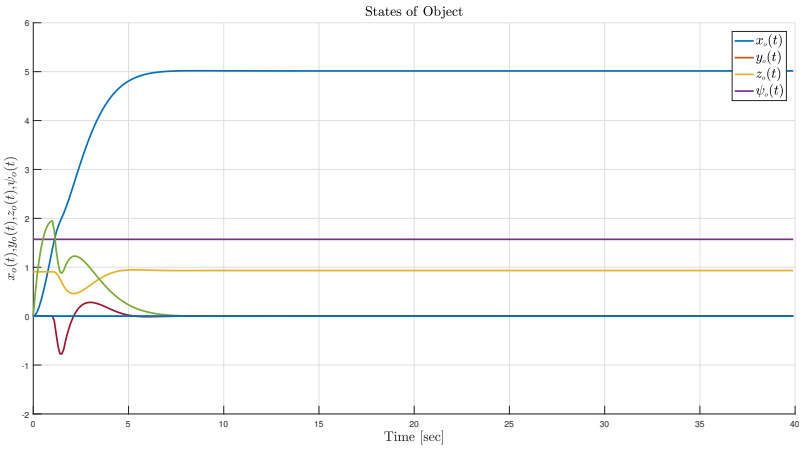
Figure 5.1: Errors using CNMPC for dynamic system in simulations



(a) States of agent 1 using CNMPC



(b) States of agent 2 using CNMPC



(c) States of object using CNMPC

Figure 5.2: States using CNMPC for dynamic system in simulations

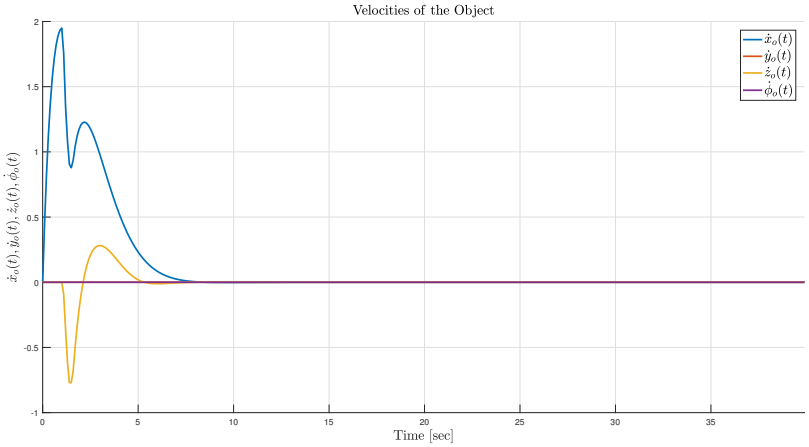


Figure 5.3: Velocity of object using CNMPC for dynamic system in simulations

Thus, the manipulators become singular when $tan(\alpha_{i_1} + \alpha_{i_2}) = tan(\alpha_{i_1})$, i.e $\alpha_{i_2} = 0$, $i \in \{1, 2\}$. Then, the constraints for the manipulator 1 are set to:

$$-\frac{\pi}{2} < \alpha_{1_1} < \frac{\pi}{2}, \epsilon < \alpha_{1_2} < \frac{\pi}{2}, -\frac{\pi}{2} < \alpha_{1_3} < \frac{\pi}{2},$$

where ϵ is an arbitrary small value, which is larger than zero.

The input constraint for agent 1 is

$$-5 \leq u_i(t) \leq 5,$$

where $i \in \{1, \ldots, 6\}$.

For manipulator 2, the state constraints are

$$-\epsilon < p_{o_1}(t) - p_{o_2}(t) < \epsilon,$$
$$-\epsilon < v_{o_1}(t) - v_{o_2}(t) < \epsilon,$$

where ϵ is an arbitrary small value, which is larger than zero, p_{o_i} is the position of the object calculated by agent i, $i \in \{1, 2\}$ and v_{o_i} is the velocity of the object calculated by agent i, $i \in \{1, 2\}$.

The input constraint for agent 2 is

$$-5 \leq u_j(t) \leq 5,$$

where $j \in \{1, \ldots, 6\}$.

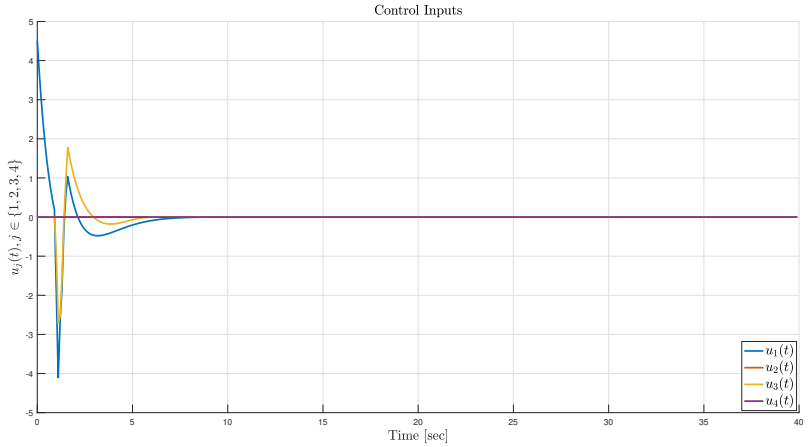


Figure 5.4: Control inputs using CNMPC for dynamic system in simulations

The initial conditions are set to:

$$
\begin{aligned}
q_1 &= \begin{bmatrix} 0, & 0, & 0, & 0, & \frac{\pi}{4}, & \frac{\pi}{4} \end{bmatrix}^\top, \\
\dot{q_1} &= \begin{bmatrix} 0, & 0, & 0, & 0, & 0, & 0 \end{bmatrix}^\top, \\
q_2 &= \begin{bmatrix} 0, & -4.4142, & 0, & 0, & -\frac{\pi}{4}, & -\frac{\pi}{4} \end{bmatrix}^\top, \\
\dot{q_2} &= \begin{bmatrix} 0, & 0, & 0, & 0, & 0, & 0 \end{bmatrix}^\top.
\end{aligned}
$$

The desired goal states are set to:

$$
\begin{aligned}
q_1 &= \begin{bmatrix} 5, & 0, & 0, & 0, & \frac{\pi}{4}, & \frac{\pi}{4} \end{bmatrix}^\top, \\
\dot{q_1} &= \begin{bmatrix} 0, & 0, & 0, & 0, & 0, & 0 \end{bmatrix}^\top, \\
q_2 &= \begin{bmatrix} 5, & -4.4142, & 0, & 0, & -\frac{\pi}{4}, & -\frac{\pi}{4} \end{bmatrix}^\top, \\
\dot{q_2} &= \begin{bmatrix} 0, & 0, & 0, & 0, & 0, & 0 \end{bmatrix}^\top.
\end{aligned}
$$

The obstacle is set between the initial and the desired position of the object. The obstacle is a sphere whose center is $\begin{bmatrix} 2.5 & , -2.2071, & 1.2 \end{bmatrix}$ and radius is 0.7071. The sampling time is 0.1 seconds. The horizon is 5 and the total simulation time is 40 seconds.The matrices P_1, Q_1, R_1, P_2, Q_2, R_2 are set to :

$$
\begin{aligned}
P_1 = 0.5I_{12\times12}, Q_1 = 0.5I_{12\times12}, R_1 = 0.5I_{6\times6}, \\
P_2 = 0.5I_{12\times12}, Q_2 = 0.5I_{12\times12}, R_2 = 0.5I_{6\times6}.
\end{aligned}
$$

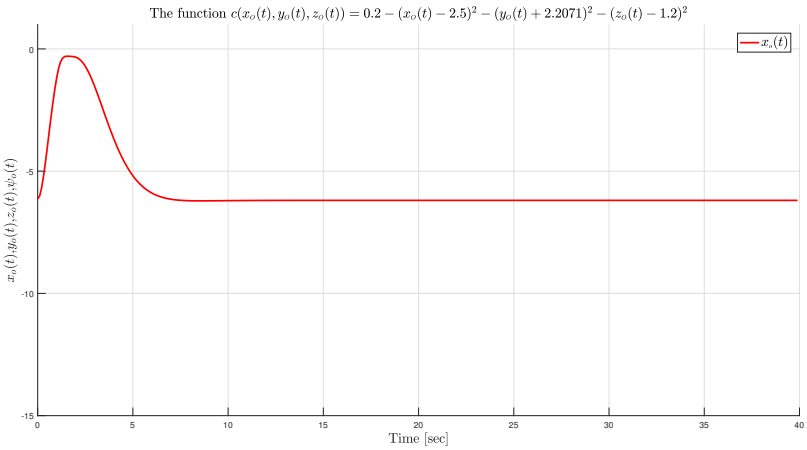


Figure 5.5: Obstacle function using CNMPC for dynamic system in simulations

Fig. 5.6a shows the error between states of agent 1 and the destination of agent 1. Similarly, Fig. 5.6b is the error of agent 2.

Fig. 5.7a and Fig. 5.7b show the states of agent 1, the states of agent 2, respectively. Fig. 5.8a and Fig. 5.8b show the states of the object calculated by the states of agent 1 and the states of agent 2, respectively.

Fig. 5.9a and Fig. 5.9b show the velocities of the agent 1 and the velocities of the agent 2 using DNMPC.

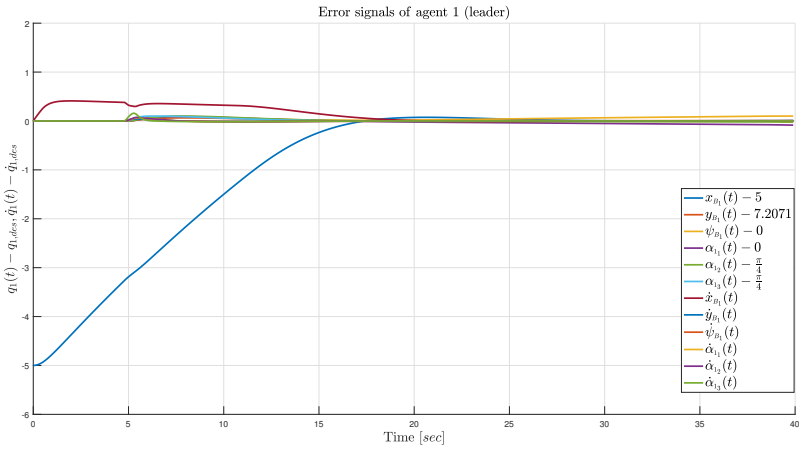
The control input of agent 1 and the control input of agent 2 using DNMPC are depicted in Fig. 5.10a and Fig. 5.10b, respectively.

The obstacle function using DNMPC is shown in Fig. 5.11.

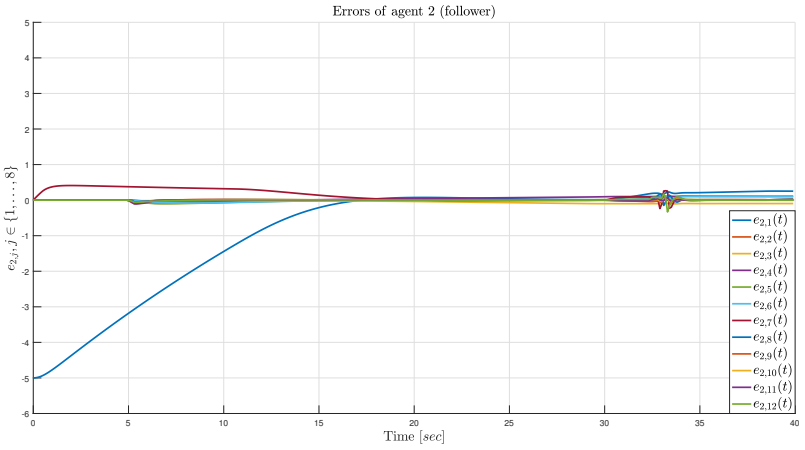
The simulation results show that the states of the agents and the object converge to the goal states and the obstacle is avoided. The simulation takes 15975.379603 seconds, while the simulation for original nonlinear system is intractable.

5.1.3 Summary

The simulation results show that for both CNMPC scheme and DNMPC scheme the states of the agents and object are able to converge to the destination with singularity avoidance and obstacle avoidance. But the simulations of linearized dynamic system for both CNMPC and DNMPC are very slow in Matlab, which is not suitable for experiments.

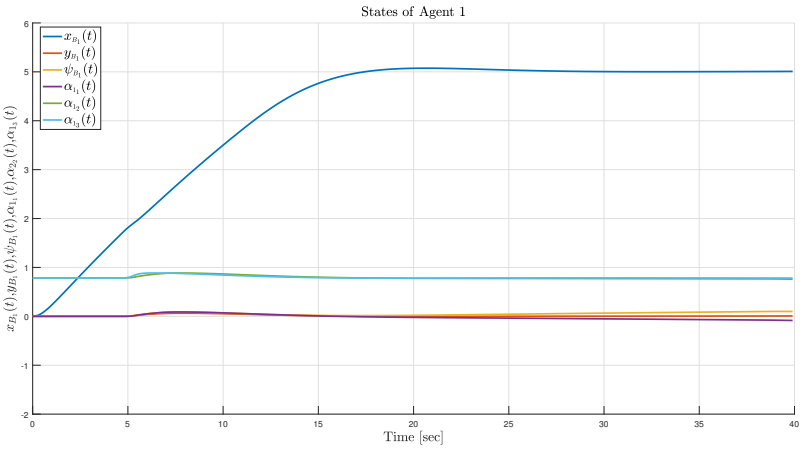


(a) Error of agent 1 using DNMPC

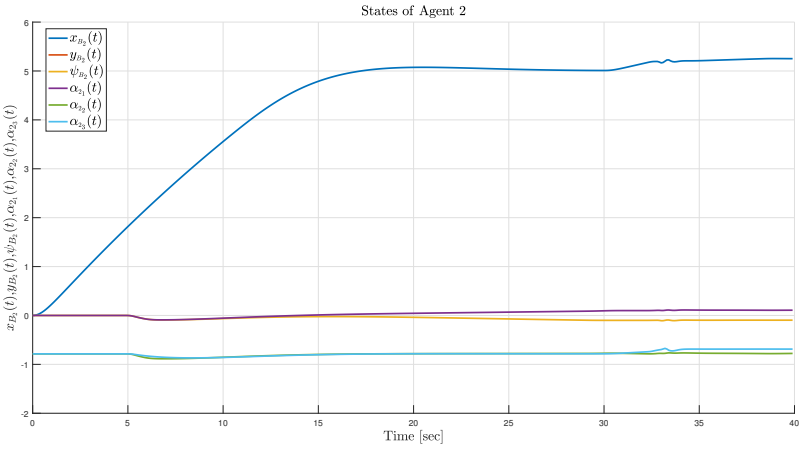


(b) Error of agent 2 using DNMPC

Figure 5.6: Errors using DNMPC for dynamic system in simulations

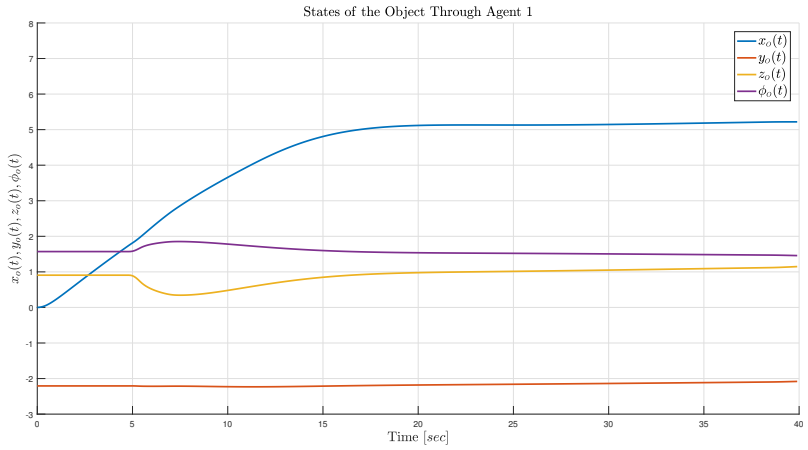


(a) States of agent 1 using DNMPC

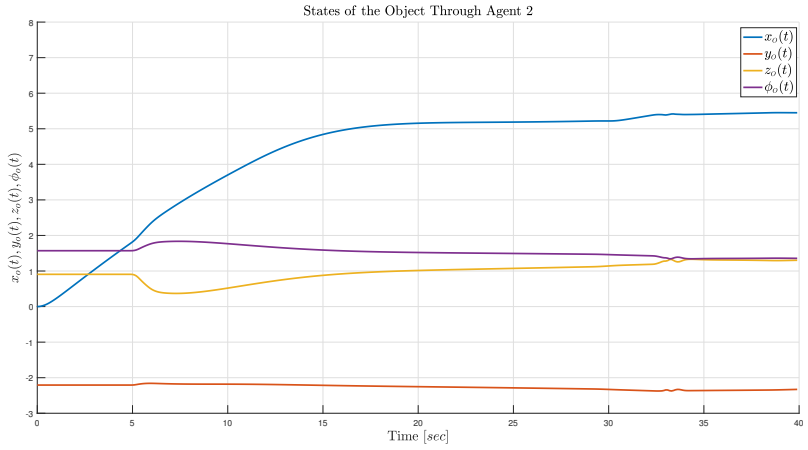


(b) States of agent 2 using DNMPC

Figure 5.7: States of agents using DNMPC for Dynamic System in simulations

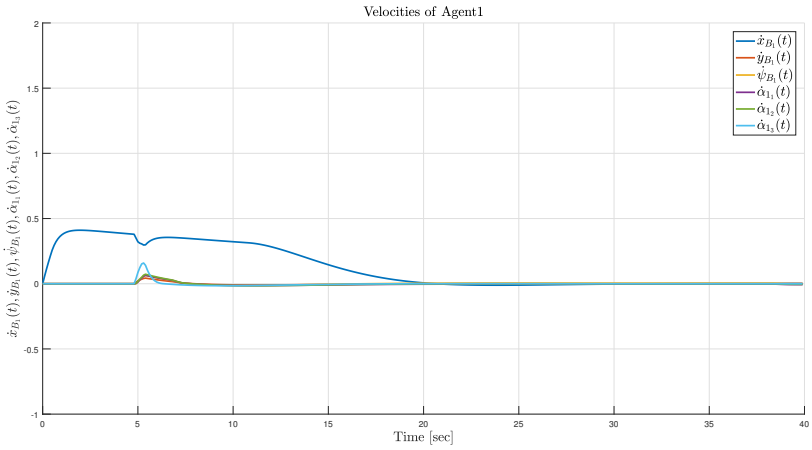


(a) States of object calculated by agent 1 using DNMPC

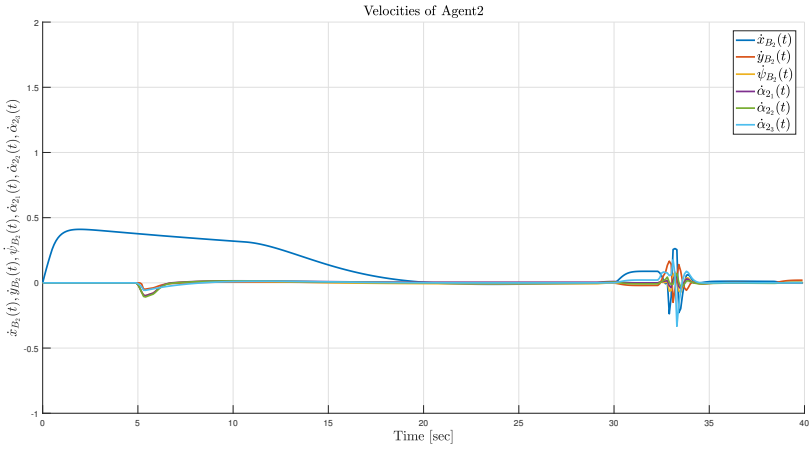


(b) States of object calculated by agent 2 using DNMPC

Figure 5.8: States of object using DNMPC for dynamic system in simulations

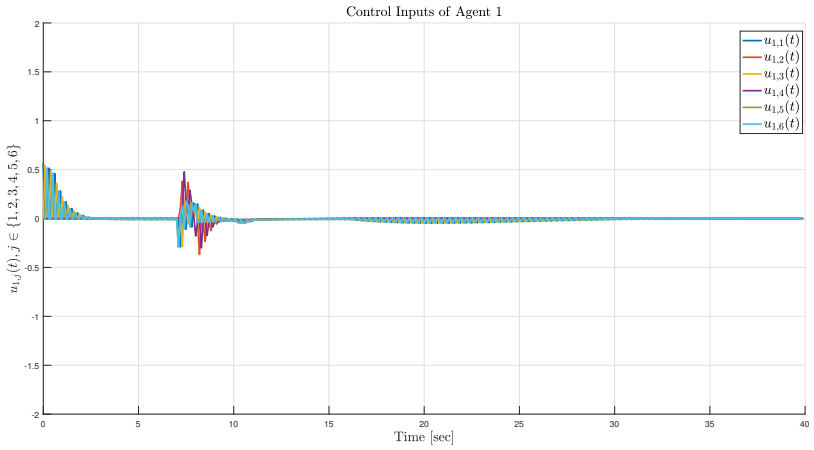


(a) Velocities of agent 1 using DNMPC

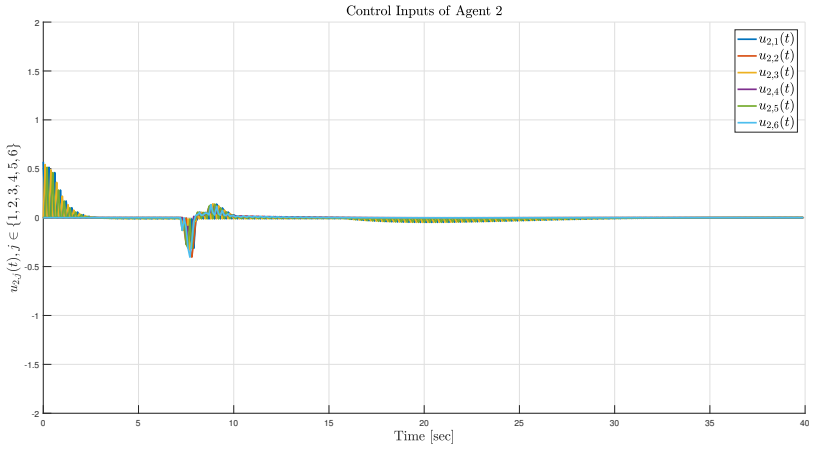


(b) Velocities of agent 2 using DNMPC

Figure 5.9: Velocities of agents using DNMPC for dynamic system in simulations



(a) Control inputs of agent 1 using DNMPC



(b) Control inputs of agent 2 using DNMPC

Figure 5.10: Control inputs using DNMPC for dynamic system in simulations

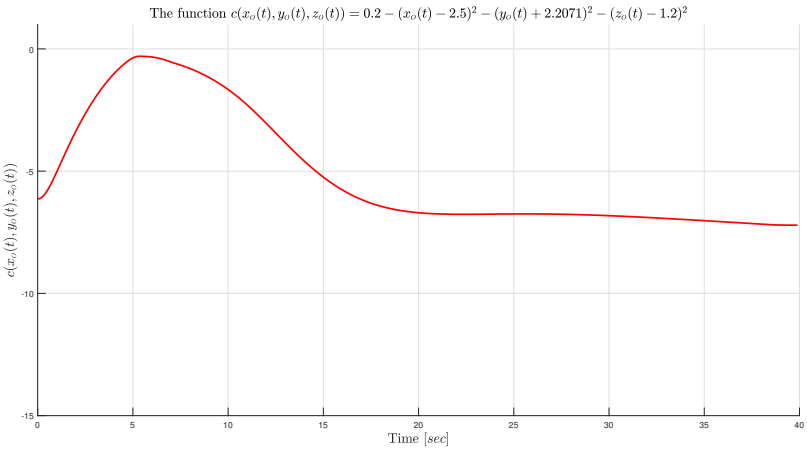


Figure 5.11: Obstacle function using DNMPC for dynamic system in simulations

5.2 Kinematic System

In simulation of kinematic system, a two-agent system, which consists of two ground vehicles equipped with 3 dof planar manipulators, rigidly grasping an object together is considered. The ground vehicle, which has three degrees of freedom, can move along x-axis, y-axis and rotate around z-axis. Obviously, for each agent, the degree of freedom is $n_1 = n_2 = 6$. The total degree of freedom of the system is $n = n_1 + n_2 = 12$. Both the manipulators and ground vehicles are only able to control linear velocity or angular velocity.

For convenience, the distance between the first joint and the second joint is denoted as $lq1$. The distance between the second joint and the third joint is denoted as $lq2$ and the length of the end-effector is denoted as $l3$. In simulation, we choose $lq1 = 15.13cm$, $lq2 = 14.32cm$ and $l3 = 11.5cm$.

The simulations are conducted using CasADi in Python.

5.2.1 Centralized MPC

According to the configuration, we have that $x = [x_o^\top, q]^\top \in \mathbb{R}^{16}$, with $x_o = [p_o^\top, \phi_o]^\top \in \mathbb{R}^4$, $p_o = [x_o, y_o, z_o]^\top \in \mathbb{R}^3$, $q = [q_1^\top, q_2^\top] \in \mathbb{R}^{12}$, $q_i = [p_{B_i}^\top, \alpha_i^\top] \in \mathbb{R}^6$, $p_{B_i} = [x_{B_i}, y_{B_i}, \psi_{B_i}]^\top \in \mathbb{R}^3$, $\alpha_i = [\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_3}]^\top \in \mathbb{R}^3$, $i \in \{1, 2\}$.

The ground vehicle can rotate around z-axis clockwise or counter-clockwise. Thus, there is a constraint for the ground vehicle:

$$-2\pi \leq \psi_{B_i} \leq 2\pi,$$

where $i \in \{1, 2\}$.

The determinant of Jacobian matrix of manipulator is

$$det(J_i) = lq1 * lq2 * sin(\alpha_{i_1} + \alpha_{i_1})cos(\alpha_{i_1}) + lq1 * lq2 * cos(\alpha_{i_1} + \alpha_{i_1})sin(\alpha_{i_1})$$

Thus, the manipulators become singular when $tan(\alpha_{i_1} + \alpha_{i_2}) = tan(\alpha_{i_1})$, i.e $\alpha_{i_2} = 0$, $i \in \{1, 2\}$. Then, the constraints for the manipulators are set to:

$$-\frac{\pi}{2} < \alpha_{1_1} < \frac{\pi}{2}, \epsilon < \alpha_{1_2} < \frac{\pi}{2}, -\frac{\pi}{2} < \alpha_{1_3} < \frac{\pi}{2},$$
$$-\frac{\pi}{2} < \alpha_{2_1} < \frac{\pi}{2}, -\frac{\pi}{2} < \alpha_{2_2} < -\epsilon, -\frac{\pi}{2} < \alpha_{2_3} < \frac{\pi}{2},$$

where ϵ is an arbitrary small value, which is larger than zero.

The inputs of the system are the velocities of the object. Assume u_1 denotes the linear velocity of the object on x-axis; u_2 denotes the linear velocity of the object on y-axis; u_3 denotes the linear velocity of the object on z-axis and u_4 denotes the angular velocity of the object around x-axis. Then the input constraints are set to:

$$-2dm/s \leq u_1(t) \leq 2dm/s;$$
$$-0.4dm/s \leq u_2(t) \leq 0.4dm/s;$$
$$-0.4dm/s \leq u_3(t) \leq 0.4dm/s;$$
$$-0.02rad/s \leq u_4(t) \leq 0.02rad/s.$$

The initial conditions are set to:

$$x_o = \begin{bmatrix} 0 & -2.8126 & 2.5256 & \frac{\pi}{2} \end{bmatrix}^\top,$$
$$q_1 = \begin{bmatrix} 0, & 0, & 0, & 0, & \frac{\pi}{4}, & \frac{\pi}{4} \end{bmatrix}^\top,$$
$$q_2 = \begin{bmatrix} 0, & -5.6252, & 0, & 0, & -\frac{\pi}{4}, & -\frac{\pi}{4} \end{bmatrix}^\top.$$

The desired goal states are set to:

$$x_o = \begin{bmatrix} 10, & -2.8126, & 2.5256, & \frac{\pi}{2} \end{bmatrix}^\top,$$
$$q_1 = \begin{bmatrix} 10, & 0, & 0, & 0, & \frac{\pi}{4}, & \frac{\pi}{4} \end{bmatrix}^\top,$$
$$q_2 = \begin{bmatrix} 10, & -5.6252, & 0, & 0, & -\frac{\pi}{4}, & -\frac{\pi}{4} \end{bmatrix}^\top.$$

The obstacle is set between the initial and the desired position of the object. The obstacle is a sphere whose center is $\begin{bmatrix} 3 & , -2.8126, & 3.5 \end{bmatrix}$ and radius is 0.7071. The sampling time is 0.1 seconds. The horizon is 3 and the total simulation time is 20 seconds. The matrices P, Q, R are set to :

$$P = diag\{1, 1, 1, 1, 1, 1, 1, 20, 20, 20, 1, 1, 1, 20, 20, 20\},$$
$$Q = diag\{1, 1, 1, 1, 1, 1, 1, 20, 20, 20, 1, 1, 1, 20, 20, 20\},$$
$$R = diag\{1, 1, 2, 2\}.$$

Fig. 5.12a and Fig. 5.13a show the error between states of agent 1 and the destination of agent 1. Similarly, Fig. 5.12b and Fig. 5.13b are the error of agent 2 while Fig. 5.12c and Fig. 5.13c are the error of object.

Fig. 5.14 and Fig. 5.15 shows the states of agent 1, the states of agent 2 and the states of the object.

Fig. 5.16 shows the control input of CNMPC and Fig. 5.17 shows the obstacle function of CNMPC.

The simulation results show that the states of the agents and the object converge to the goal states and the obstacle is avoided. The simulation takes 10.98 seconds. Each iteration takes 0.0549697446823 seconds.

5.2.2 Decentralized MPC

For agent i, we have that $x_i = [q_i]^\top \in \mathbb{R}^6$, with $q_i = [p_{B_i}^\top, \alpha_i^\top]^\top \in \mathbb{R}^6$, $p_{B_i} = [x_{B_i}, y_{B_i}, \psi_{B_i}]^\top \in \mathbb{R}^3$, $\alpha_i = [\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_3}]^\top \in \mathbb{R}^3$, $i \in \{1, 2\}$.

The ground vehicle can rotate around z-axis clockwise or counter-clockwise. Thus, there is a constraint for the ground vehicle:

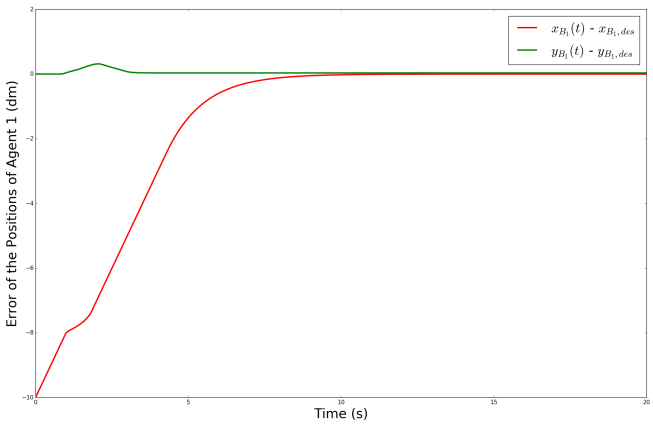
$$-2\pi \leq \psi_{B_i} \leq 2\pi,$$

where $i \in \{1, 2\}$.

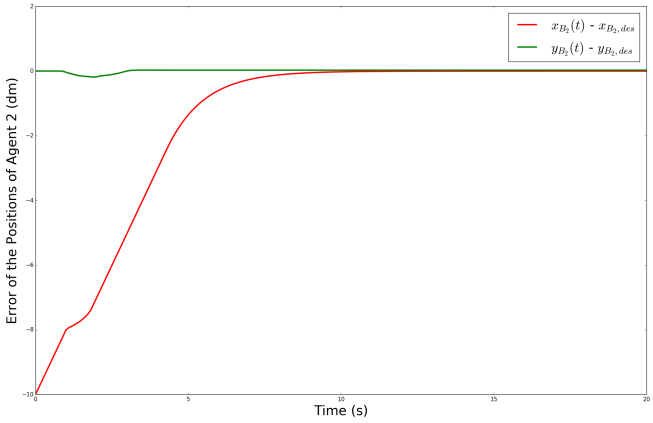
The determinant of Jacobian matrix of manipulator is

$$det(J_i) = lq1 * lq2 * sin(\alpha_{i_1} + \alpha_{i_1})cos(\alpha_{i_1}) + lq1 * lq2 * cos(\alpha_{i_1} + \alpha_{i_1})sin(\alpha_{i_1})$$

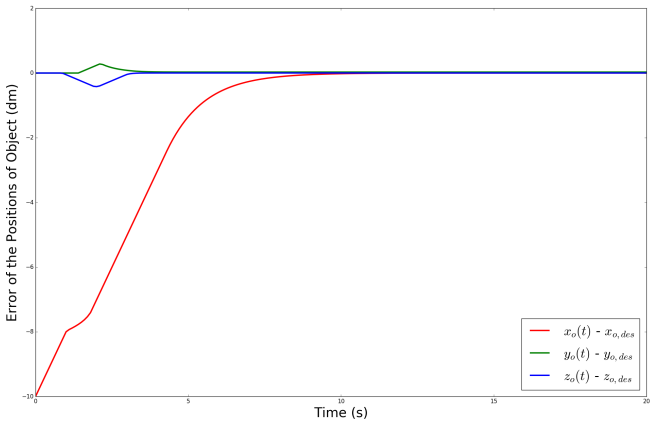
Thus, the manipulators become singular when $tan(\alpha_{i_1} + \alpha_{i_2}) = tan(\alpha_{i_1})$, i.e $\alpha_{i_2} = 0$, $i \in \{1, 2\}$. Then, the constraints for the manipulator 1 are set to:



(a) Error of agent 1's positions using CNMPC

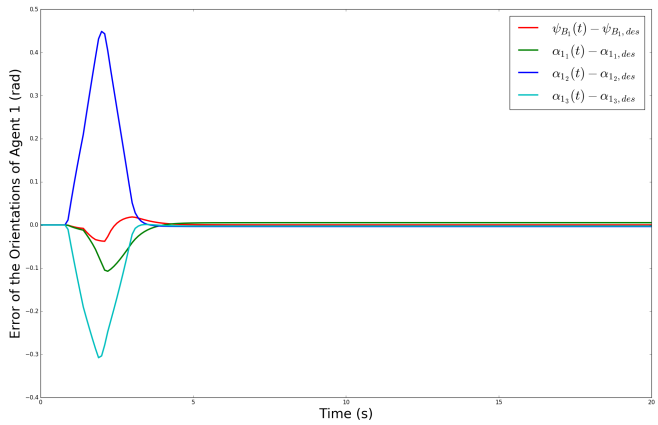


(b) Error of agent 2's positions using CNMPC

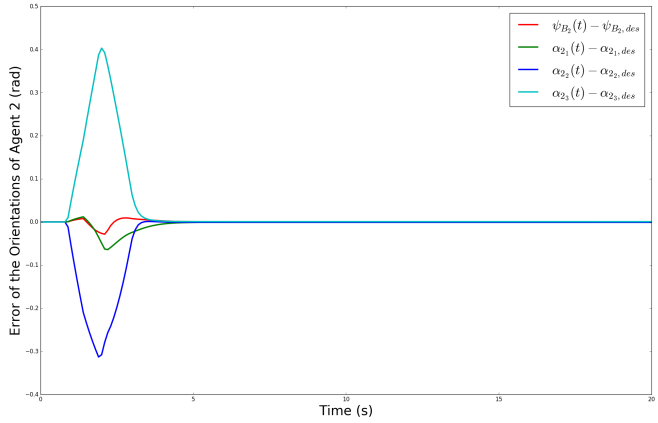


(c) Error of object's position using CNMPC

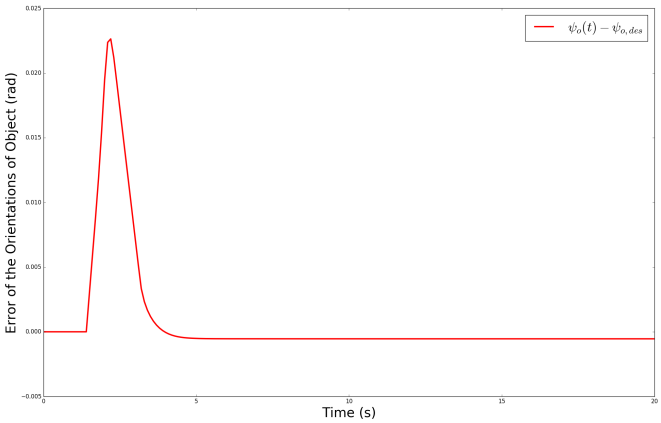
Figure 5.12: Errors of positions using CNMPC for kinematic system in simulations



(a) Error of agent 1's orientations using CNMPC

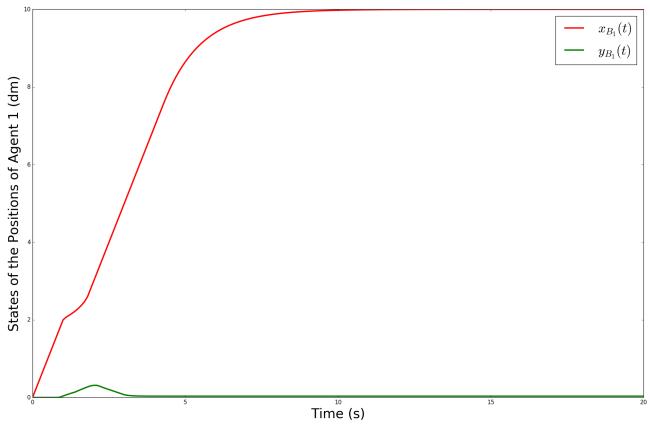


(b) Error of agent 2's orientations using CNMPC

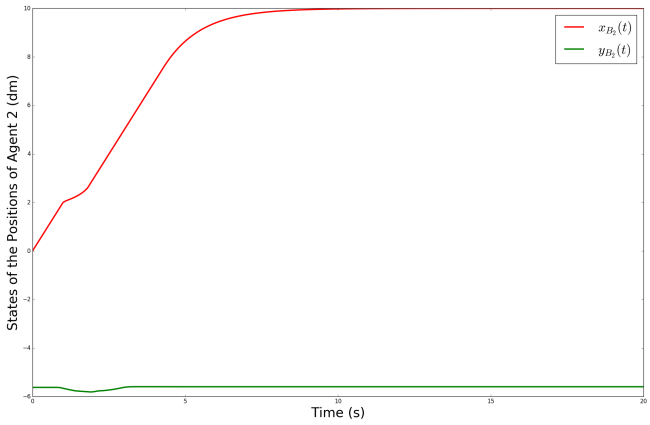


(c) Error of object's orientations using CNMPC

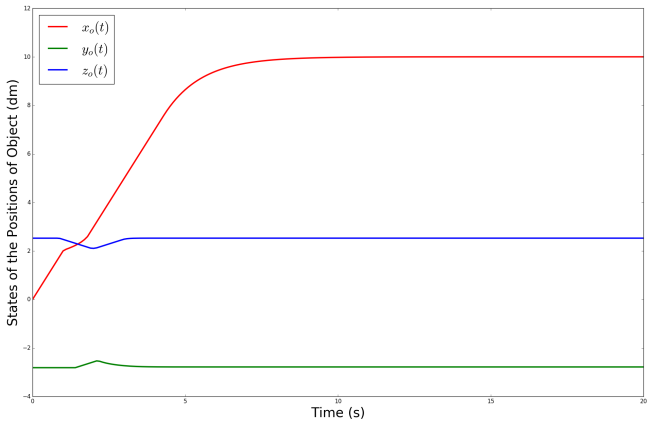
Figure 5.13: Errors of orientations using CNMPC for kinematic system in simulations



(a) States of agent 1's positions using CNMPC

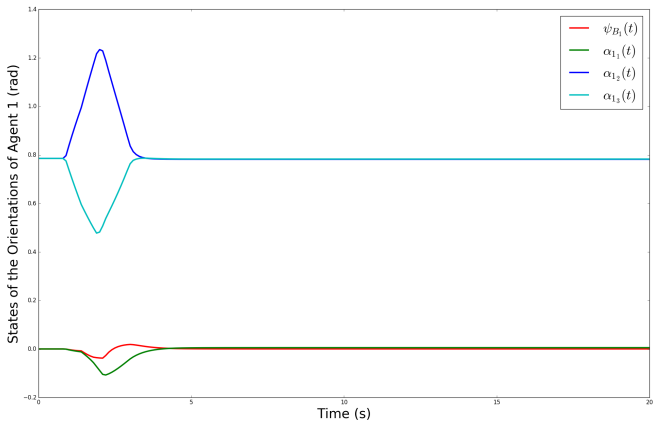


(b) States of agent 2's positions using CNMPC

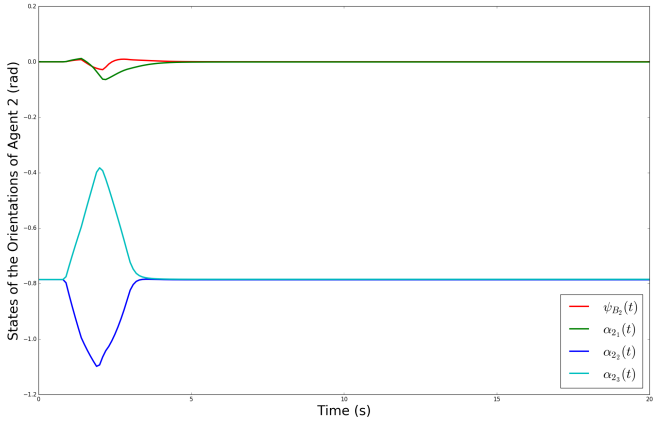


(c) States of object's positions using CNMPC

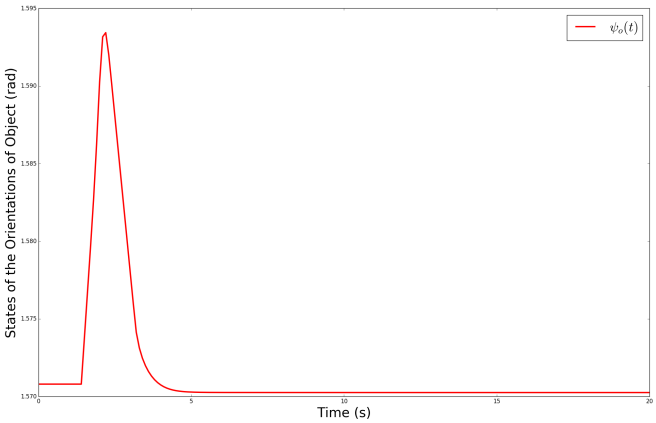
Figure 5.14: States of positions using CNMPC for kinematic system in simulations



(a) States of agent 1's orientations using CNMPC

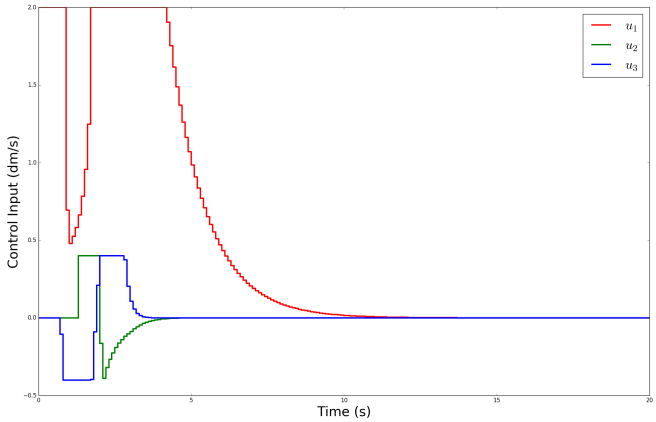


(b) States of agent 2's orientations using CNMPC

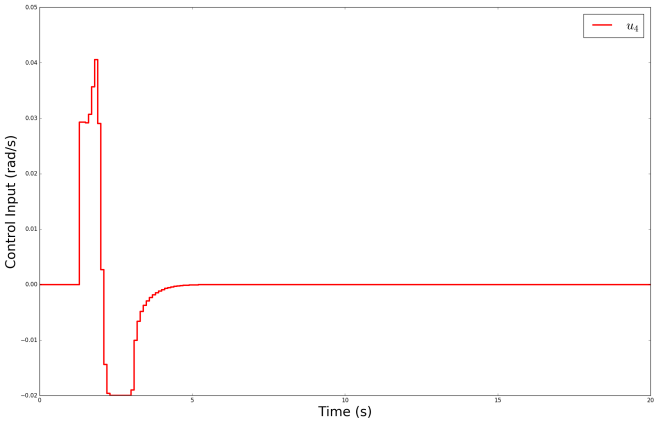


(c) States of object's orientations using CNMPC

Figure 5.15: States of orientations using CNMPC for kinematic system in simulations



(a) Control inputs of positions using CNMPC



(b) Control inputs of orientations using CNMPC

Figure 5.16: Control inputs using CNMPC for kinematic system in simulations

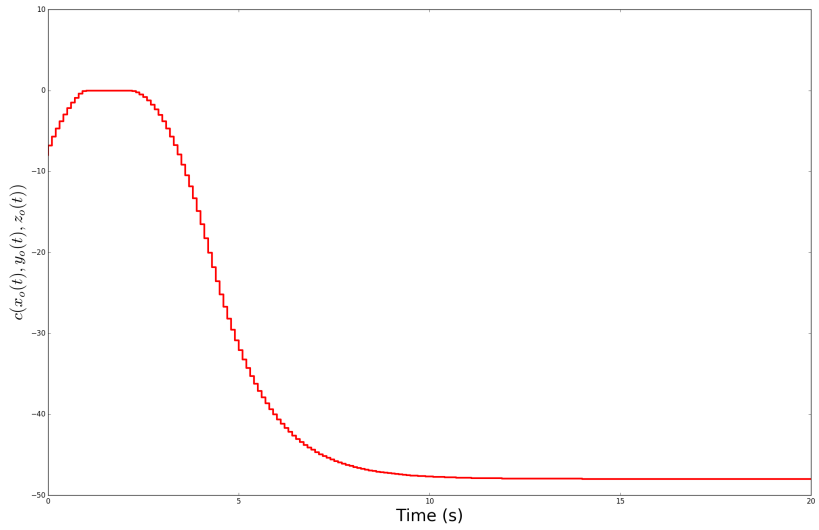


Figure 5.17: Obstacle function using CNMPC for kinematic system in simulations

$$-\frac{\pi}{2} < \alpha_{1_1} < \frac{\pi}{2}, \epsilon < \alpha_{1_2} < \frac{\pi}{2}, -\frac{\pi}{2} < \alpha_{1_3} < \frac{\pi}{2},$$

where ϵ is an arbitrary small value, which is larger than zero.

The inputs of agent i are denoted as $\{u_{i_1}, ..., u_{i_6}\}$, where $u_{i_1} = \dot{x}_{B_i}$, $u_{i_2} = \dot{y}_{B_i}$, $u_{i_3} = \dot{\psi}_{B_i}$, $u_{i_4} = \dot{\alpha}_{i_1}$, $u_{i_5} = \dot{\alpha}_{i_2}$, and $u_{i_6} = \dot{\alpha}_{i_3} \forall i \in \{1, 2\}$. The input constraints are:

$$\begin{aligned}
-2dm/s \leq u_{i_1}(t) \leq 2dm/s \\
-2dm/s \leq u_{i_2}(t) \leq 2dm/s \\
-0.7rad/s \leq u_{i_3}(t) \leq 0.7rad/s \\
-0.7rad/s \leq u_{i_4}(t) \leq 0.7rad/s \\
-0.7rad/s \leq u_{i_5}(t) \leq 0.7rad/s \\
-0.7rad/s \leq u_{i_6}(t) \leq 0.7rad/s,
\end{aligned}$$

where $i \in \{1, 2\}$.

For manipulator 2, the state constraints are

$$-\epsilon < p_{o_1}(t) - p_{o_2}(t) < \epsilon$$

,where ϵ is an arbitrary small value, which is larger than zero, p_{o_i} is the pose of the object calculated by agent i, $i \in \{1, 2\}$.

The initial conditions are set to:

$$q_1 = \begin{bmatrix} 0, & 0, & 0, & 0, & \frac{\pi}{4}, & \frac{\pi}{4} \end{bmatrix}^\top,$$
$$q_2 = \begin{bmatrix} 0, & -5.6252, & 0, & 0, & -\frac{\pi}{4}, & -\frac{\pi}{4} \end{bmatrix}^\top.$$

The desired goal states are set to:

$$q_1 = \begin{bmatrix} 10, & 0, & 0, & 0, & \frac{\pi}{4}, & \frac{\pi}{4} \end{bmatrix}^\top,$$
$$q_2 = \begin{bmatrix} 10, & -5.6252, & 0, & 0, & -\frac{\pi}{4}, & -\frac{\pi}{4} \end{bmatrix}^\top.$$

The obstacle is set between the initial and the desired position of the object. The obstacle is a sphere whose center is $\begin{bmatrix} 3 & , -2.8126, & 3.5 \end{bmatrix}$ and radius is 0.7071. The sampling time is 0.1 seconds. The horizon is 3 and the total simulation time is 20 seconds.The matrices P_1, Q_1, R_1, P_2, Q_2, R_2 are set to :

$$P_1 = I_{6\times6}, Q_1 = I_{6\times6}, R_1 = diag\{1, 1, 1, 10, 10, 10\},$$
$$P_2 = I_{6\times6}, Q_2 = I_{6\times6}, R_2 = diag\{1, 1, 1, 10, 10, 10\}.$$

Fig. 5.18 shows the error between states of positions and the destination of positions of two agent. Similarly, Fig. 5.19 is the error of agent 2.

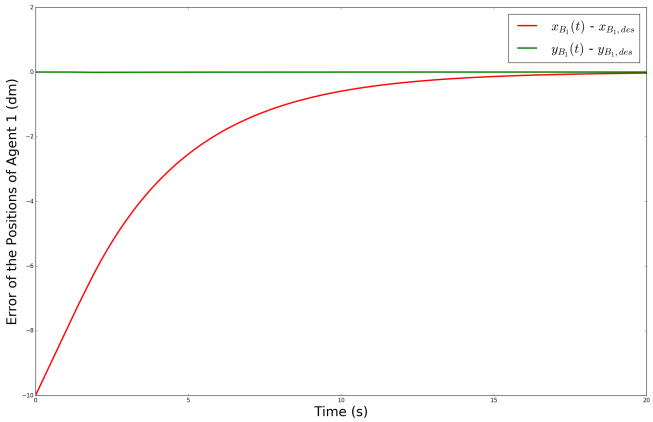
Fig. 5.20a and Fig. 5.20b show the states of agent 1's positions, the states of agent 2's positions, respectively. Fig. 5.21a and Fig. 5.21b show the states of agent 1's orientations, the states of agent 2's orientations, respectively. Fig. 5.22a and Fig. 5.22b show the states of the object calculated by the states of agent 1 and the states of agent 2, respectively.

The control input of agent 1 and the control input of agent 2 using DNMPC are depicted in Fig. 5.23 and Fig. 5.24, respectively.

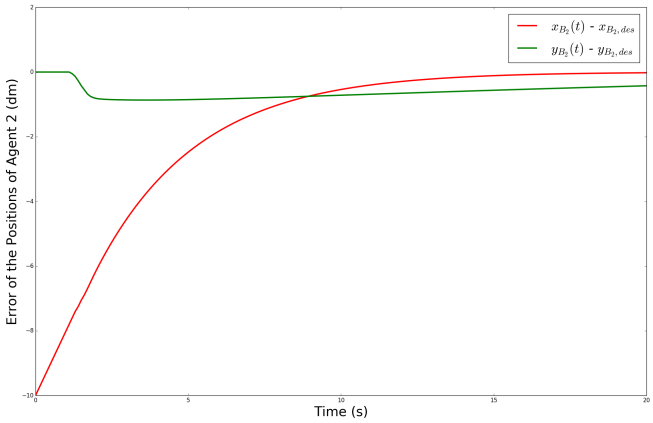
The obstacle function using DNMPC is shown in Fig. 5.25.

The simulation results show that the states of the agents and the object converge to the goal states and the obstacle is avoided.

The simulation takes 4.098 seconds. Each iteration takes 0.0204915177822 seconds.

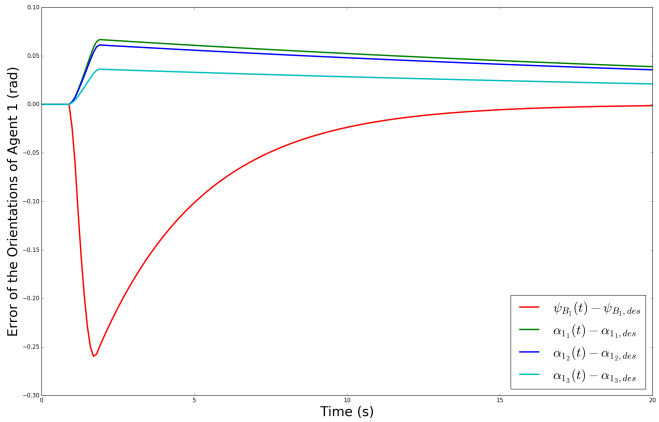


(a) Error of agent 1's positions using DNMPC

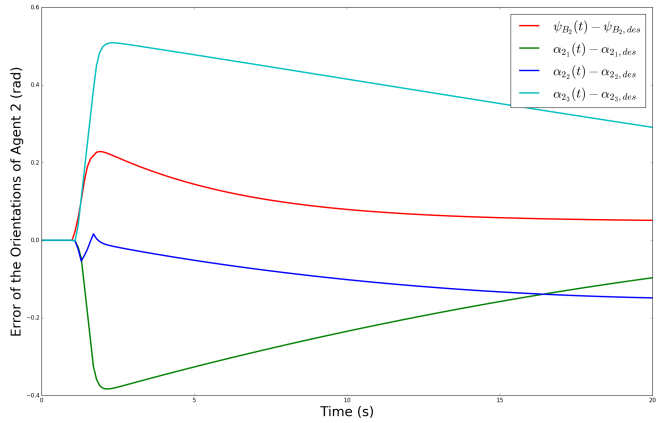


(b) Error of agent 2's positions using DNMPC

Figure 5.18: Errors of positions using DNMPC for kinematic system in simulations

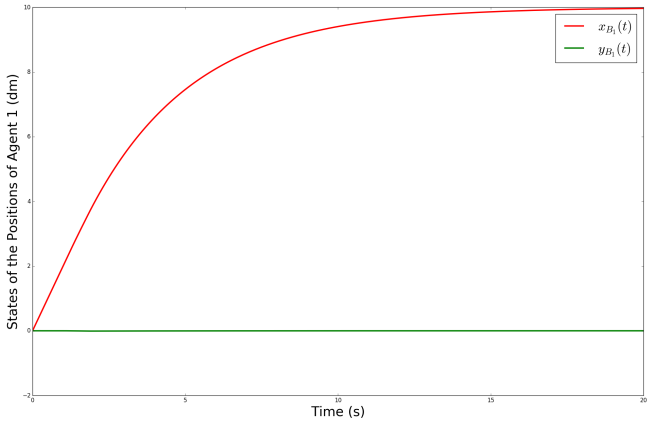


(a) Error of agent 1's orientations using DNMPC

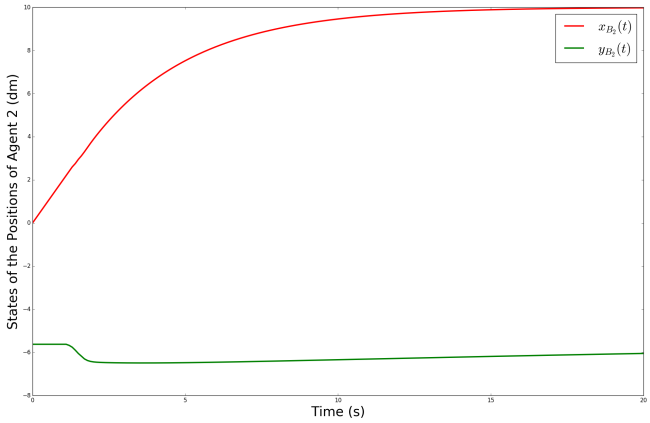


(b) Error of agent 2's orientations using DNMPC

Figure 5.19: Errors of orientations using DNMPC for kinematic system in simulations

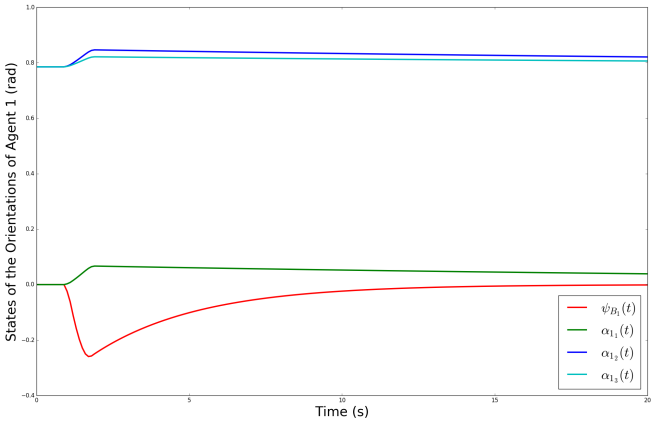


(a) States of agent 1's positions using DNMPC

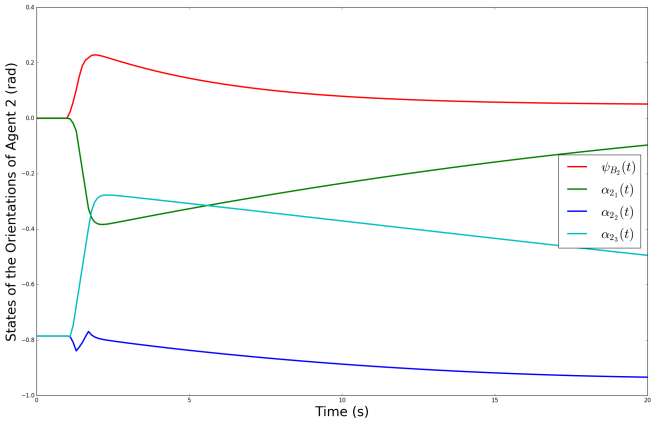


(b) States of agent 2's positions using DNMPC

Figure 5.20: States of agents' positions using DNMPC for kinematic system in simulations

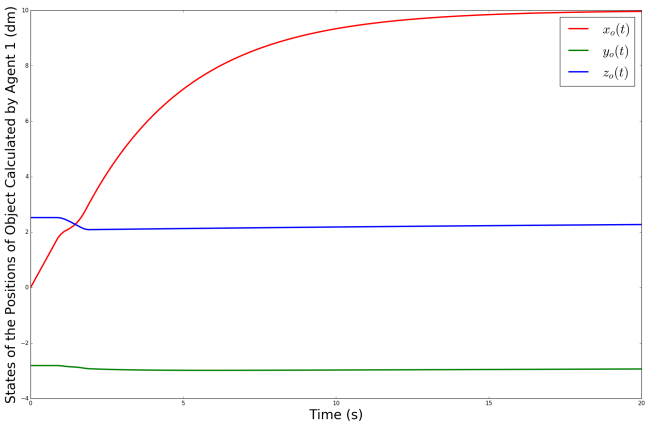


(a) States of agent 1's orientations using DNMPC

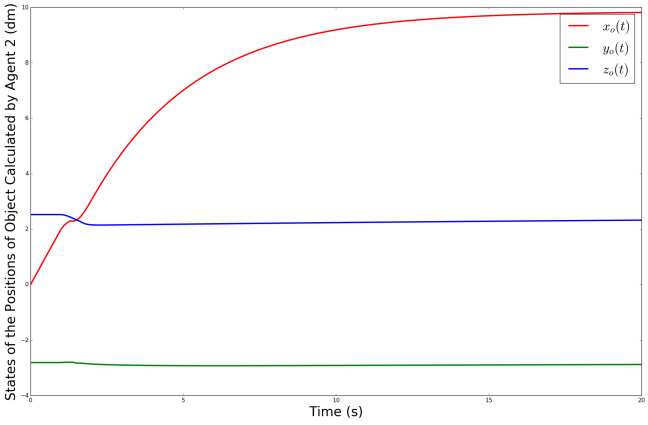


(b) States of agent 2's orientations using DNMPC

Figure 5.21: States of agents' orientations using DNMPC for kinematic system in simulations

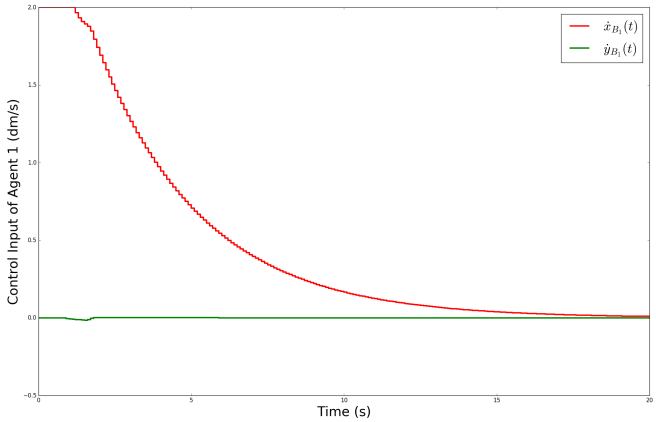


(a) States of object's positions calculated by agent 1 using DNMPC

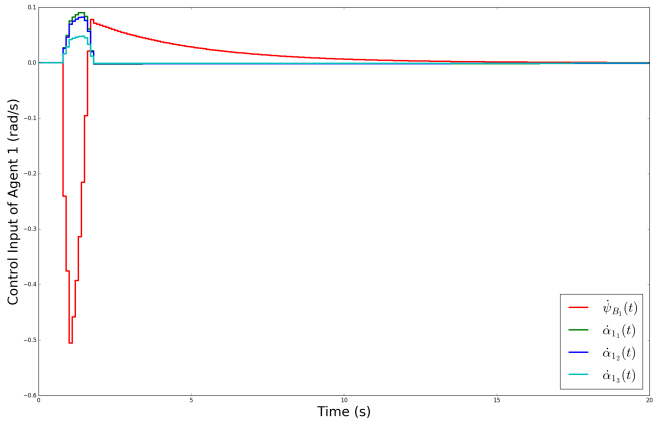


(b) States of object's positions calculated by agent 2 using DNMPC

Figure 5.22: States of object using DNMPC for kinematic system in simulations

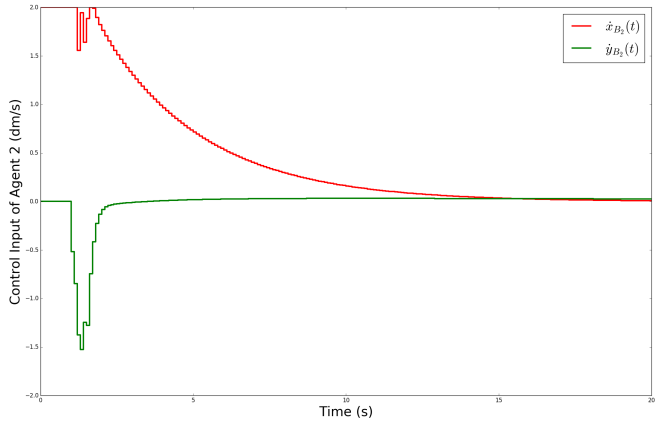


(a) Control inputs of agent 1's positions using DNMPC

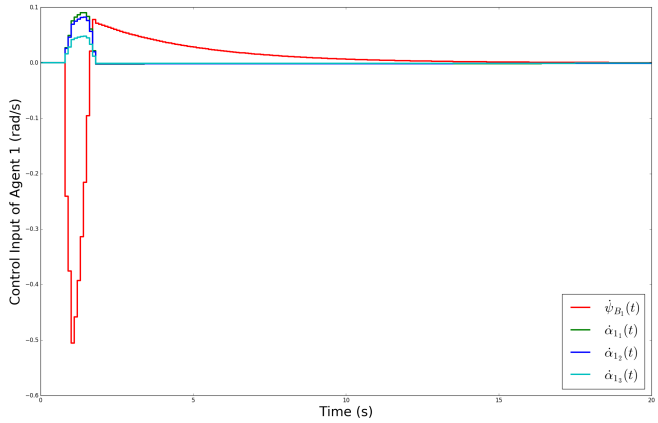


(b) Control inputs of agent 1's orientations using DNMPC

Figure 5.23: Control inputs of agent 1 using DNMPC for dynamic system in simulations



(a) Control input of agent 2's positions using DNMPC



(b) Control input of agent 2's orientations using DNMPC

Figure 5.24: Control inputs of agent 2 using DNMPC for dynamic system in simulations

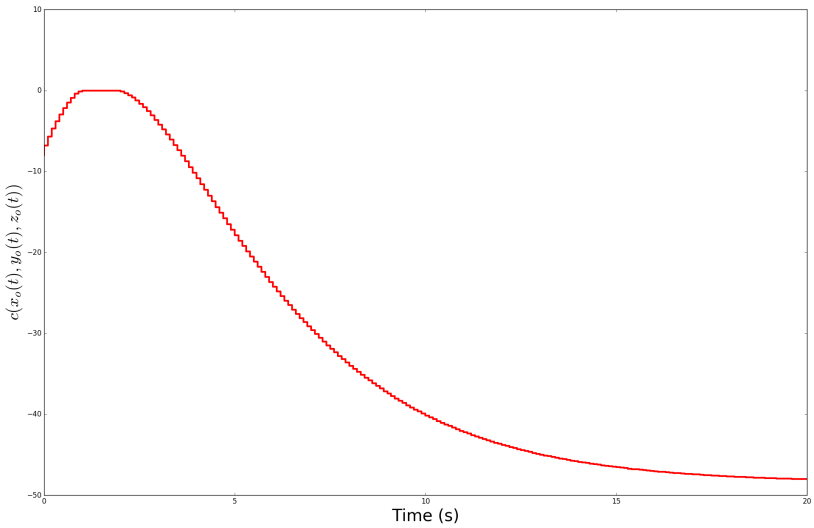


Figure 5.25: Obstacle function using DNMPC for kinematic system in simulations

5.2.3 Summary

The simulation results show that for both CNMPC scheme and DNMPC scheme the states of the agents and object are able to converge to the destination with singularity avoidance and obstacle avoidance. The DNMPC is faster than CNMPC, since it reduces the computational burden as expected. Both DNMPC and CNMPC are fast enough for experiments.

5.3 Summary

In this chapter, the simulation results for CNMPC and DNMPC are presented. The simulation results prove the convergence and feasibility of CNMPC and DNMPC. The simulation results also show that kinematic model is more suitable for experiments.

Chapter 6

Experiments

Experiments are very important, especially for robotic researchers.

There are a lot of things that are not considered in simulation but exist in real life. In simulation, the model is considered as an accurate model and there is no noise and time delay. Experiments can evaluate the performance of controllers in real world, which will have model uncertainty, some measurement noise, and time delay. A good controller should be robust enough to deal with the noise and uncertainty in real life.

This chapter is divided into three sections. Section 6.1 introduces components used in experiments. System architecture is revealed in Section 6.2. Experiment results are presented in Section 6.3.

6.1 Experimental Setup

There are three components are used in experiments.

- Mobile base
- Manipulators
- Object

The position of the components are captured by Mocap. The experiments are conducted via ROS. In this section, the components used in experiments, Mocap and their connection via ROS are introduced.

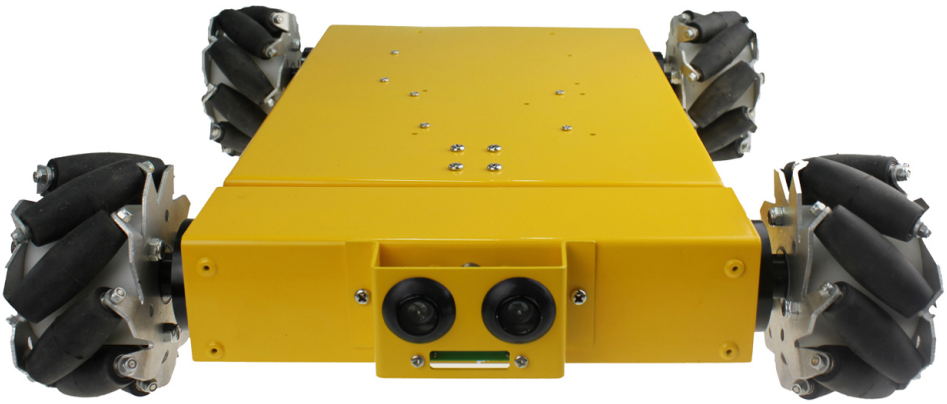


Figure 6.1: 4WD mecanum wheel mobile arduino robotics car

6.1.1 Mobile Base

The mobile bases provide a great mobility of agents. In experiments, we use two 4WD mecanum wheel mobile Arduino robotics cars, Nexus Robot 10011 [1], as mobile bases of agents. A figure of Nexus Robot 10011 is shown in Fig. 6.1. The size of Nexus Robot 10011 is $400mm \times 360mm \times 100mm$. Nexus Robot 10011 is a 4-wheel-drive aluminum alloy vehicle with four $100mm$ Aluminium Mecanum wheels. Mecanum wheel is also known as Swedish wheel, which is shown in Fig. 6.2[2]. The rollers of a mecanum wheel is attached to its circumference. The rollers have a 45-degree rotation relative to the plane of the wheel. Mecanum wheel provides a compact size, a high load capacity, and omni direction capabilities of Nexus Robot 10011.

Each mecanum wheel on Nexus Robot 10011 is connected to an independent Faulhaber 12V motors with optical encoders. By controlling the speed of each wheel, Nexus Robot 10011 is able to move forward, backward, left, and right. Nexus Robot can also rotate clockwise and counterclockwise. In other word, Nexus Robot has three degree of freedom, i.e moving forward/backward, moving left/right and rotation. By combining the three degree of freedom, Nexus Robot is able to

[1] http://www.nexusrobot.com/product/4wd-mecanum-wheel-mobile-arduino-robotics-car-10011.html

[2] http://www.nexusrobot.com/product/4-inch100mm-aluminum-mecanum-wheels-set-basic-2-left-2-right-14162.html

Figure 6.2: 100mm mecanum wheel

move towards any direction. Fig. 6.3 [3] shows the relationship between the direction of wheels and the movement of Nexus Robot 10011.

The speed of the mecanum wheels is controlled by a PID controller on Arduino 328 Controller and Arduino IO expansion board. The wheels will adjust the velocity to the reference velocity using PID controller according to the feedback signal of encoders.

6.1.2 Manipulators

Manipulators are the core of cooperative transportation tasks. Manipulators can grasp the object and avoid collision with obstacles. In this thesis, two WidowX Robot Arms [4] from Trossen Robotics [5] are used as manipulators.

A WindowX Robot Arm provides five degrees of freedom and a custom designed parallel gripper for high precision and maximum gripping strength by using two Dynamixel MX-64T Robot Actuators

[3] https://seeeddoc.github.io/4WD_Mecanum_Wheel_Robot_Kit_Series/
[4] http://www.trossenrobotics.com/widowxrobotarm
[5] http://www.trossenrobotics.com/

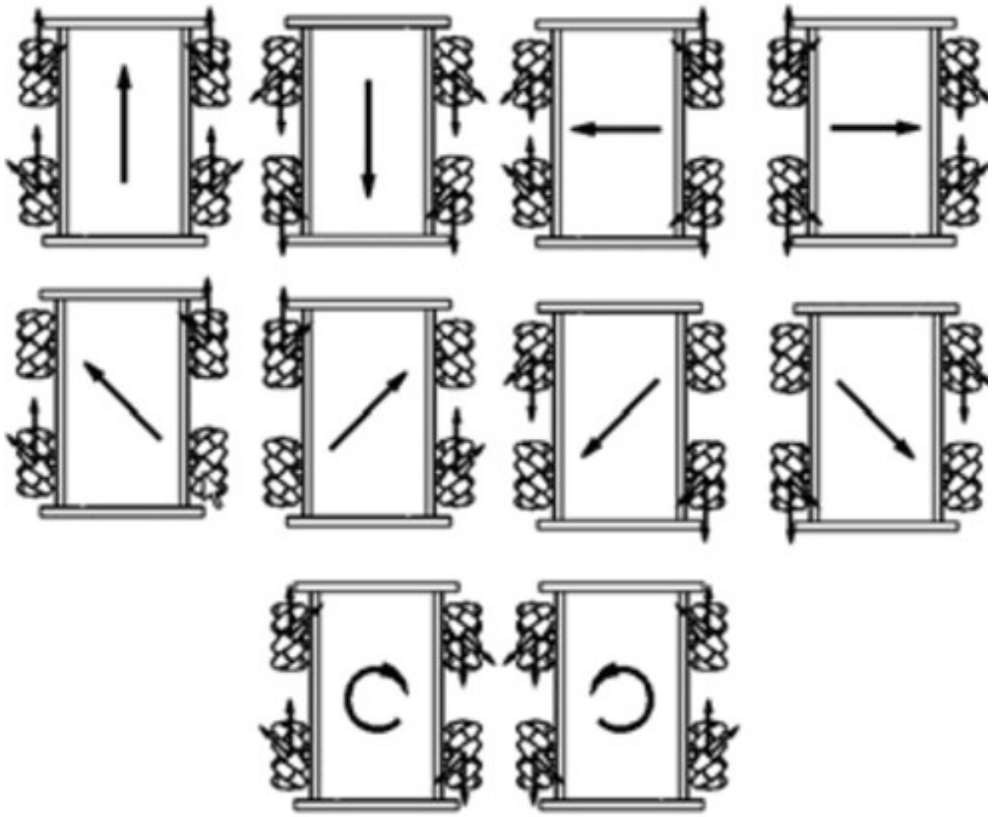


Figure 6.3: The working principle of Nexus Robot 10011

[6], two Dynamixel MX-28T Robot Actuators [7] and two Dynamixel AX-12A Robot Actuators [8]. As is shown in Fig. 6.4, Motor 2 and Motor 3 are two Dynamixel MX-64T Robot Actuators. Motor 1 and Motor 4 are two Dynamixel MX-28T Robot Actuators. Motor 5 and Motor 6 are two Dynamixel AX-12A Robot Actuators. In this thesis, only Motor 2, Motor 3 and Motor 4 are used as the joints of manipulators.

Each Dynamixel MX actuator provides a 360-degree movement with a ultra-high resolution of 4096 positions. Two user-defined PID controllers, which provide position control and speed control respectively, are provided by each Dynamixel MX actuator. Dynamixel AX-12A Robot Actuators act as the actuator of the parallel gripper. The gripper can holding an object up to 500g.

The WindowX Robot Arm is powered by a 12 Volts, 10 Amps Power

[6]https://www.trossenrobotics.com/p/mx-64t-dynamixel-robot-actuator.aspx
[7]https://www.trossenrobotics.com/dynamixel-mx-28-robot-actuator.aspx
[8]https://www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx

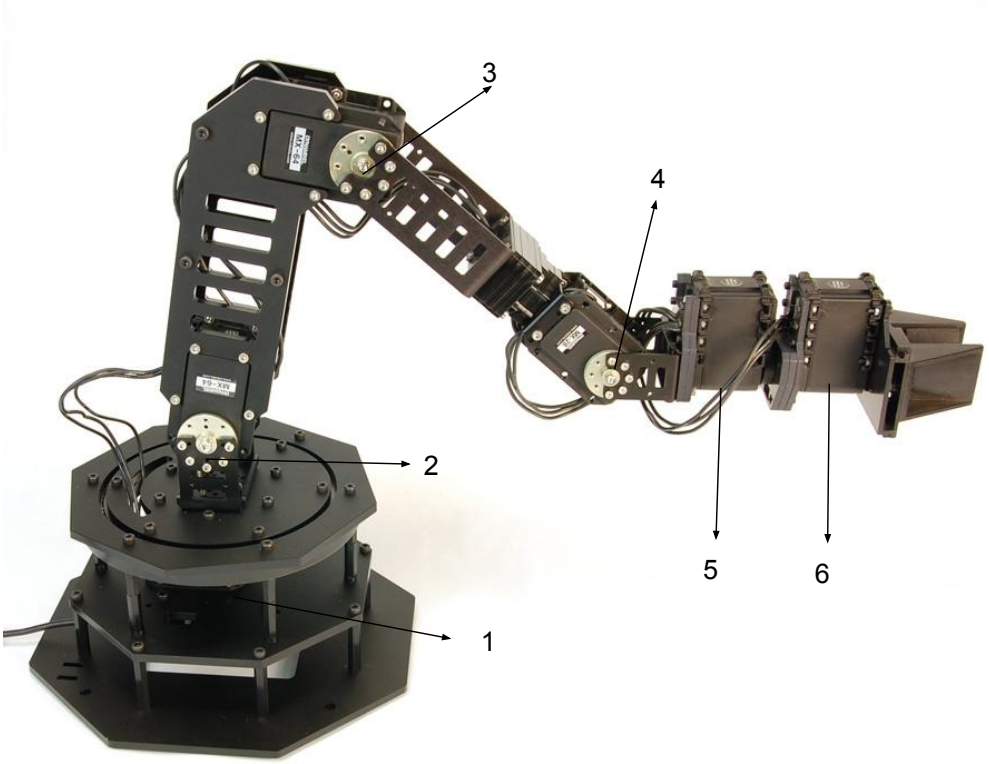


Figure 6.4: WindowX robot arm

Supply. Each WindowX Robot Arm has a ArbotiX-M Robocontroller [9], which is an Arduino compatible board, as controller. The figure of ArbotiX-M Robocontroller is shown in Fig. 6.5. The microprocessor on ArbotiX-M Robocontroller is ATMega644p, which is a quite powerful microprocessor. The ArbotiX-M Robocontroller can be programmed using the Arduino IDE or custom firmware. It also provides 8 digital and analog IOs, Xbee wireless and USB connectivity.

6.1.3 Object

The test object for cooperative transportation is a $13cm \times 7.5cm \times 3cm$ paper box in this thesis. The object will be hold by two mobile manipulators together. A picture of the test object is shown in Fig. 6.6.

[9] https://www.trossenrobotics.com/p/arbotix-robot-controller.aspx

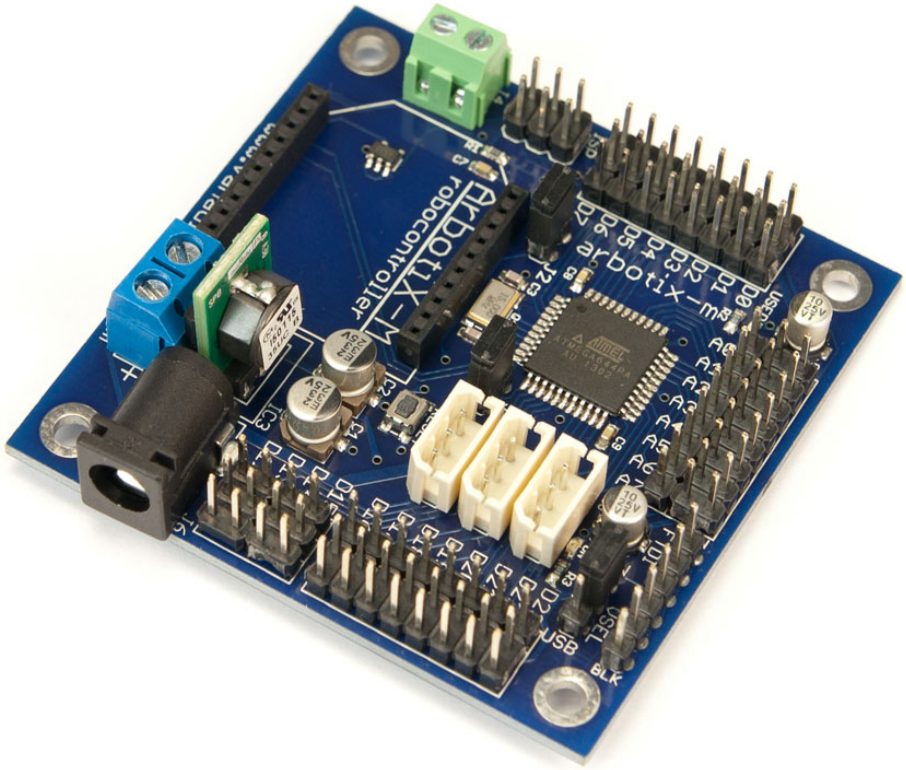


Figure 6.5: ArbotiX-M Robocontroller

6.1.4 Mobile Manipulators

In cooperative transportation experiments, two mobile manipulators are used as agents. As is shown in Fig. 6.7, a mobile manipulators consists of a battery, a voltage converter, a Raspberry Pi, a Nexus Robot 10011, and a WidowX Robot Arm.

The battery of a mobile manipulator is a 'Storm 14.8V 4000mAh 30C Pro Series Li-Po Battery (XT60)' [10], which is shown in Fig. 6.8. The battery, whose weight is 407 gram, has four cells with size $140mm \times 41mm \times 31mm$. Each cell will stay at 3.7 Volts when working and reach

[10] http://www.helipal.com/storm-14-8v-4000mah-30c-pro-series-li-po-battery-xt60.html

Figure 6.6: Test object

a flat voltage drop at late stage. The working voltage of the battery is 14.8 Volts, while the capacity of the battery is 4000 mAh.

A three-bit digital tube is connected with the battery. When the battery level is below the working voltage, the digital tube alarmed to protect the battery.

A Raspberry Pi 3, which is shown in Fig. 6.9, is used to control the mobile manipulator. The Raspberry Pi is able to run ROS node on it and communicate with other ROS node by connecting in the same network. The nodes of the mobile base and the manipulator are run on the Raspberry Pi. The Raspberry Pi will send control signal to the mobile base and the joints of the manipulator.

The Raspberry Pi 3 is powered by a 5.1 V micro USB supply. To provide the working voltage for the Raspberry Pi, a voltage converter is needed to convert the working voltage of the Li-Po battery, i.e 14.8 V, to 5.1 V. The voltage converter is shown in Fig. 6.10. The manipulator is powered by the Li-Po battery directly. The mobile base has its own independent battery.

The distance between the first joint of the mobile manipulator and

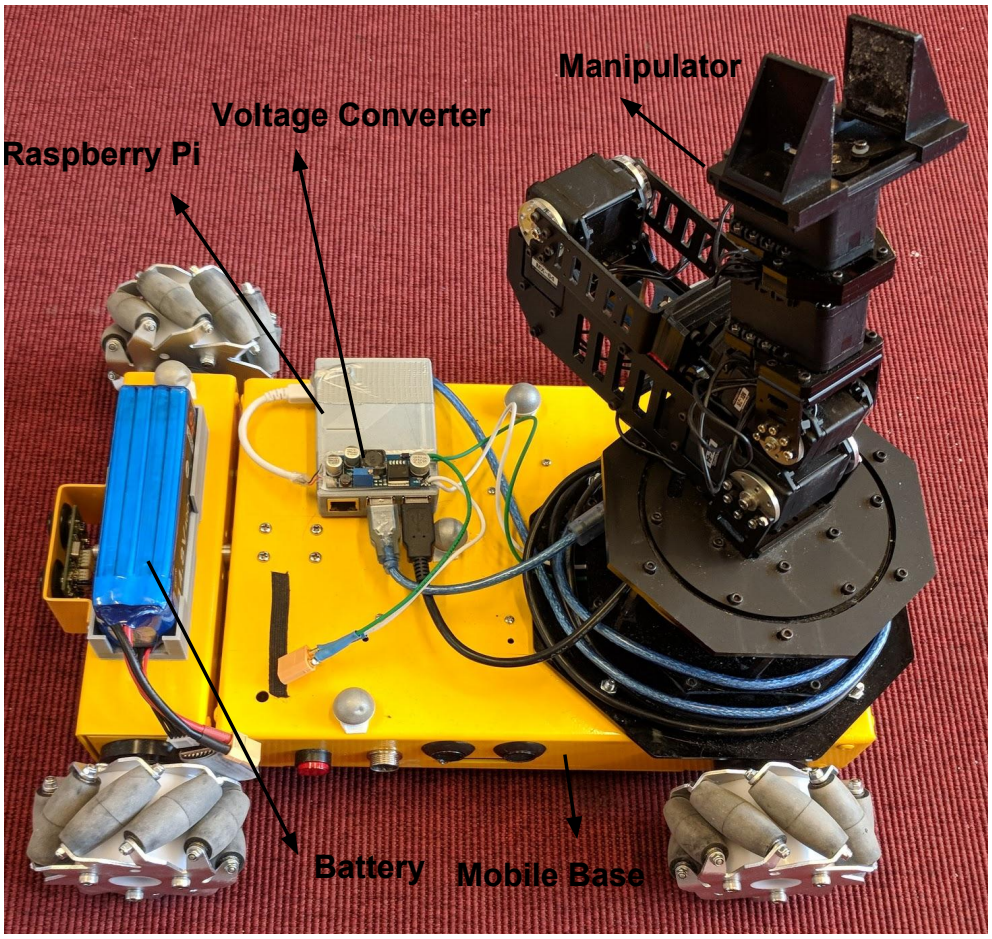


Figure 6.7: Mobile manipulator

the second joint of the mobile manipulator is 15.13 cm. The distance between the second joint of the mobile manipulator and the thrid joint of the mobile manipulator is 14.32 cm. The length of the end-effector is 11.5 cm.

6.1.5 Mocap

All the experiments are conducted in Smart Mobility Lab (SML), KTH, Stockholm. SML provides a motion capture system (Mocap) with 12 Qualisys cameras [11] spread across the lab. The Mocap can track a rigid body at freqency of 100 Hz and broadcast its pose, linear velocities and angular velocities by using ROS. In this thesis, only pose of the robots

[11] https://www.qualisys.com/cameras/oqus/

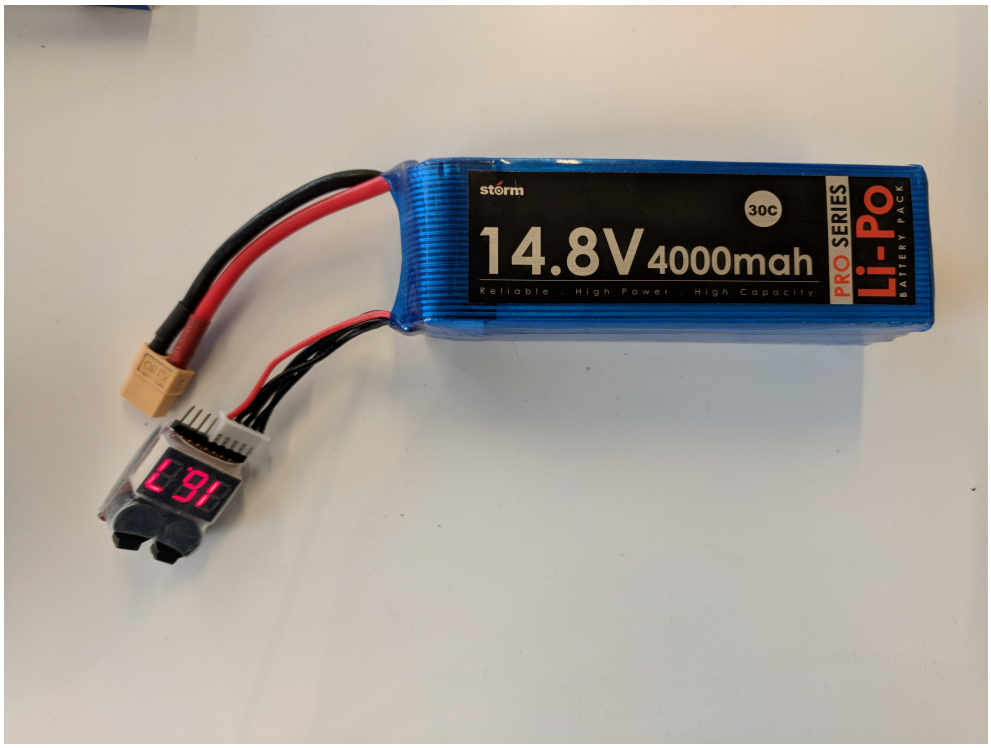


Figure 6.8: Battery

are needed. For pose measurement, the maximum error of Mocap is 6 mm.

6.2 System Architecture

The hardware used in experiments are connected using Gigabit Ethernet connections, USB connections, and wireless 5.8Ghz connections. As is shown in Fig. 6.11, each agent consists of a Nexus Robot, a WindowX Arm, and a Raspberry Pi. The Nexus Robot and the WindowX Arm are connected with the Raspberry Pi using USB connections. The two Raspberry Pi on the agents are connected with a TP-Link Router via wireless 5.8Ghz connections. Host Computer, which will run MPC controllers, is connected with the TP-Link Router via Gigabit Ethernet connections. Mocap, which will provide the pose of agents, is also connected with the TP-Link Router using Gigabit Ethernet connections.

Fig. 6.12 shows the ROS architecture of experiments. There are four devices, which has ROS nodes running on them. Host computer per-

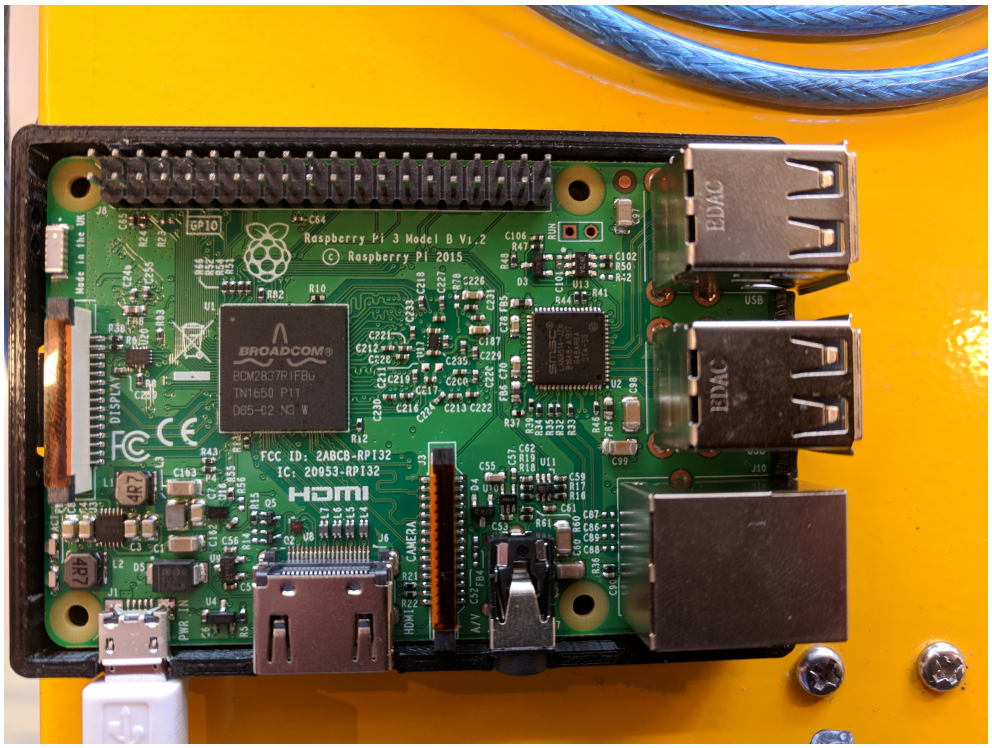


Figure 6.9: Raspberry Pi

forms as roscore in the network, which provides interface with all the nodes of vehicles, manipulators, and MoCap. Host computer runs a node of controller, which receives the measurement from MoCap and encoders on the arms, calculates the control signal using CNMPC or DNMPC scheme and send it to the vehicles and arms. Host computer is with 8 cores, 3.40GHz Intel ® Core™ i7-6700 CPU, 32GB of RAM. There are two nodes run on a Raspberry Pi. One node is to control the vehicle, which subscribes the control input and actuates on the vehicle. The other node is used to control the arm, which subscribes the control input, actuates on the arm and publish the position of the joints.

6.3 Experiment Result

Experiments based on kinematic systems are conducted. Experiment results show the robustness and efficiency of DNMPC scheme and CNMPC scheme.

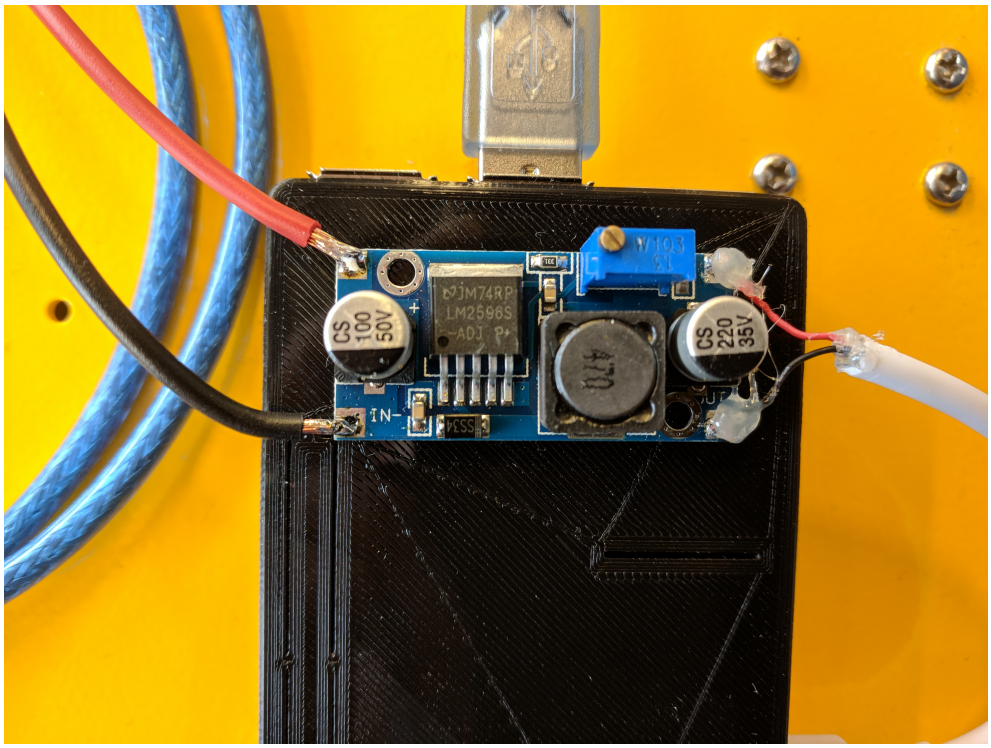


Figure 6.10: Converter

6.3.1 Centralized MPC

According to the configuration, we have that $x = [x_o^\top, q]^\top \in \mathbb{R}^{16}$, with $x_o = [p_o^\top, \phi_o]^\top \in \mathbb{R}^4$, $p_o = [x_o, y_o, z_o]^\top \in \mathbb{R}^3$, $q = [q_1^\top, q_2^\top] \in \mathbb{R}^{12}$, $q_i = [p_{B_i}^\top, \alpha_i^\top] \in \mathbb{R}^6$, $p_{B_i} = [x_{B_i}, y_{B_i}, \psi_{B_i}]^\top \in \mathbb{R}^3$, $\alpha_i = [\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_3}]^\top \in \mathbb{R}^3$, $i \in \{1, 2\}$. The ground vehicle can rotate around z-axis clockwise or counter-clockwise. Thus, there is a constraint for the ground vehicle:

$$-2\pi \leq \psi_{B_i} \leq 2\pi,$$

where $i \in \{1, 2\}$.

The manipulators become singular when $\alpha_{i_2} = 0$, $i \in \{1, 2\}$. Then, the constraints for the manipulators are set to:

$$-\frac{\pi}{2} < \alpha_{1_1} < \frac{\pi}{2}, \epsilon < \alpha_{1_2} < \frac{\pi}{2}, -\frac{\pi}{2} < \alpha_{1_3} < \frac{\pi}{2},$$
$$-\frac{\pi}{2} < \alpha_{2_1} < \frac{\pi}{2}, -\frac{\pi}{2} < \alpha_{2_2} < -\epsilon, -\frac{\pi}{2} < \alpha_{2_3} < \frac{\pi}{2},$$

where ϵ is an arbitrary small value, which is larger than zero.

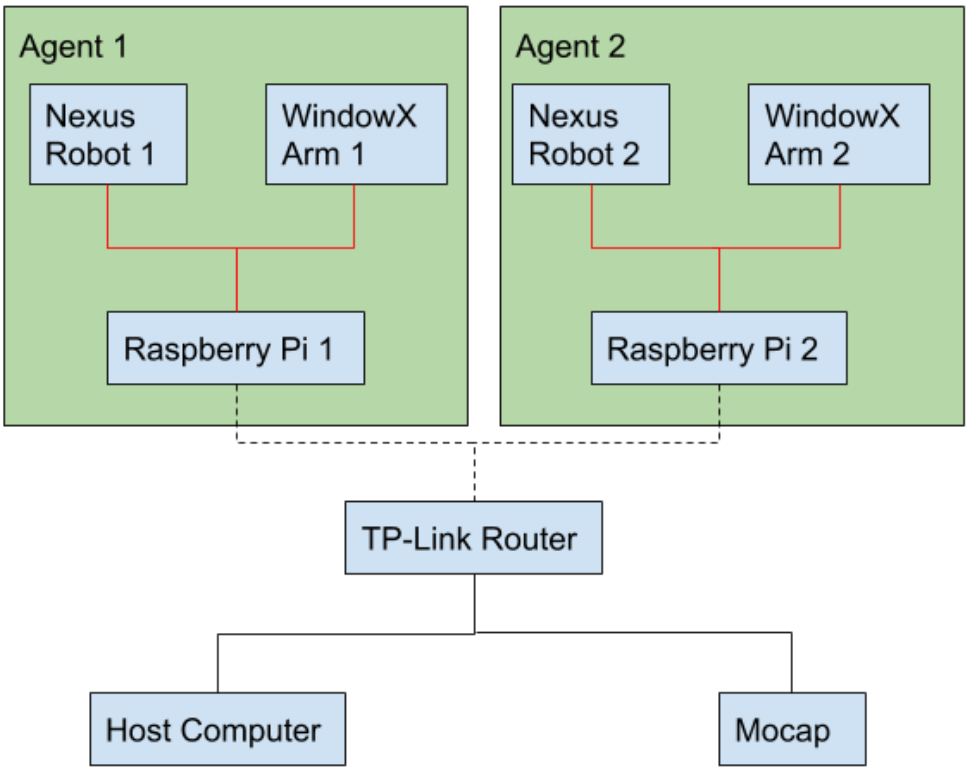


Figure 6.11: Hardware connections: Gigabit Ethernet connections are in black. USB connections are in red. wireless 5.8Ghz connections are in dash.

The input of the system are the velocities of the object. Assume u_1 denotes the linear velocity of the object on x-axis; u_2 denotes the linear velocity of the object on y-axis; u_3 denotes the linear velocity of the object on z-axis and u_4 denotes the angular velocity of the object around x-axis. Then the input constraints are set to:

$$
\begin{aligned}
&-2dm/s \leq u_1(t) \leq 2dm/s;\\
&-0.4dm/s \leq u_2(t) \leq 0.4dm/s;\\
&-0.4dm/s \leq u_3(t) \leq 0.4dm/s;\\
&-0.02rad/s \leq u_4(t) \leq 0.02rad/s.
\end{aligned}
$$

The initial conditions are set to:

$$x_o = \begin{bmatrix} -8 & -2.8126 & 2.5256 & \frac{\pi}{2} \end{bmatrix}^\top,$$
$$q_1 = \begin{bmatrix} -8, & 0, & 0, & 0, & \frac{\pi}{4}, & \frac{\pi}{4} \end{bmatrix}^\top,$$
$$q_2 = \begin{bmatrix} -8, & -5.6252, & 0, & 0, & -\frac{\pi}{4}, & -\frac{\pi}{4} \end{bmatrix}^\top.$$

The desired goal states are set to:

$$x_o = \begin{bmatrix} 10 & -2.8126 & 2.5256 & \frac{\pi}{2} \end{bmatrix}^\top,$$
$$q_1 = \begin{bmatrix} 10, & 0, & 0, & 0, & \frac{\pi}{4}, & \frac{\pi}{4} \end{bmatrix}^\top,$$
$$q_2 = \begin{bmatrix} 10, & -5.6252, & 0, & 0, & -\frac{\pi}{4}, & -\frac{\pi}{4} \end{bmatrix}^\top.$$

The obstacle is set between the initial and the desired position of the object. The obstacle is a sphere whose center is $\begin{bmatrix} 3 & , -2.8126, & 3.5 \end{bmatrix}$ and radius is 0.7071. The sampling time is 0.1 seconds. The horizon is 3 and the total simulation time is 20 seconds. The matrices P, Q, R are set to :

The desired goal states are set to:

$$P = diag\{1, 1, 1, 1, 1, 1, 1, 20, 20, 20, 1, 1, 1, 20, 20, 20\},$$
$$Q = diag\{1, 1, 1, 1, 1, 1, 1, 20, 20, 20, 1, 1, 1, 20, 20, 20\},$$
$$R = diag\{1, 1, 2, 2\}.$$

Fig. 6.13a and Fig. 6.14a show the error between states of agent 1 and the destination of agent 1. Similarly, Fig. 6.13b and Fig. 6.14b are the error of agent 2 while Fig. 6.13c and Fig. 6.14c are the error of object.

Fig. 6.15 and Fig. 6.16 shows the states of agent 1, the states of agent 2 and the states of the object.

Fig. 6.17 and Fig. 6.18 show the control inputs of CNMPC, which are applied to agents. Fig. 6.19 shows the obstacle function of CNMPC.

A 3D figure of the trajectory of object and the obstacle is depicted in Fig. (6.20). Fig. (6.20) shows that the obstacle is avoided by using CNMPC controller. The experimental results show that the states of the agents and the object converge to the goal states and the obstacle is avoided. The experiment takes around 35 seconds. Each iteration takes 0.05 seconds.

6.3.2 Decentralized MPC

For agent i, we have that $x_i = [q_i]^\top \in \mathbb{R}^6$, with $q_i = [p_{B_i}^\top, \alpha_i^\top]^\top \in \mathbb{R}^6$, $p_{B_i} = [x_{B_i}, y_{B_i}, \psi_{B_i}]^\top \in \mathbb{R}^3$, $\alpha_i = [\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_3}]^\top \in \mathbb{R}^3$, $i \in \{1, 2\}$.

The ground vehicle can rotate around z-axis clockwise or counter-clockwise. Thus, there is a constraint for the ground vehicle:

$$-2\pi \leq \psi_{B_i} \leq 2\pi$$

,where $i \in \{1, 2\}$.

Thus, the manipulators become singular when $tan(\alpha_{i_1} + \alpha_{i_2}) = tan(\alpha_{i_1})$, i.e $\alpha_{i_2} = 0$, $i \in \{1, 2\}$. Then, the constraints for the manipulator 1 are set to:

$$-\frac{\pi}{2} < \alpha_{1_1} < \frac{\pi}{2}, \epsilon < \alpha_{1_2} < \frac{\pi}{2}, -\frac{\pi}{2} < \alpha_{1_3} < \frac{\pi}{2}$$

, where ϵ is an arbitrary small value, which is larger than zero.

The inputs of agent i are denoted as $\{u_{i_1}, ..., u_{i_6}\}$, where $u_{i_1} = \dot{x}_{B_i}$, $u_{i_2} = \dot{y}_{B_i}$, $u_{i_3} = \dot{\psi}_{B_i}$, $u_{i_4} = \dot{\alpha}_{i_1}$, $u_{i_5} = \dot{\alpha}_{i_2}$, and $u_{i_6} = \dot{\alpha}_{i_3} \forall i \in \{1, 2\}$. The input constraints are

$$\begin{aligned}
&-2dm/s \leq u_{i_1}(t) \leq 2dm/s, \\
&-2dm/s \leq u_{i_2}(t) \leq 2dm/s, \\
&-0.7rad/s \leq u_{i_3}(t) \leq 0.7rad/s, \\
&-0.7rad/s \leq u_{i_4}(t) \leq 0.7rad/s, \\
&-0.7rad/s \leq u_{i_5}(t) \leq 0.7rad/s, \\
&-0.7rad/s \leq u_{i_6}(t) \leq 0.7rad/s,
\end{aligned}$$

where $i \in \{1, 2\}$.

For manipulator 2, the state constraints are

$$\begin{aligned}
&-0.15dm < p_{o_{1,x}}(t) - p_{o_{2,x}}(t) < 0.15dm, \\
&-0.15dm < p_{o_{1,y}}(t) - p_{o_{2,y}}(t) < 0.15dm, \\
&-0.15dm < p_{o_{1,z}}(t) - p_{o_{2,z}}(t) < 0.15dm, \\
&-0.05rad < p_{o_{1,\psi}}(t) - p_{o_{2,\psi}}(t) < 0.15rad,
\end{aligned}$$

where $p_{o_{1,x}}$ is the position of the object on x-axis, $p_{o_{1,y}}$ is the position of the object on y-axis, $p_{o_{1,z}}$ is the position of the object on z-axis and $p_{o_{1,\psi}}$ is the orientation of the object around x-axis calculated by agent i, $i \in \{1, 2\}$.

The initial conditions are set to:

$$q_1 = \left[\begin{array}{cccccc} -7.8, & 0, & 0, & 0, & \frac{\pi}{4}, & \frac{\pi}{4} \end{array} \right]^\top,$$
$$q_2 = \left[\begin{array}{cccccc} -7.8, & -5.6252, & 0, & 0, & -\frac{\pi}{4}, & -\frac{\pi}{4} \end{array} \right]^\top.$$

The desired goal states are set to:

$$q_1 = \left[\begin{array}{cccccc} 10, & 0, & 0, & 0, & \frac{\pi}{4}, & \frac{\pi}{4} \end{array} \right]^\top,$$
$$q_2 = \left[\begin{array}{cccccc} 10, & -5.6252, & 0, & 0, & -\frac{\pi}{4}, & -\frac{\pi}{4} \end{array} \right]^\top.$$

The obstacle is set between the initial and the desired position of the object. The obstacle is a sphere whose center is $\left[\begin{array}{ccc} 3 & , -2.8126, & 3.5 \end{array} \right]$ and radius is 0.7071. The sampling time is 0.1 seconds. The horizon is 3 and the total simulation time is 20 seconds.The matrices P_1, Q_1, R_1, P_2, Q_2, R_2 are set to :

$$P_1 = I_{6\times 6}, Q_1 = I_{6\times 6}, R_1 = diag\{1, 1, 1, 10, 10, 10\},$$
$$P_2 = I_{6\times 6}, Q_2 = I_{6\times 6}, R_2 = diag\{1, 1, 1, 10, 10, 10\}.$$

Fig. 6.21 shows the error between states of positions and the destination of positions of two agent. Similarly, Fig. 6.22 is the error of agent 2.

Fig. 6.23a and Fig. 6.23b show the states of agent 1's positions, the states of agent 2's positions, respectively. Fig. 6.24a and Fig. 6.24b show the states of agent 1's orientations, the states of agent 2's orientations, respectively. Fig. 6.25a and Fig. 6.25b show the states of the object.

The control input of agent 1 and the control input of agent 2 using DNMPC are depicted in Fig. 6.26 and Fig. 6.27, respectively.

The obstacle function using DNMPC is shown in Fig. 6.28.

A 3D figure of the trajectory of object and the obstacle is depicted in Fig. 6.29. Fig. 6.29 shows that the obstacle is avoided by using DNMPC controller.

The experiment results show that the states of the agents and the object converge to the goal states and the obstacle is avoided.

The experiment takes around 16 seconds. Each iteration takes around 0.02 seconds.

6.4 Summary

Both CNMPC scheme and DNMPC scheme are able to implement cooperative transportation tasks with singularity avoidance and collision avoidance. DNMPC is more robust in experiment.

The CNMPC scheme calculates the control inputs of the arms and vehicles from the states of the object, which makes it more sensitive to the position of the object. The DNMPC scheme calculates the control inputs of the arms and vehicles from their own states. CNMPC is more sensitive to model uncertainty and measurement noise. Moreover, the computational time of CNMPC is larger than DNMPC. Thus, DNMPC is more robust.

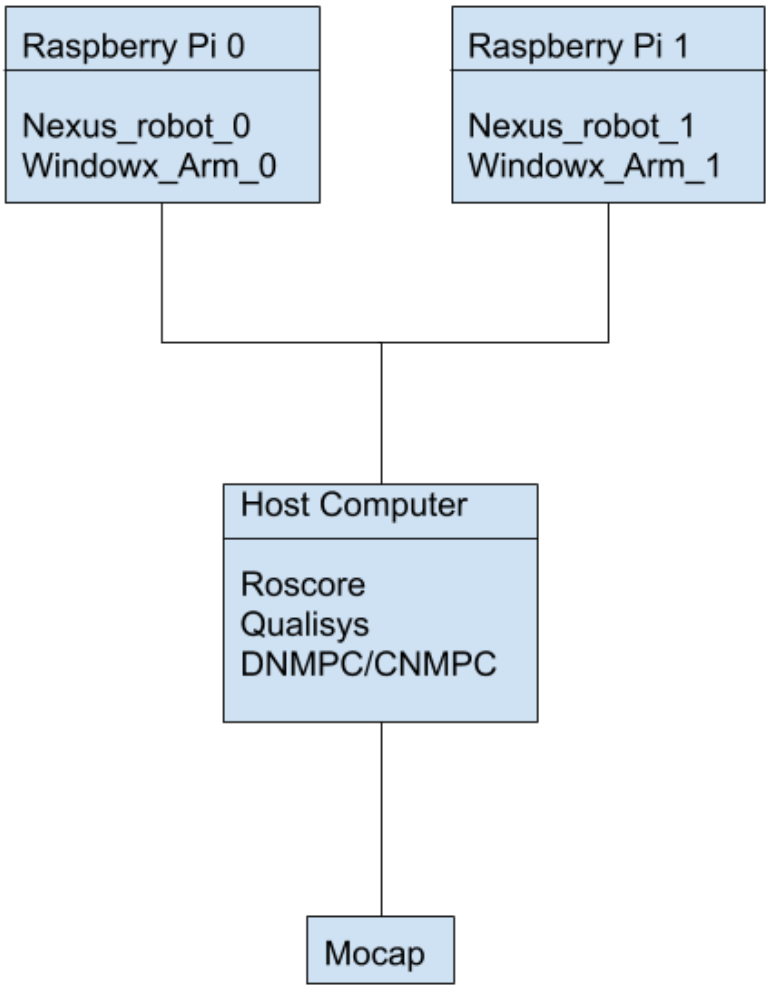
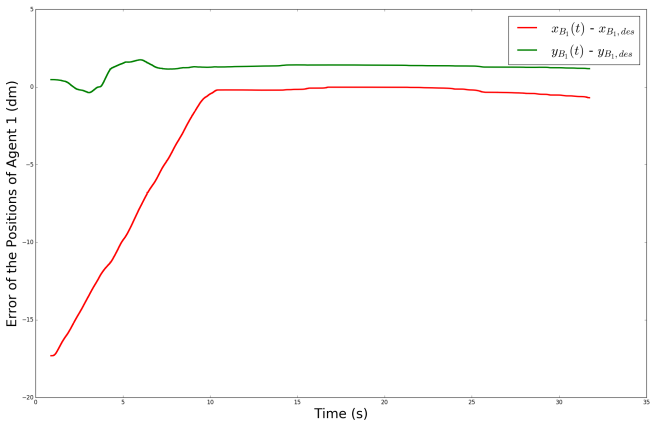
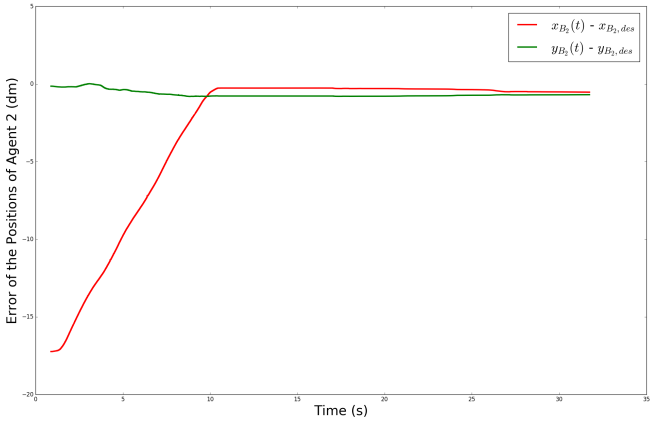


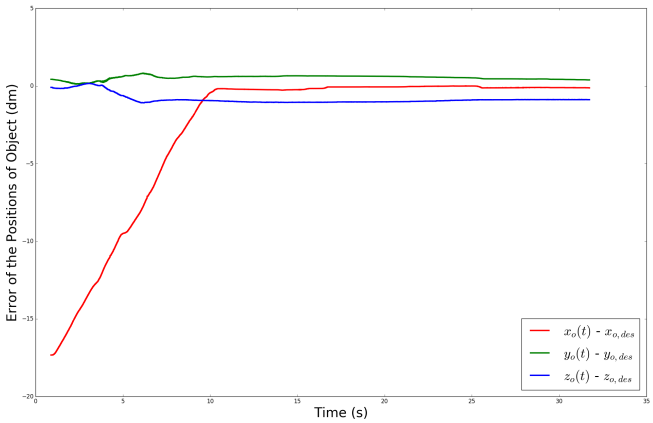
Figure 6.12: ROS architecture



(a) Error of agent 1's positions using CNMPC

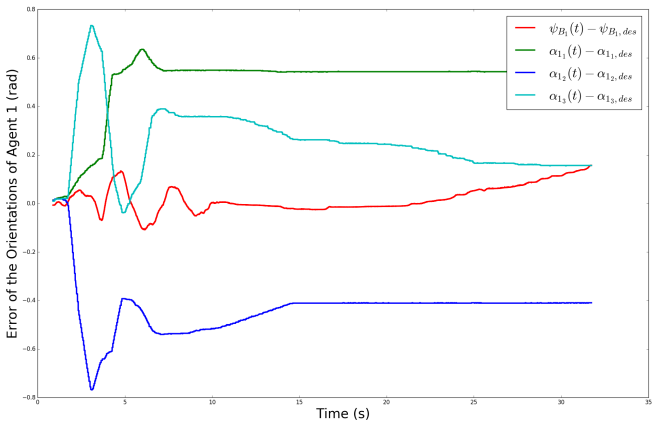


(b) Error of agent 2's positions using CNMPC

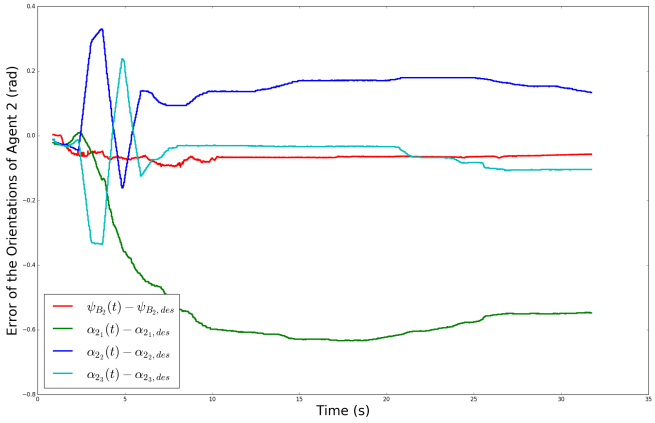


(c) Error of object's position using CNMPC

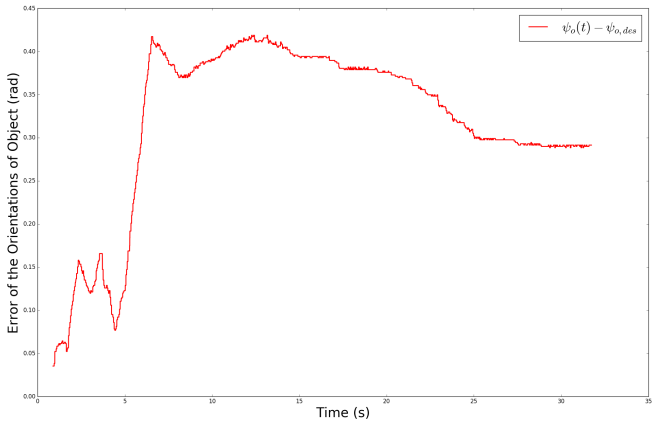
Figure 6.13: Errors of positions using CNMPC in experiments



(a) Error of agent 1's orientations using CNMPC

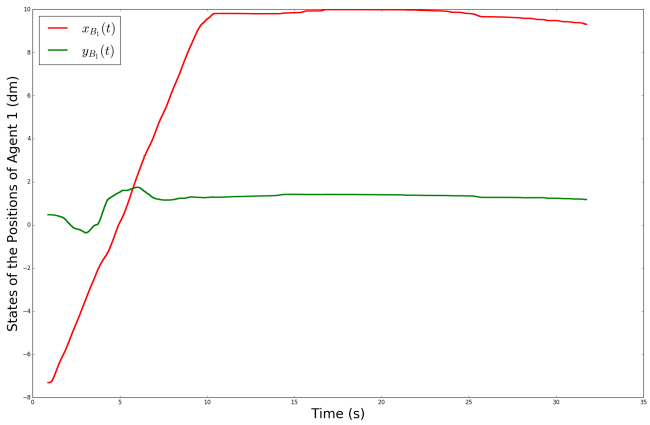


(b) Error of agent 2's orientations using CNMPC

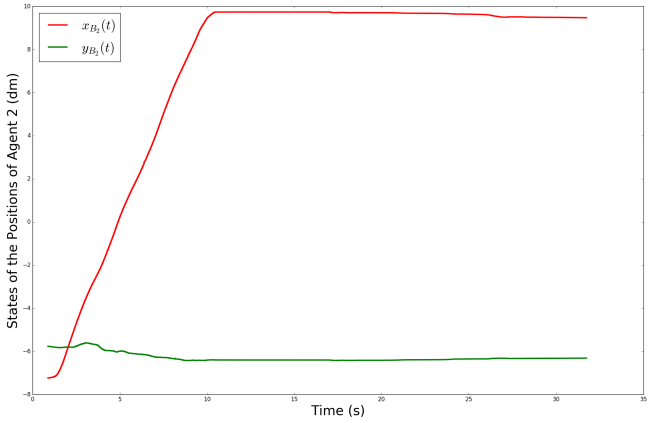


(c) Error of object's orientations using CNMPC

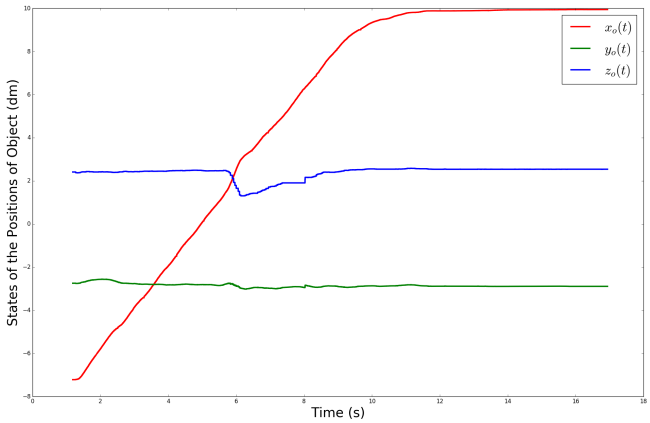
Figure 6.14: Errors of orientations using CNMPC in experiments



(a) States of agent 1's positions using CNMPC

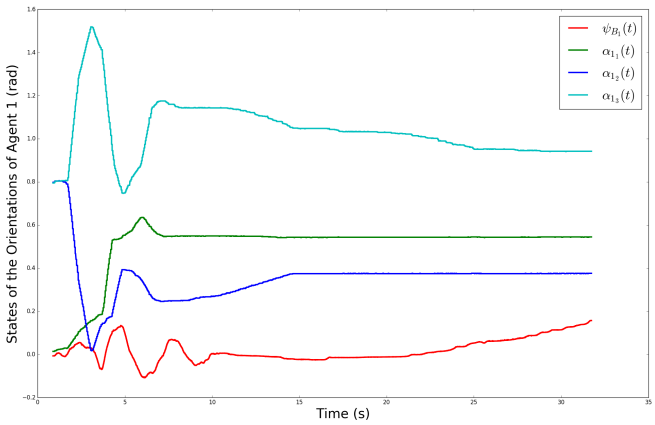


(b) States of agent 2's positions using CNMPC

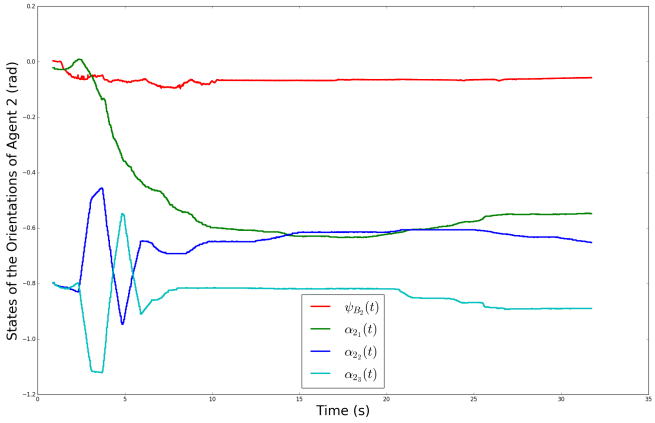


(c) States of object's positions using CNMPC

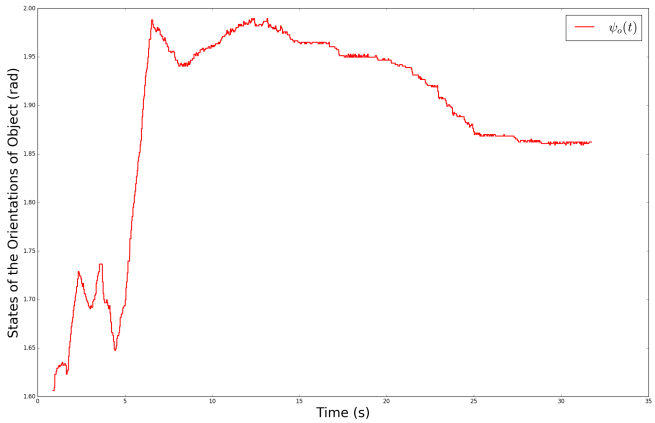
Figure 6.15: States of positions using CNMPC in experiments



(a) States of agent 1's orientations using CNMPC

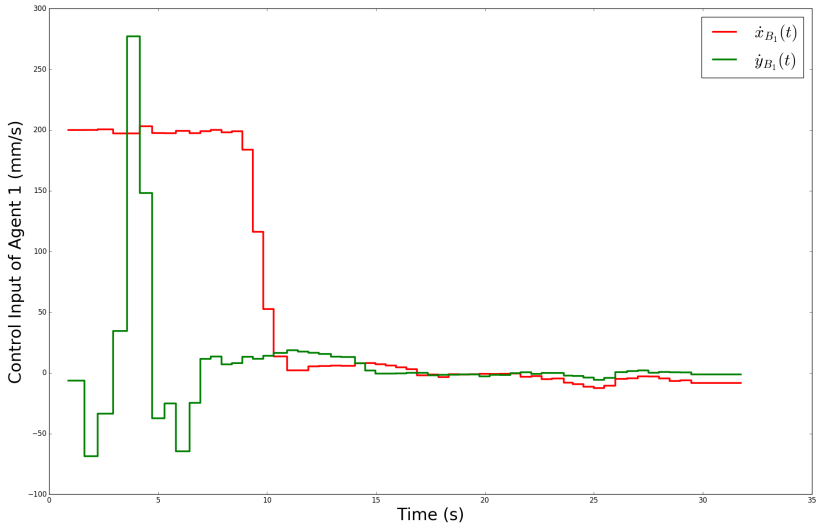


(b) States of agent 2's orientations using CNMPC

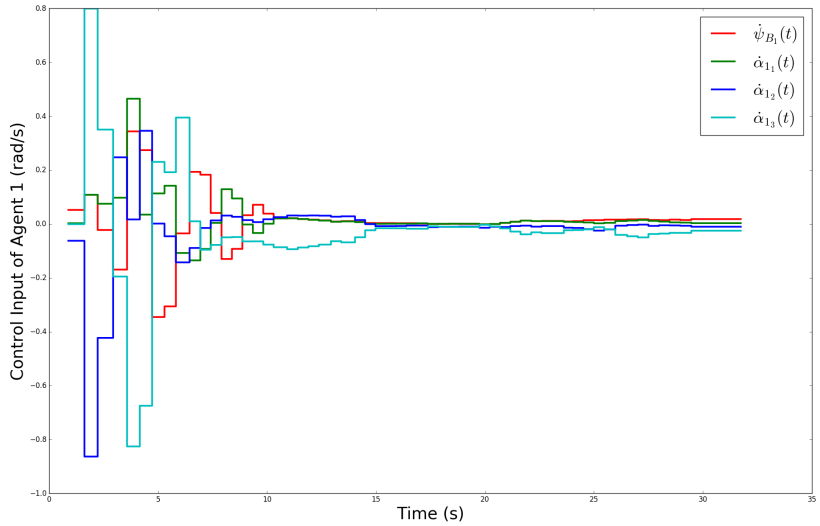


(c) States of object's orientations using CNMPC

Figure 6.16: States of orientations using CNMPC in experiments

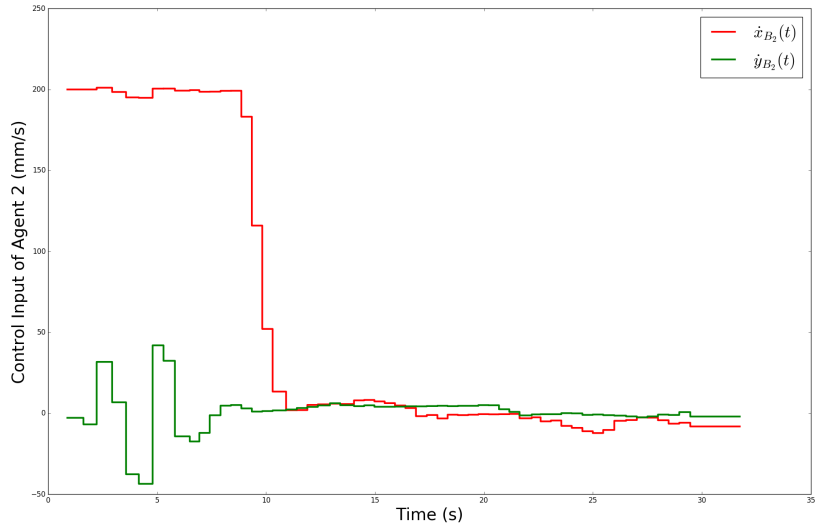


(a) Control inputs of agent 1's positions using CNMPC

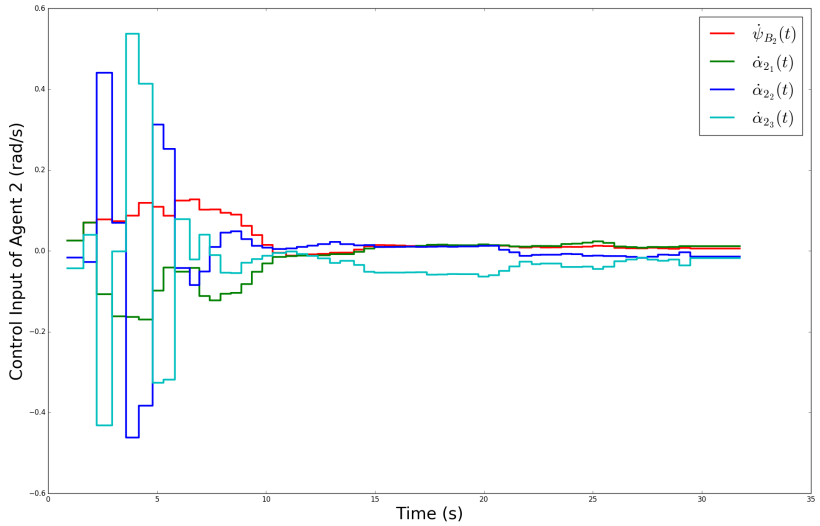


(b) Control inputs of agent1's orientations using CNMPC

Figure 6.17: Control inputs of agent 1 using CNMPC in experiments



(a) Control inputs of agent 2's positions using CNMPC



(b) Control inputs of agent2's orientations using CNMPC

Figure 6.18: Control inputs of agent 2 using CNMPC in experiments

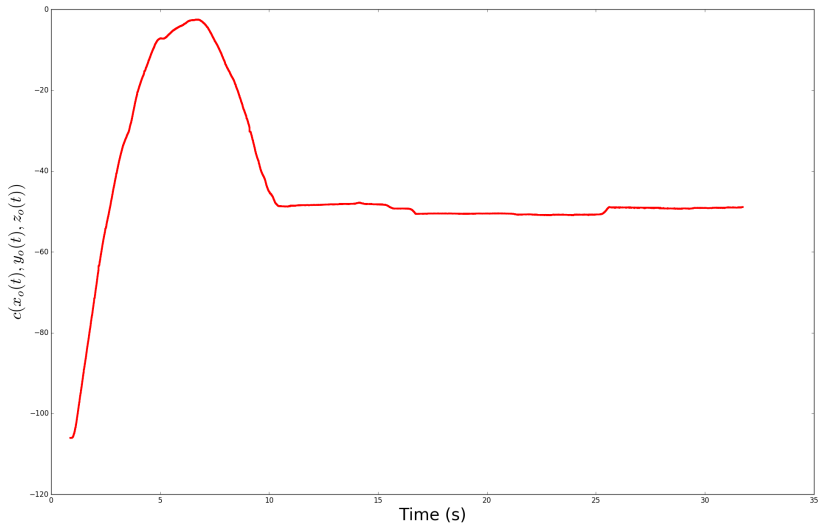


Figure 6.19: Obstacle function using CNMPC in experiments

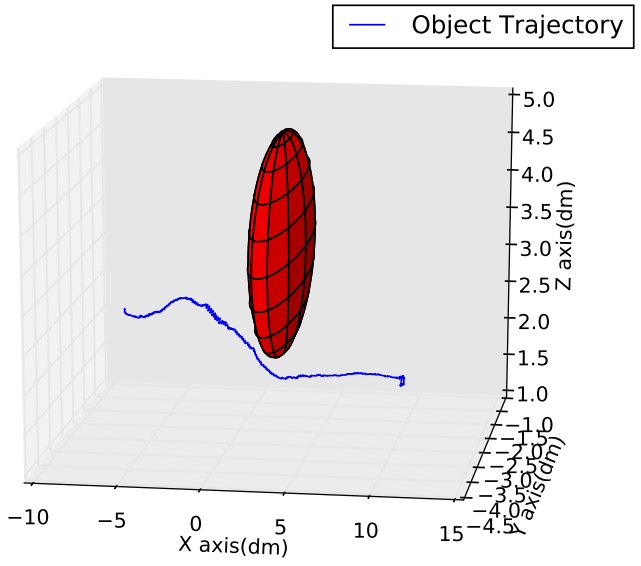
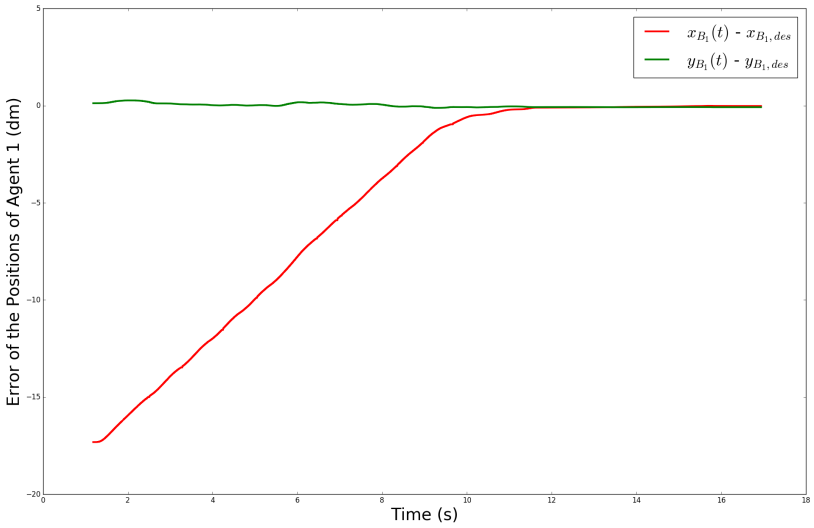
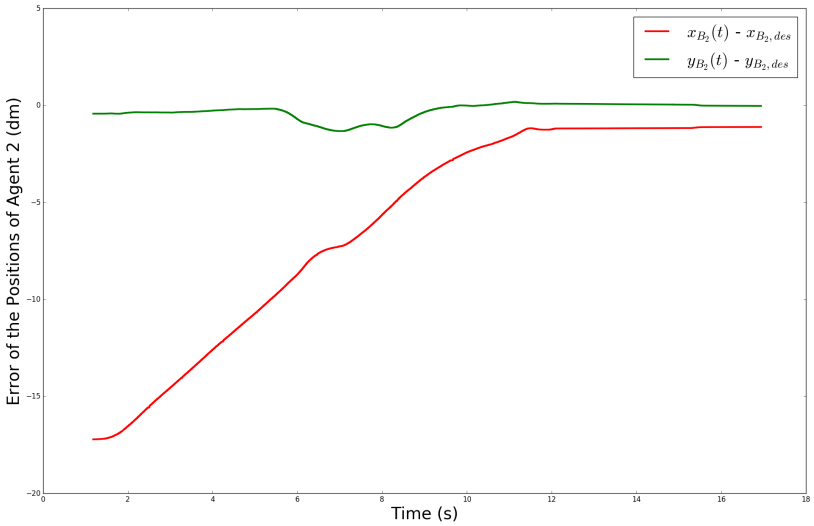


Figure 6.20: The trajectory of the object using CNMPC in experiments

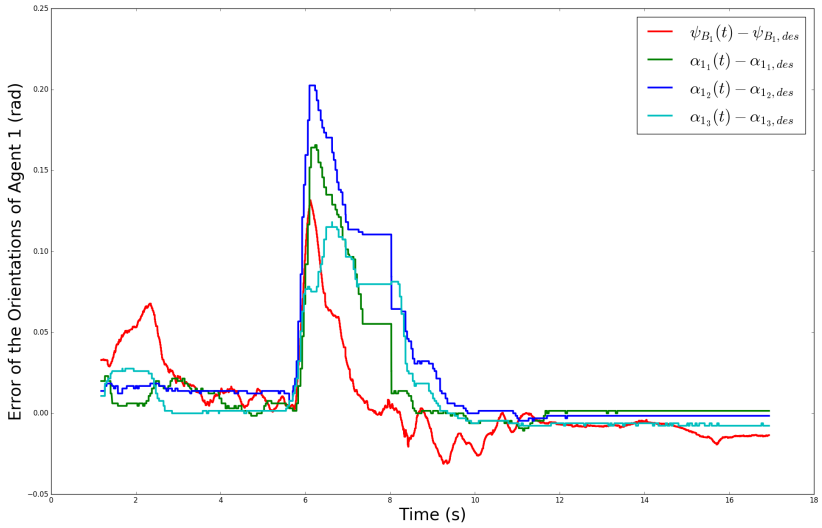


(a) Error of agent 1's positions using DNMPC

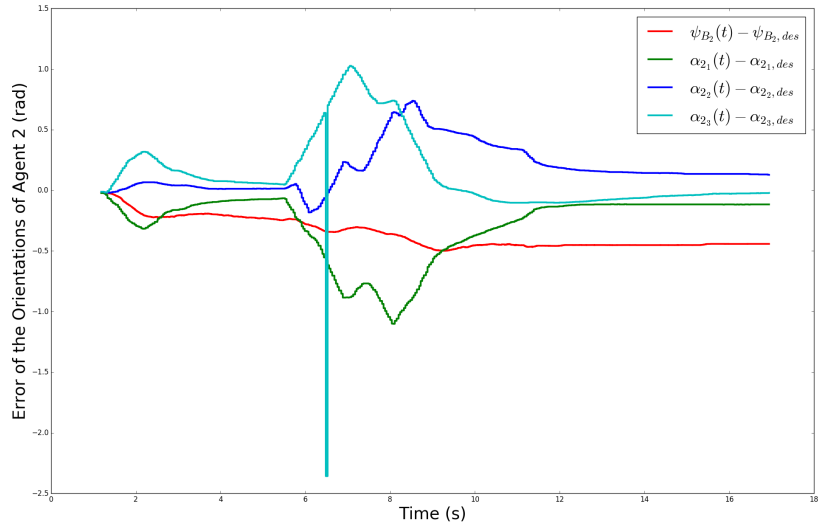


(b) Error of agent 2's positions using DNMPC

Figure 6.21: Errors of positions using DNMPC in experiments

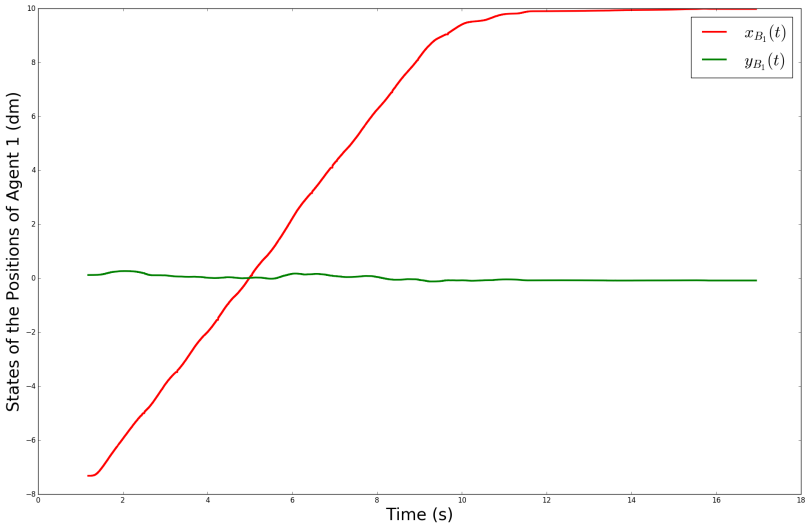


(a) Error of agent 1's orientations using DNMPC

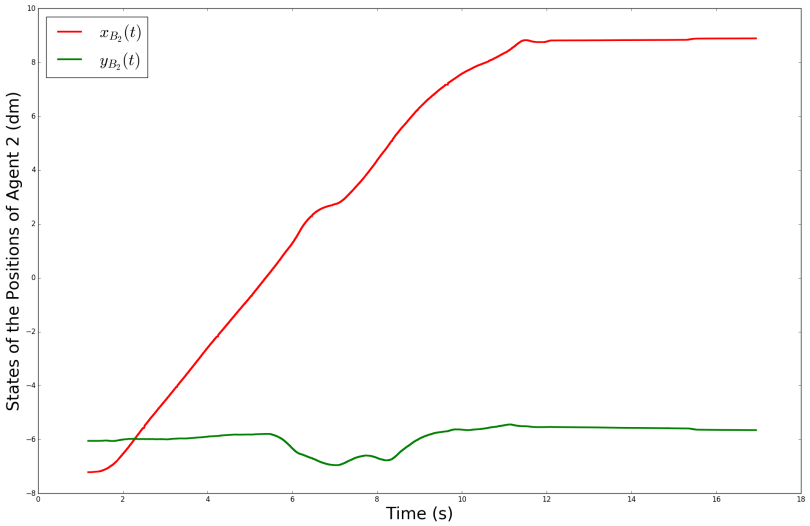


(b) Error of agent 2's orientations using DNMPC

Figure 6.22: Errors of orientations using DNMPC in experiments

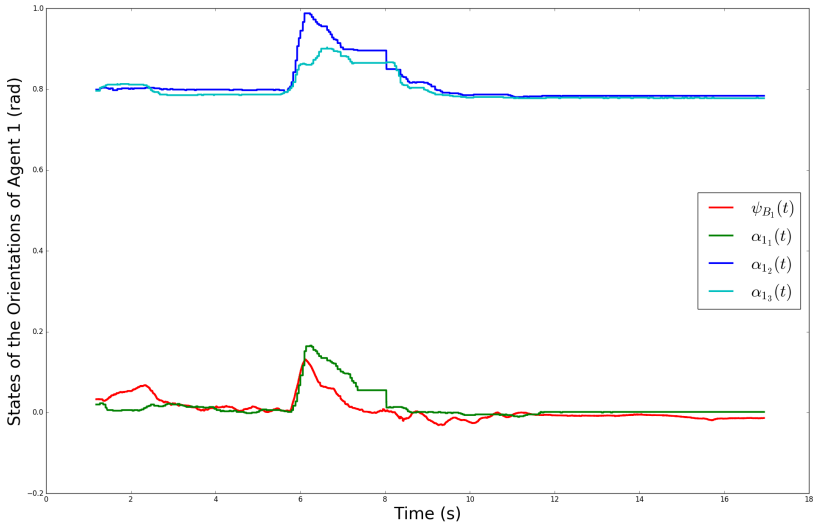


(a) States of agent 1's positions using DNMPC

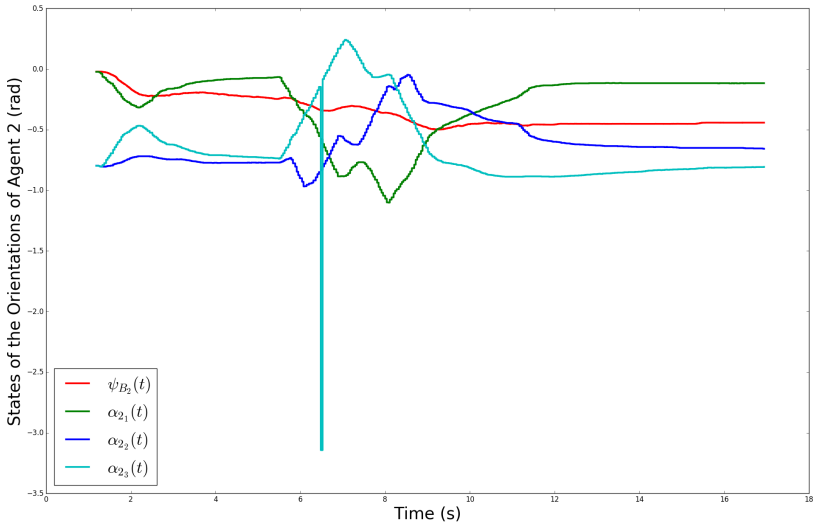


(b) States of agent 2's positions using DNMPC

Figure 6.23: States of agents' positions using DNMPC in experiments

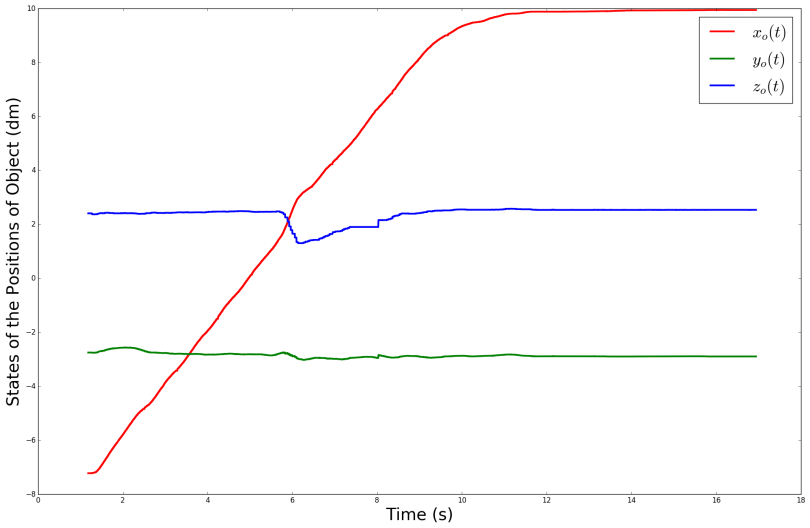


(a) States of agent 1's orientations using DNMPC

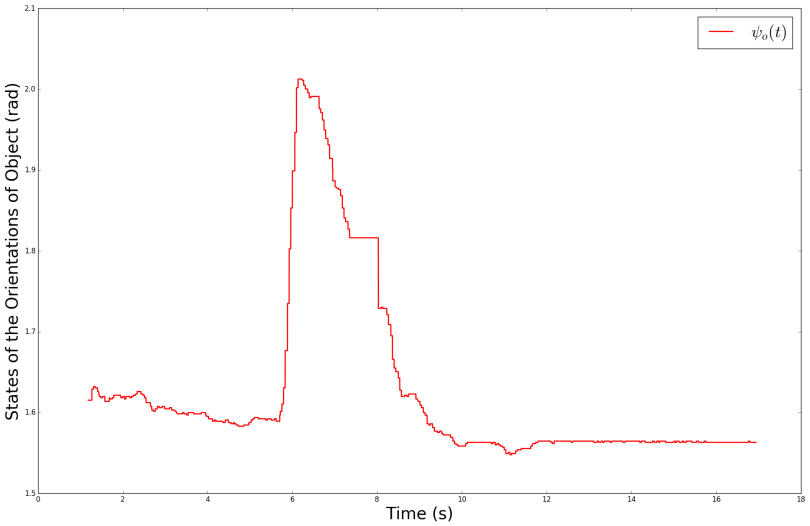


(b) States of agent 2's orientations using DNMPC

Figure 6.24: States of agents' orientations using DNMPC in experiments

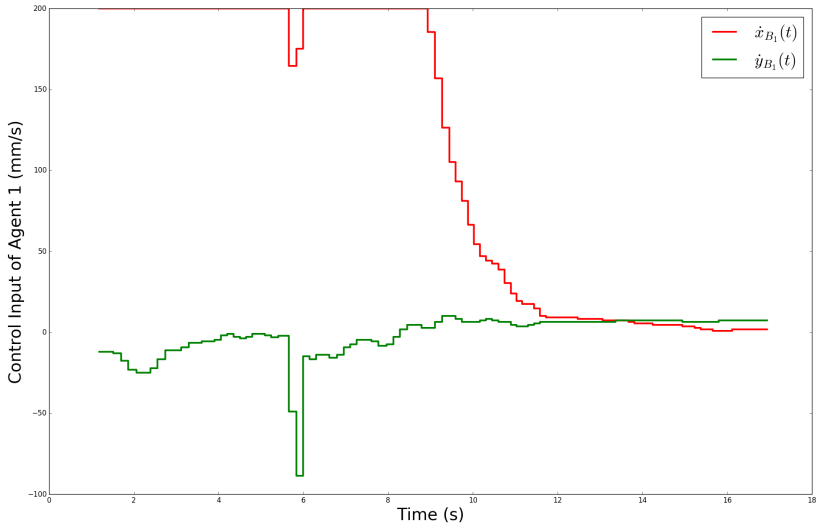


(a) States of object's positions using DNMPC

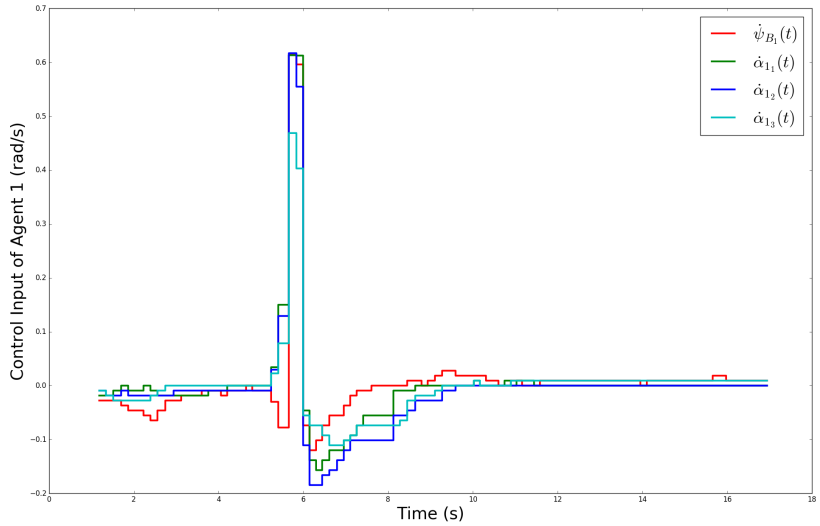


(b) States of object's orientations using DNMPC

Figure 6.25: States of object using DNMPC in experiments

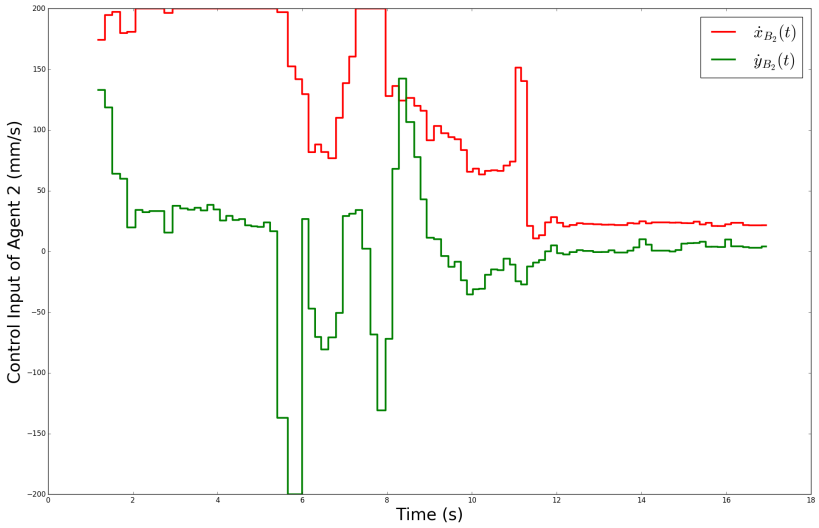


(a) Control inputs of agent 1's positions using DNMPC

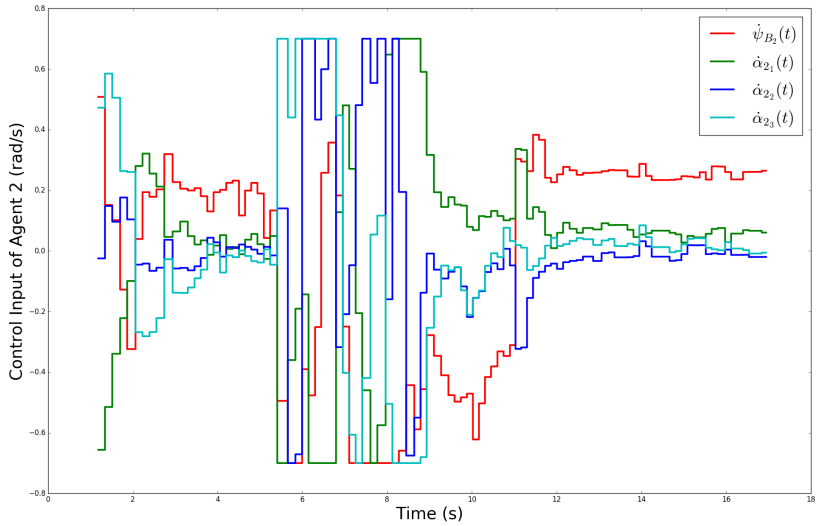


(b) Control inputs of agent 1's orientations using DNMPC

Figure 6.26: Control inputs of agent 1 using DNMPC in experiments



(a) Control inputs of agent 2's positions using DNMPC



(b) Control inputs of agent 2's orientations using DNMPC

Figure 6.27: Control inputs of agent 2 using DNMPC in experiments

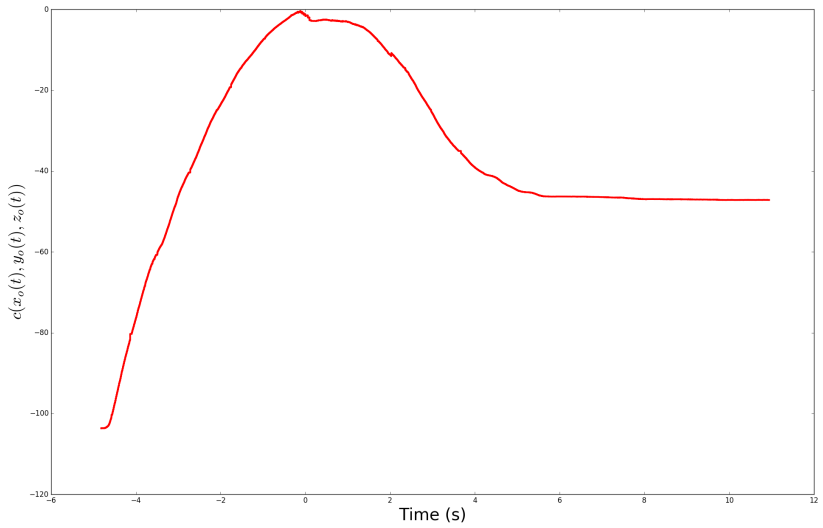


Figure 6.28: Obstacle function using DNMPC in experiments

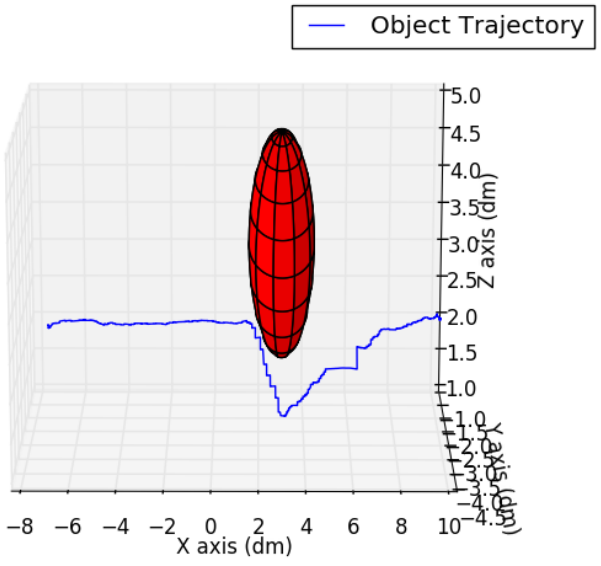


Figure 6.29: The trajectory of the object using DNMPC in experiments

Part IV

Future Work and Conclusion

Chapter 7

Conclusion and Future work

In this thesis, two NMPC schemes, DNMPC and CNMPC, for cooperative transportation of an object rigidly grasped by N agents are proposed. Both DNMPC scheme and CNMPC scheme are able to deal with collision avoidance and singularity avoidance. To reduce computational burden and overall complexity, state feedback linearization and model reduction are used. Feasibility and convergence for proposed DNMPC and CNMPC schemes are proved. Simulation results shows the feasibility and convergence of DNMPC and CNMPC schemes. Real time experiments based on ROS validate the efficiency of both DNMPC and CNMPC. DNMPC is more robust and suitable for experiments.

In future work, the overall complexity of dynamic systems needs to be reduced. Some techniques such as linear time varying MPC may be able to be applied. Communication delay of DNMPC scheme can be investigated. Finally, experiments of dynamic systems needs to be carried out. It might be possible to integrate current experiments with LTL.

Bibliography

[1] Javier Alonso-Mora, Stuart Baker, and Daniela Rus. "Multi-robot formation control and object transport in dynamic environments via constrained optimization". en. In: *The International Journal of Robotics Research* 36.9 (Aug. 2017), pp. 1000–1021. ISSN: 0278-3649. DOI: 10.1177/0278364917719333. URL: https://doi.org/10.1177/0278364917719333.

[2] J. Alonso-Mora et al. "Local motion planning for collaborative multi-robot manipulation of deformable objects". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. May 2015, pp. 5495–5502. DOI: 10.1109/ICRA.2015.7139967.

[3] Joel Andersson, Johan Åkesson, and Moritz Diehl. "CasADi: A Symbolic Package for Automatic Differentiation and Optimal Control". en. In: *Recent Advances in Algorithmic Differentiation*. Lecture Notes in Computational Science and Engineering. Springer, Berlin, Heidelberg, 2012, pp. 297–307. ISBN: 978-3-642-30022-6 978-3-642-30023-3. DOI: 10.1007/978-3-642-30023-3_27. URL: https://link.springer.com/chapter/10.1007/978-3-642-30023-3_27 (visited on 04/13/2018).

[4] A. Bemporad, W. P. M. H. Heemels, and B. De Schutter. "On hybrid systems and closed-loop MPC systems". In: *IEEE Transactions on Automatic Control* 47.5 (May 2002), pp. 863–869. ISSN: 0018-9286. DOI: 10.1109/TAC.2002.1000287.

[5] F. Caccavale, P. Chiacchio, and S. Chiaverini. "Task-space regulation of cooperative manipulators". In: *Automatica* 36.6 (June 2000), pp. 879–887. ISSN: 0005-1098. DOI: 10.1016/S0005-1098(99)00215-0. URL: http://www.sciencedirect.com/science/article/pii/S0005109899002150.

[6] F. Caccavale et al. “Six-DOF Impedance Control of Dual-Arm Cooperative Manipulators”. In: *IEEE/ASME Transactions on Mechatronics* 13.5 (Oct. 2008), pp. 576–586. ISSN: 1083-4435. DOI: 10.1109/TMECH.2008.2002816.

[7] Eduardo F. Camacho and Carlos Bordons. “Nonlinear Model Predictive Control: An Introductory Review”. en. In: *Assessment and Future Directions of Nonlinear Model Predictive Control*. Lecture Notes in Control and Information Sciences. Springer, Berlin, Heidelberg, 2007, pp. 1–16. ISBN: 978-3-540-72698-2 978-3-540-72699-9. DOI: 10.1007/978-3-540-72699-9_1. URL: https://link.springer.com/chapter/10.1007/978-3-540-72699-9_1 (visited on 04/28/2018).

[8] Kathryn G. Chan, Tim Fielding, and Mehran Anvari. “An image-guided automated robot for MRI breast biopsy”. en. In: *The International Journal of Medical Robotics and Computer Assisted Surgery* 12.3 (Sept. 2016), pp. 461–477. ISSN: 1478-596X. DOI: 10.1002/rcs.1760. URL: http://onlinelibrary.wiley.com/doi/10.1002/rcs.1760/abstract.

[9] G. S. Chirikjian and J. W. Burdick. “An obstacle avoidance algorithm for hyper-redundant manipulators”. In: *, IEEE International Conference on Robotics and Automation Proceedings*. May 1990, 625–631 vol.1. DOI: 10.1109/ROBOT.1990.126052.

[10] Matei Ciocarlie et al. “The Velo gripper: A versatile single-actuator design for enveloping, parallel and fingertip grasps”. In: *The International Journal of Robotics Research* 33.5 (Apr. 2014), pp. 753–767. ISSN: 0278-3649. DOI: 10.1177/0278364913519148. URL: https://doi.org/10.1177/0278364913519148.

[11] C. P. Connette et al. “Singularity avoidance for over-actuated, pseudo-omnidirectional, wheeled mobile robots”. In: *2009 IEEE International Conference on Robotics and Automation*. May 2009, pp. 4124–4130. DOI: 10.1109/ROBOT.2009.5152450.

[12] T. W. Danko and P. Y. Oh. “A hyper-redundant manipulator for Mobile Manipulating Unmanned Aerial Vehicles”. In: *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*. May 2013, pp. 974–981. DOI: 10.1109/ICUAS.2013.6564784.

[13] Todd W. Danko and Paul Y. Oh. "Design and Control of a Hyper-Redundant Manipulator for Mobile Manipulating Unmanned Aerial Vehicles". en. In: *Journal of Intelligent & Robotic Systems* 73.1-4 (Jan. 2014), pp. 709–723. ISSN: 0921-0296, 1573-0409. DOI: 10.1007/s10846-013-9935-2. URL: https://link.springer.com/article/10.1007/s10846-013-9935-2.

[14] Steven Diamond and Stephen Boyd. "CVXPY: A Python-Embedded Modeling Language for Convex Optimization". In: *Journal of Machine Learning Research* 17.83 (2016), pp. 1–5.

[15] D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson. "Distributed self-triggered control for multi-agent systems". In: *49th IEEE Conference on Decision and Control (CDC)*. Dec. 2010, pp. 6716–6721. DOI: 10.1109/CDC.2010.5717444.

[16] Hui Dong and Zhijiang Du. "Obstacle Avoidance Path Planning of Planar Redundant Manipulators Using Workspace Density". en. In: *International Journal of Advanced Robotic Systems* 12.2 (Feb. 2015), p. 9. ISSN: 1729-8814. DOI: 10.5772/59973. URL: https://doi.org/10.5772/59973 (visited on 02/19/2018).

[17] S. Erhart and S. Hirche. "Adaptive force/velocity control for multi-robot cooperative manipulation under uncertain kinematic parameters". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Nov. 2013, pp. 307–314. DOI: 10.1109/IROS.2013.6696369.

[18] S. Erhart, D. Sieber, and S. Hirche. "An impedance-based control architecture for multi-robot cooperative dual-arm mobile manipulation". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Nov. 2013, pp. 315–322. DOI: 10.1109/IROS.2013.6696370.

[19] F. Ficuciello et al. "Cartesian impedance control of redundant manipulators for human-robot co-manipulation". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sept. 2014, pp. 2120–2125. DOI: 10.1109/IROS.2014.6942847.

[20] Alexandros Filotheou, Alexandros Nikou, and Dimos V. Dimarogonas. "Decentralized Control of Uncertain Multi-Agent Systems with Connectivity Maintenance and Collision Avoidance". In: *arXiv:1710.09204 [cs]* (Oct. 2017). arXiv: 1710.09204. URL: http://arxiv.org/abs/1710.09204 (visited on 02/20/2018).

[21] Rolf Findeisen et al. “State and Output Feedback Nonlinear Model Predictive Control: An Overview”. In: *European Journal of Control* 9.2 (Jan. 2003), pp. 190–206. ISSN: 0947-3580. DOI: 10.3166/ejc.9.190-206. URL: http://www.sciencedirect.com/science/article/pii/S0947358003702751 (visited on 04/28/2018).

[22] Fernando A. C. C. Fontes. “A general framework to design stabilizing nonlinear model predictive controllers”. In: *Systems & Control Letters* 42.2 (Feb. 2001), pp. 127–143. ISSN: 0167-6911. DOI: 10.1016/S0167-6911(00)00084-0. URL: http://www.sciencedirect.com/science/article/pii/S0167691100000840 (visited on 04/28/2018).

[23] Michael Grant and Stephen Boyd. *CVX: Matlab Software for Disciplined Convex Programming, version 2.1*. http://cvxr.com/cvx. Mar. 2014.

[24] Michael Grant and Stephen Boyd. “Graph implementations for nonsmooth convex programs”. In: *Recent Advances in Learning and Control*. Ed. by V. Blondel, S. Boyd, and H. Kimura. Lecture Notes in Control and Information Sciences. http://stanford.edu/~boyd/graph_dcp.html. Springer-Verlag Limited, 2008, pp. 95–110.

[25] Lars Grüne and Jürgen Pannek. *Nonlinear Model Predictive Control: Theory and Algorithms*. en. Communications and Control Engineering. London: Springer-Verlag, 2011. ISBN: 978-0-85729-500-2. URL: //www.springer.com/us/book/9780857295002 (visited on 02/20/2018).

[26] J. Gudino-Lau et al. “On the control of cooperative robots without velocity measurements”. In: *IEEE Transactions on Control Systems Technology* 12.4 (July 2004), pp. 600–608. ISSN: 1063-6536. DOI: 10.1109/TCST.2004.824965.

[27] W. Gueaieb, F. Karray, and S. Al-Sharhan. “A Robust Hybrid Intelligent Position/Force Control Scheme for Cooperative Manipulators”. In: *IEEE/ASME Transactions on Mechatronics* 12.2 (Apr. 2007), pp. 109–125. ISSN: 1083-4435. DOI: 10.1109/TMECH.2007.892820.

[28] D. Heck et al. “Internal and external force-based impedance control for cooperative manipulation”. In: *2013 European Control Conference (ECC)*. July 2013, pp. 2299–2304.

[29] Carlos Hernandez et al. “Team Delft’s Robot Winner of the Amazon Picking Challenge 2016”. In: *CoRR* abs/1610.05514 (2016). arXiv: 1610.05514. URL: http://arxiv.org/abs/1610.05514.

[30] Liguo Huo and Luc Baron. “The joint-limits and singularity avoidance in robotic welding”. en. In: *Industrial Robot: An International Journal* (Apr. 2013). DOI: 10.1108/01439910810893626. URL: http://www.emeraldinsight.com/doi/full/10.1108/01439910810893626 (visited on 02/20/2018).

[31] Yutaka Inoue, Takahiro Tohge, and Hitoshi Iba. “Cooperative transportation system for humanoid robots using simulation-based learning”. In: *Applied Soft Computing* 7.1 (Jan. 2007), pp. 115–125. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2005.05.001. URL: http://www.sciencedirect.com/science/article/pii/S1568494605000293.

[32] *Jacques Ferber: Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. URL: http://jasss.soc.surrey.ac.uk/4/2/reviews/rouchier.html.

[33] A. E. Jimenez-Cano et al. “Control of an aerial robot with multi-link arm for assembly tasks”. In: *2013 IEEE International Conference on Robotics and Automation*. May 2013, pp. 4916–4921. DOI: 10.1109/ICRA.2013.6631279.

[34] Hassan K. Khalil. *Nonlinear Control*. English. 1 edition. Boston: Pearson, Feb. 2014. ISBN: 978-0-13-349926-1.

[35] O. Khatib et al. “Coordination and decentralized cooperation of multiple mobile manipulators”. en. In: *Journal of Robotic Systems* 13.11 (Nov. 1996), pp. 755–764. ISSN: 1097-4563. DOI: 10.1002/(SICI)1097-4563(199611)13:11<755::AID-ROB6>3.0.CO;2-U. URL: http://onlinelibrary.wiley.com/doi/10.1002/(SICI)1097-4563(199611)13:11%3C755::AID-ROB6%3E3.0.CO;2-U/abstract.

[36] Jinhyun Kim et al. "A general singularity avoidance framework for robot manipulators: task reconstruction method". In: *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*. Vol. 5. Apr. 2004, 4809–4814 Vol.5. DOI: 10.1109/ROBOT.2004.1302479.

[37] A. Leonessa. "Underwater Robots: Motion and Force Control of Vehicle-Manipulator Systems (G. Antonelli; 2006) [Book Review]". In: *IEEE Control Systems* 28.5 (Oct. 2008), pp. 138–139. ISSN: 1066-033X. DOI: 10.1109/MCS.2008.927329.

[38] Yun-Hui Liu, S. Arimoto, and T. Ogasawara. "Decentralized cooperation control: Non-communication object handling". In: *Proceedings of IEEE International Conference on Robotics and Automation*. Vol. 3. Apr. 1996, 2414–2419 vol.3. DOI: 10.1109/ROBOT.1996.506525.

[39] Yun-Hui Liu and Suguru Arimoto. "Decentralized Adaptive and Nonadaptive Position/Force Controllers for Redundant Manipulators in Cooperations". en. In: *The International Journal of Robotics Research* 17.3 (Mar. 1998), pp. 232–247. ISSN: 0278-3649. DOI: 10.1177/027836499801700302. URL: https://doi.org/10.1177/027836499801700302.

[40] J. Lofberg. "YALMIP : a toolbox for modeling and optimization in MATLAB". In: *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*. Sept. 2004, pp. 284–289. DOI: 10.1109/CACSD.2004.1393890.

[41] H. F. Machiel Van der Loos, David J. Reinkensmeyer, and Eugenio Guglielmelli. "Rehabilitation and Health Care Robotics". en. In: *Springer Handbook of Robotics*. DOI: 10.1007/978-3-319-32552-1_64. Springer, Cham, 2016, pp. 1685–1728. ISBN: 978-3-319-32550-7 978-3-319-32552-1. URL: https://link.springer.com/chapter/10.1007/978-3-319-32552-1_64.

[42] Simone Loureiro De Oliveira Kothare and Manfred Morari. "Contractive model predictive control for constrained nonlinear systems". In: *Automatic Control, IEEE Transactions on* 45 (July 2000), pp. 1053–1071. DOI: 10.1109/9.863592.

[43] Jacob Mattingley and Stephen Boyd. "CVXGEN: a code generator for embedded convex optimization". en. In: *Optimization and Engineering* 13.1 (Mar. 2012), pp. 1–27. ISSN: 1389-4420, 1573-2924. DOI: 10.1007/s11081-011-9176-9. URL: https://link.springer.com/article/10.1007/s11081-011-9176-9 (visited on 04/13/2018).

[44] A. Motahari, H. Zohoor, and M. Habibnejad Korayem. "A new obstacle avoidance method for discretely actuated hyper-redundant manipulators". In: *Scientia Iranica* 19.4 (Aug. 2012), pp. 1081–1091. ISSN: 1026-3098. DOI: 10.1016/j.scient.2012.06.017. URL: http://www.sciencedirect.com/science/article/pii/S1026309812001381 (visited on 02/19/2018).

[45] Alexandros Nikou et al. "A Nonlinear Model Predictive Control Scheme for Cooperative Manipulation with Singularity and Collision Avoidance". In: *arXiv:1705.01426 [cs]* (May 2017). arXiv: 1705.01426. URL: http://arxiv.org/abs/1705.01426.

[46] R. Olfati-Saber, J. A. Fax, and R. M. Murray. "Consensus and Cooperation in Networked Multi-Agent Systems". In: *Proceedings of the IEEE* 95.1 (Jan. 2007), pp. 215–233. ISSN: 0018-9219. DOI: 10.1109/JPROC.2006.887293.

[47] Eugénio Oliveira, Klaus Fischer, and Olga Stepankova. "Multi-agent systems: which research for which applications". In: *Robotics and Autonomous Systems*. Multi-Agent Systems Applications 27.1 (Apr. 1999), pp. 91–106. ISSN: 0921-8890. DOI: 10.1016/S0921-8890(98)00085-2. URL: http://www.sciencedirect.com/science/article/pii/S0921889098000852.

[48] A. N. Ponce-Hinestroza et al. "Cooperative redundant omnidirectional mobile manipulators: Model-free decentralized integral sliding modes and passive velocity fields". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. May 2016, pp. 2375–2380. DOI: 10.1109/ICRA.2016.7487387.

[49] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA Workshop on Open Source Software*. 2009.

[50] James B. Rawlings, David Q. Mayne, and Moritz M. Diehl. *Model Predictive Control: Theory, Computation, and Design, 2nd Edition*. 2nd edition. Madison, Wis: Nob Hill Publishing, LLC, Oct. 2017. ISBN: 978-0-9759377-3-0.

[51] Linda Resnik, Shana L. Klinger, and Katherine Etter. "The DEKA Arm: Its features, functionality, and evolution during the Veterans Affairs Study to optimize the DEKA Arm". en. In: *Prosthetics and Orthotics International* 38.6 (Dec. 2014), pp. 492–504. ISSN: 0309-3646. DOI: 10.1177/0309364613506913. URL: https://doi.org/10.1177/0309364613506913.

[52] S. A. Schneider and R. H. Cannon. "Object impedance control for cooperative manipulation: theory and experimental results". In: *IEEE Transactions on Robotics and Automation* 8.3 (June 1992), pp. 383–394. ISSN: 1042-296X. DOI: 10.1109/70.143355.

[53] Jeff S. Shamma, ed. *Cooperative Control of Distributed Multi-Agent Systems: Shamma/Cooperative Control of Distributed Multi-Agent Systems*. DOI: 10.1002/9780470724200. Chichester, UK: John Wiley & Sons, Ltd, Nov. 2007. ISBN: 978-0-470-72420-0 978-0-470-06031-5. URL: http://doi.wiley.com/10.1002/9780470724200 (visited on 02/16/2018).

[54] Bruno Siciliano et al. *Robotics: Modelling, Planning and Control*. en. Advanced Textbooks in Control and Signal Processing. London: Springer-Verlag, 2009. ISBN: 978-1-84628-641-4. URL: //www.springer.com/us/book/9781846286414 (visited on 02/20/2018).

[55] D. Simon, J. Löfberg, and T. Glad. "Nonlinear model predictive control using Feedback Linearization and local inner convex constraint approximations". In: *2013 European Control Conference (ECC)*. July 2013, pp. 2056–2061.

[56] Jérôme Szewczyk, Frédéric Plumet, and Philippe Bidaud. "Planning and controlling cooperating robots through distributed impedance". en. In: *Journal of Robotic Systems* 19.6 (June 2002), pp. 283–297. ISSN: 1097-4563. DOI: 10.1002/rob.10041. URL: http://onlinelibrary.wiley.com/doi/10.1002/rob.10041/abstract.

[57] Lei Tang et al. "A novel potential field method for obstacle avoidance and path planning of mobile robot". In: *2010 3rd International Conference on Computer Science and Information Technology*. Vol. 9. July 2010, pp. 633–637. DOI: 10.1109/ICCSIT.2010.5565069.

[58] "The graphic: Curiosity Mars Rover". en. In: *Engineering & Technology* 7.8 (Sept. 2012), pp. 12–12. ISSN: 1750-9645. DOI: 10.1049/et.2012.0823. URL: http://digital-library.theiet.org/content/journals/10.1049/et.2012.0823.

[59] A. Tsiamis et al. "Cooperative manipulation exploiting only implicit communication". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2015, pp. 864–869. DOI: 10.1109/IROS.2015.7353473.

[60] Christos K. Verginis, Alexandros Nikou, and Dimos V. Dimarogonas. "Communication-based Decentralized Cooperative Object Transportation Using Nonlinear Model Predictive Control". In: *arXiv:1803.07940 [cs]* (Mar. 2018). arXiv: 1803.07940. URL: http://arxiv.org/abs/1803.07940 (visited on 04/28/2018).

[61] Samer Yahya, M. Moghavvemi, and Haider A. F. Mohamed. "Singularity avoidance of a six degree of freedom three dimensional redundant planar manipulator". In: *Computers & Mathematics with Applications*. Advanced Technologies in Computer, Consumer and Control 64.5 (Sept. 2012), pp. 856–868. ISSN: 0898-1221. DOI: 10.1016/j.camwa.2011.12.073. URL: http://www.sciencedirect.com/science/article/pii/S0898122111011448 (visited on 02/20/2018).

[62] Qiushi Zhang, Dandan Chen, and Ting Chen. "An Obstacle Avoidance Method of Soccer Robot Based on Evolutionary Artificial Potential Field". In: *Energy Procedia*. 2012 International Conference on Future Energy, Environment, and Materials 16 (Jan. 2012), pp. 1792–1798. ISSN: 1876-6102. DOI: 10.1016/j.egypro.2012.01.276. URL: http://www.sciencedirect.com/science/article/pii/S187661021200286X (visited on 02/19/2018).

[63] M. Zribi and S. Ahmad. "Adaptive control for multiple cooperative robot arms". In: *[1992] Proceedings of the 31st IEEE Conference on Decision and Control*. 1992, 1392–1398 vol.2. DOI: 10.1109/CDC.1992.371481.

TRITA TRITA-EECS-EX-2018:105
ISSN 1653-5146