Politecnico di Torino

Department of Control and Computer Engineering
(DAUIN)

Master Degree
in
Mechatronics Engineering

Implementation of a Decentralized Parameter Estimation Strategy for Unknown Loads Through an Over-actuated Robotic Platform

Supervisor:
Prof. Alessandro Rizzo
Company Tutor:
Prof. Antonio Franchi

Author:
Hosameldin Awadalla Omer Mohamed

April 2018

Acknowledgment

First of all, I would like to thank my family for always being caring, understanding and supporting me throughout my life.

I am deeply grateful to my thesis supervisors Prof. Antonio Franchi and Prof. Alessandro Rizzo. In spite of their great responsibilities, they managed to spare some of their valuable time to advise and guide me throughout this work. Moreover, for responding promptly to my requests about administrative procedures at LAAS and the Politecnico.

I would also like to express my sincere gratitude to Dr. Michele Furci, a former post-doc researcher at LAAS-CNRS for the help and support I received from him throughout this journey. Specially his and input in programming and operating the robotic platform. I also thank Eng. Hermes Tello-Chavez for providing me with pictures of the platform after I left LAAS.

My deep appreciation goes to Antonio Franchi's group at LAAS, for being very kind and welcoming and for their valuable help and advices. Also to my friends in Turin, specially Osman Abdalla, Talal Almutaz Abdalla and Dionysia Varvarigou for Their sustainable support and encouragement.

Abstract

The thesis deals with the cooperative manipulation control of a load on a plane by a team of mobile robots moving in 2-D. The work is based on an algorithm, developed by the supervisors and co-workers, which consists of a decentralized estimation of the parameters and the twist of the load utilizing only local measurements of velocities of contact points and a decentralized controller of the load's linear and angular velocities.

A rolling platform actuated by rotors to move on a plane, was used to validate the above mentioned decentralized estimation strategy. It can apply forces, whose magnitudes are determined by the rotational speeds of the rotors and are dictated by angles of rotation of servo-motors, to which the rotors are attached.

The platform belongs to the Laboratory of Analysis and Architecture of Systems (**LAAS**), (a **CNRS** research unit "Centre Nationnal de la Recherche Scientifique"), where the experiments were conducted.

The hardware of the Platform consists of an on-board computer (ODROID-XU4) running on Linux to execute a high-level control algorithm (taking commands from the joystick, applying a force trajectory to the actuators, recording the inputs and various measurements of the platforms in a text file, and other tasks), a flight controller (Mikrokopter) that contains an Inertial Measurement Unit (IMU), an Arduino micro-controller to control Servo-motors angles, the rotors brush-less controllers and voltage regulators.

The Platform utilizes OptiTrack Motion Capture System that is installed at LAAS infrastructure, moreover, it exchanges information through the network using a WiFi module and operates with a LiPo battery.

The software utilizes the architecture and some modules of a software package developed at LAAS known as **Openrobots**, and the collected data were applied to the algorithm in MATLAB/SIMULINK environment.

The estimation algorithm consists two cascaded phases, a kinematic phase that takes measurements of velocities of contact points and uses rigid body kinematics, and a dynamic phase that uses the outcomes of the kinematic part in addition to force inputs, utilizing rigid body physics and dynamics. The two phases contain cascaded stages of computation, linear parameter-estimation, non-linear state-estimation and dynamic average consensus blocks.

The performance of the estimation algorithm was analyzed, and the algorithm was tested against the uncertainties emerging from the conducted experiments, such as the noise in measurements, the friction in the wheels and the accumulation of errors and uncertainties of the cascaded stages. The implemented algorithm exhibit satisfactory performance in terms of efficiency and conceptual flow. However, we observed that it is very sensitive to model uncertainties. Even though some sources of disturbances have been modeled, additional work is necessary to include more accurate disturbance models toward performing accurate estimations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the topic of the thesis by defining basic concepts as a preface. In particular it defines robots Cooperative Manipulation which is an important field in robotics, then addresses adding a manipulator to mobile robotics and its usefulness and at the same time its additional requirements on the control algorithm, specifically, distribution of the control law. Then introduces an algorithm in which the thesis is interested in. After, it explains the purpose and the methodology of the thesis. Following a brief description about LAAS-CNRS, where the thesis activities are conducted.

What is Cooperative Manipulation?

The term cooperative implies the willingness and ability to work with others [1]. Cooperative work means the joint work of the cooperators to achieve a common task or goal. The task can be controlling the motion of an object.

Manipulating an object is performed with the aim of changing the space position or orientation of object, tracking a given trajectory of it. If the manipulation is performed in cooperation, each cooperator performs its own work, taking into account the state of the other participants in the cooperation besides the manipulated object itself. In general, the cooperative system consists of multiple stages which are: planning of the approach, approaching to the object, grasping, gripping, lifting, **transferring**, lowering, releasing and withdrawing. The topic of the thesis is related to the transferring phase, which consists of moving the object along a predetermined trajectory at a predetermined orientation. The manipulators move in such a way as to force the object to satisfy the preset motion requirements and/or produce the required gripping loads. [3]

Using multiple robots to perform a task has numerous advantages over a single robot, for instance, it enables complex assembly tasks. Moreover, it allows manipulation of heavy and/or large objects that are far beyond the capabilities of one robot. In addition, it enables dexterous manipulation of objects (The dexterity can be defined as the capability of changing the position and orientation of the manipulated object from a given reference configuration to a different one, arbitrarily chosen within the hand workspace [4]).

[1] www.merriam-webster.com

Adding Manipulators to Mobile Robots

At the moment mobile manipulation is a subject of major focus in development and research environments. Such systems combine the advantages of mobile platforms and robotic manipulator arms and reduce their drawbacks. For instance, the mobile platform extends the workspace of the arm, and the arm adds dexterity to the mobile platform.

Mobile manipulators, either autonomous or tele-operated, are used in many areas, e.g. space exploration, military operations, home-care and health-care. However, within the industrial field the implementation of mobile manipulators has been limited, although the needs for intelligent and flexible automation are present. In addition, the necessary technology entities (mobile platforms, robot manipulators, vision and tooling) are, to a large extent, available off-the-shelf components [5].

Control Architecture: Centralized VS Decentralized Cooperation

Much research has been done for the motion control of multiple robots manipulating an object. However, most of the control algorithms proposed so far are designed based on the centralized control system; that adopts a single controller to control all of the robots in a centralized way based on the global information. The centralized control system may be effective in case of the coordinated motion control of fixed manipulators since the number of the manipulators in coordination is usually limited to two or three [6].

The use of mobile manipulators in the real world is more challenging than fixed manipulators, considering the case where such mobile manipulators transport a single object in coordination, a single controller could not control a large number of robots because of the real-time communication problem around the robots and the computational burden of the single controller. Hence centralized control is no more realistic to control a large number of mobile manipulators.

In addition, a mobile manipulator has slippage between its wheels and the ground and we could not position the mobile manipulator precisely. Therefore, we could not apply the same control principle of manipulators for controlling the multiple mobile manipulators, and the control system of the mobile manipulators has to be redesigned robust against the inevitable positioning error of each mobile manipulator.

Most of these control algorithms proposed so far have been designed under the assumption that the geometric relations among the robots are known precisely. However, it is not easy to know the geometric relations among them precisely, especially when the robots handle an unknown object in coordination in a real environment, besides Mobile manipulator have slippage between its wheels and the ground and we could not position the mobile manipulator precisely [6].

Considering a network of robots performing a control law. The control law is considered decentralized if, for each robot i, the size of the communication bandwidth, the computation time (per

step) and the memory usage (storage of inputs, outputs and local variables) depend only on the number of communication neighbors ($|\mathcal{N}_i|$) and **not** on the number of robots (n). This property of decentralized control makes them scalable [7].

The De-centralized Algorithm Under Study

The field of cooperative manipulation with multiple mobile robots has been deeply studied in the past (see the next chapter 2). However in this thesis we are interested in a particular solution, which is a recent work of (Franchi, Petitti, and Rizzo 2014 [8] ; Franchi, Petitti, and Rizzo 2015 [2]), in which the authors studied and solved the problem of estimating in a fully distributed way all the inertial parameters of an unknown load, and in (Petitti et al. 2016) [1], where the authors have developed a robust decentralized controller for the cooperative manipulation with a distributed team of networked mobile manipulators, based on the estimation method mentioned above can be used to control the motion of the unknown load. Figure 1.1 explains the idea. The algorithm has the following characteristics:

- All the parameters of the grasping, and the full state of the load are considered unknown to the robots.
- Each robot only knows the local force it applies to the load, is able to measure the linear velocity of its contact point with the load and to communicate only with its neighbors through a wireless communication network.
- The algorithm is designed in 2-dimensional Cartesian space.
- The algorithm controls the linear and angular velocities of the load to follow a desired trajectory.

The idea of the decentralized control algorithm is further explained with the block diagram in figure 1.2.

So far the algorithm was tested only through numerical simulation, and has not yet been experimentally validated. This thesis is interested in the experimental validation of the Estimation part in Decentralized Control algorithm

The estimation algorithm is implemented in each robot to obtain locally the load's the dynamical parameters; its mass and inertia, the load's linear velocity in World frame, the angular velocity (a scalar value in 2D), and each robot obtain the distance of its contact point from the Center of Mass of the load. All the estimated values are essential for the proposed control algorithm.

The inputs to the estimation algorithm are the applied force in World frame, a measurement of the linear velocity of the contact point. In addition, the algorithm exchanges some signals in a communication network with the neighboring robots. Figure 1.3 shows the inputs and outputs of the estimation algorithm of one robot i. Thus the algorithm is considered decentralized because no need for a central node to provide measurements of the object's parameters and state.

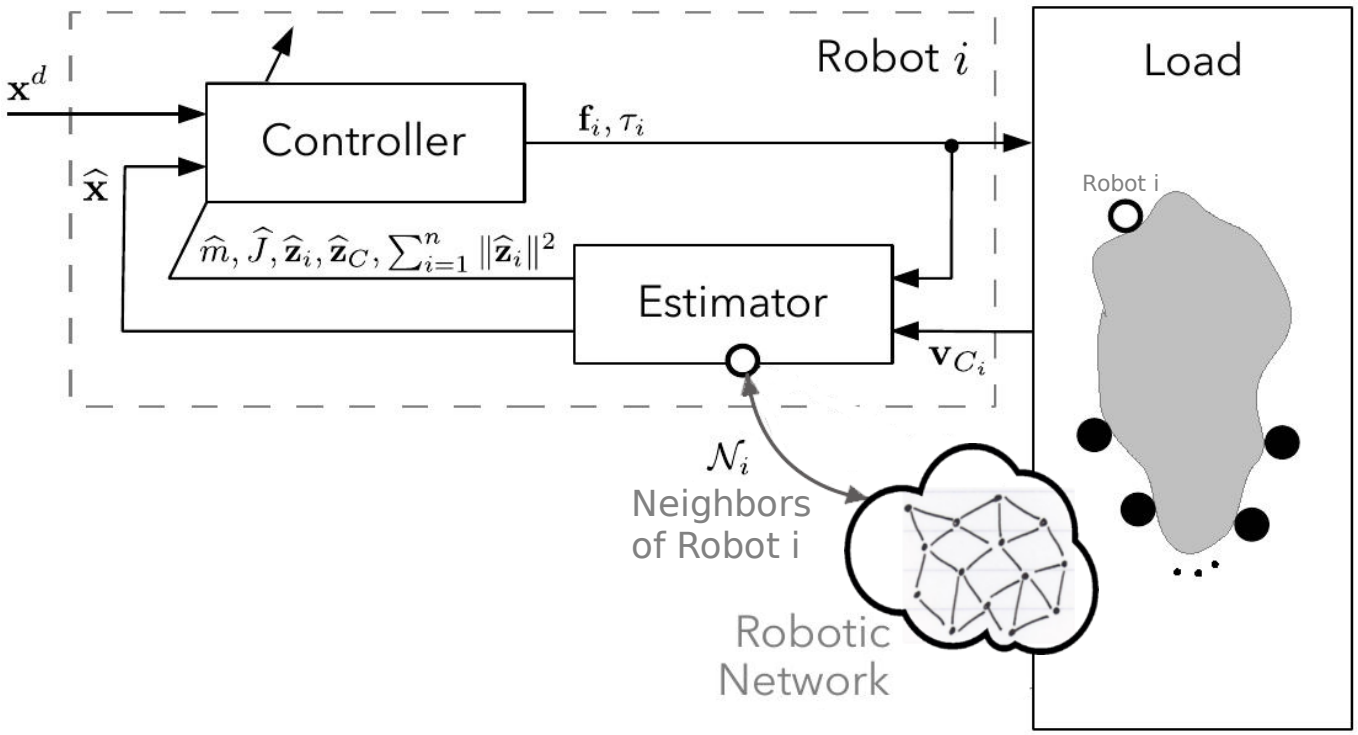


Figure 1.2: Fully-decentralized cooperative manipulation with unknown load parameter, unknown grasping parameters and unknown load state. each robot excerts a force $\mathbf{f}_i$ and a torque τ_i, can measure the linear velocity of its contact point $\mathbf{v}_{ci}$, and is able to obtain some information from the neighboring robots through wireless communication

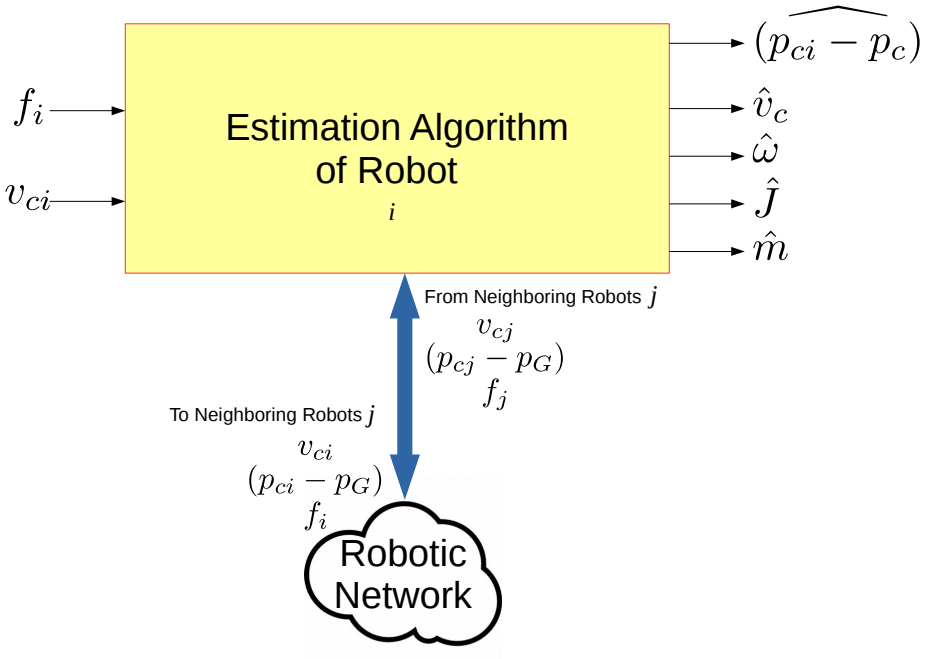


Figure 1.3: The inputs and outputs of the estimation algorithm. $\mathbf{f}_i$ is the force applied by robot i, $\mathbf{v}_{ci}$ is the linear velocity of contact point of robot i and $\mathbf{p}_{ci}$ is its position, $\mathbf{p}_c$ is the CoM position of the load, $\mathbf{p}_G$ is the geomentrical center of contact points, m and J are the load's mass and inertia respectively, and ω is the angular velocity of the load

The estimation algorithm consists two cascaded phases, a kinematic phase that takes measurements of velocities of contact points and uses rigid body kinematics, and a dynamic phase that uses the outcomes of the Kinematic part in addition to force inputs, utilizing rigid body physics and dynamics. The two phases contain cascaded stages of computation, linear parameter-estimation, non-linear state-estimation and dynamic average consensus blocks.

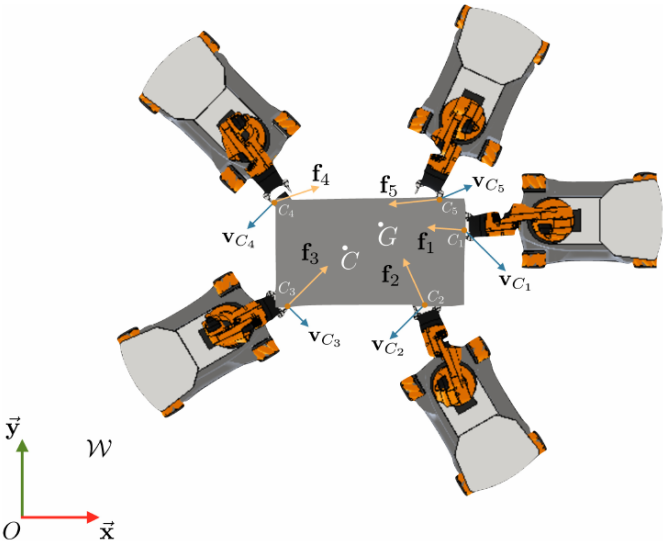


Figure 1.1: A group of KUKA mobile manipulators cooperatively moving a rigid body, where C_i $\mathbf{f}_i$ and $\mathbf{v}_{ci}$ $(i = 1, 2, ..., 5)$ are the location of the contact point of robot i, the force applied by robot i, and the linear velocity of the contact point, respectively. [2]

What is the Purpose of the Thesis?

The main purpose of this thesis is to test the estimation algorithm on a robotic platform to assess its ability to provide estimates that can be utilized for successful decentralized control, to identify its weak points and try to improve its performance.

Objectives

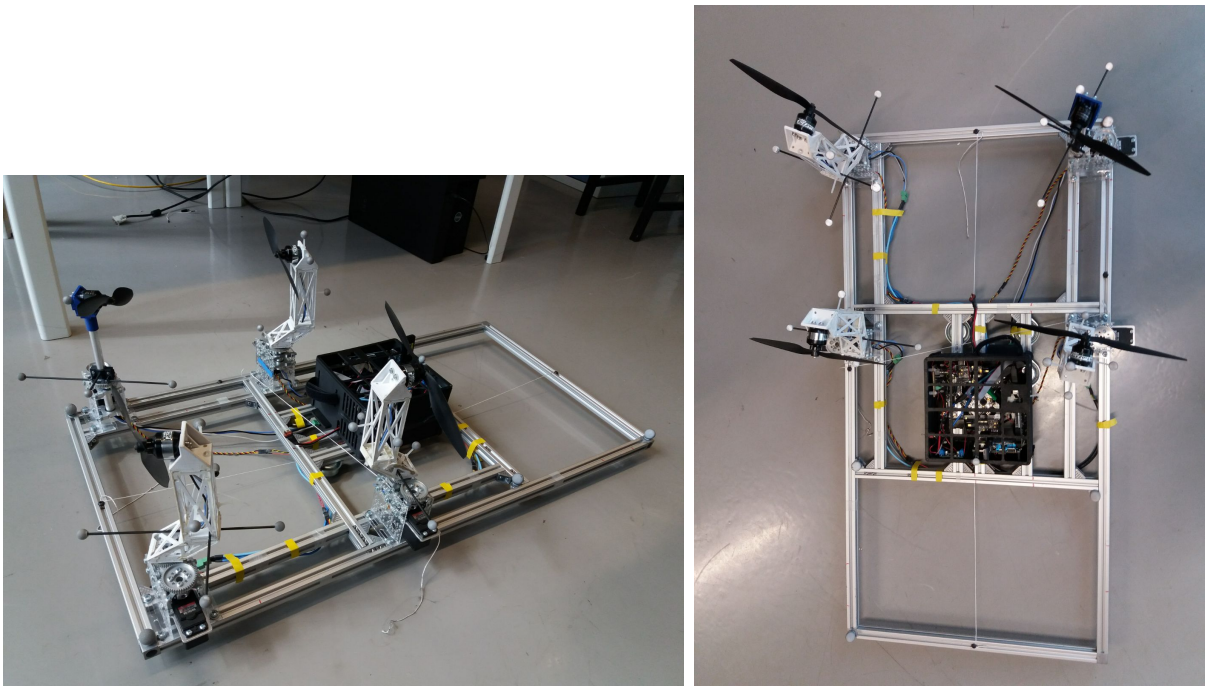
My internship assignment was to experimentally validate the methods developed by my supervisor and his colleagues. The main targets of the assignment are as follows:

- To study the methods used in the estimation algorithm and build the required background.
- To implement the estimation algorithm and assess its performance using the data collected from the experiments.
- improvement of the algorithm based on experimental assessment.

How is the De-centralized algorithm validated?

Due to the unavailability of mobile manipulators another solution is adopted, the use of a rotor-actuated platform to simulate the function of multiple mobile manipulators and the load too. The platform is called (ROSPO) and is shown in figure 1.4 and 1.4

The ROtor-graSPing Omnidirectional (ROSPO) platform, is a platform designed and manufactured at LAAS-CNRS to test control algorithms. ROSPO is used in this thesis to validate the



(a) Perspective View (b) Top View

Figure 1.4: The ROSPO platform

decentralized estimation algorithm. It consists of a frame with four modules composed by a **tilting turret** and a **motor-propeller actuator**. As compared to an unmanned aerial vehicle (UAV) , this platform is simplified because its dynamics is constrained to a 2D plane, having thus only 3 degrees of freedom. The platform is able to slide on the floor by 4 ball-bearing omni-directional wheels. The wheels are shown in figure 1.5. Each tilting turret consists of a motor and a propeller, and servo-motor to tilt them to a desired angle, a zoomed view of the actuator is shown in figure 1.6.

The idea is to simulate each mobile manipulator with a tilting turret, and the whole platform as the common load, which means that we are simulating a system of four mobile manipulators rigidly grasping a common load. Moreover, the ROSPO is equipped with an IMU (Inertial Measurement Unit) and a Motion Capture system (section 3.1), and it is possible to utilize motors speed measurements to compute the applied force using a speed-thrust model, all the inputs and the outputs and the internal signals in the estimation algorithm can be either measured or directly derived from a measurement, as will be demonstrated in chapter 3.

It is important to note that the estimation algorithms of the robots are implemented in a single on-board computer that controls all the four actuators, however the software is designed to be 'conceptually' distributed in order to simulate the software in each robot and the communication network, the idea is illustrated in figure 1.7. This fact is expected not to affect the integrity of the assessment, of course in this case the architecture assumes instantaneous communication between the robots and we move on with this assumption.

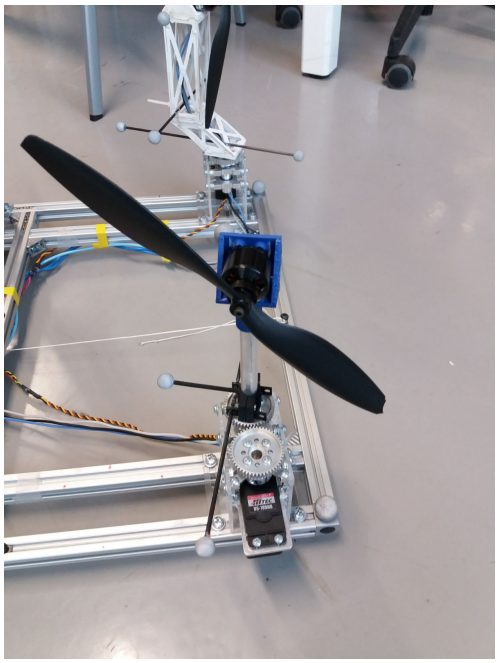


Figure 1.6: One turret in ROSPO

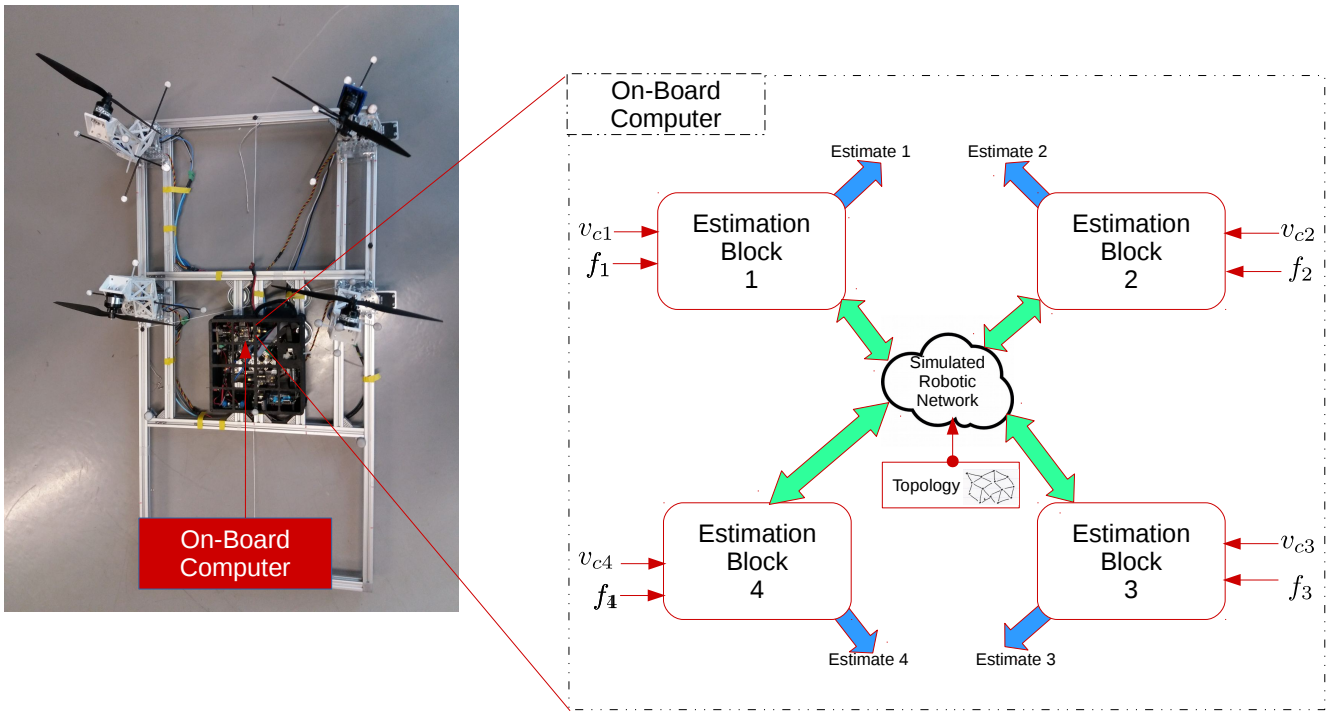


Figure 1.7: The ROSPO has a single on-board computer, but the software makes the turrets (actuators) 'conceptually' distributed

A brief Introduction to the Company

The CNRS "Centre Nationale de Recherche Scientifique" is a very well known French research center. My internship took place in the LAAS, a subdivision of the CNRS, which I will introduce in this section.

Figure 1.5: The ball bearing omni-directional wheels used to enable the rospo to move

CNRS

The CNRS is the French largest governmental research organization. It was created during the second world war (October 19th, 1939) in order to gather all the state scientific organization (specialized, pure or applied science) and to coordinate them at a greater scale.
Then it has risen over the years until being as known as it is today. Its governance is made by Alain Fuchs (President of the CNRS). The Research Center has a budget of 3309.13 Meuro which is mainly subsided by the government. The labour force is formed with 31944 people with 11106 permanent researchers, 13511 engineers and technicians and 7327 temporary researchers or students. In 2015, the CNRS hired 300 researchers and 300 engineers and technicians. The center is also part of the Top 100 Global Innovators since 2011 and has been ranked fifth in the new Top 100 Thomson Reuters. Moreover, more than 43000 papers have been published last year and 60 percent of them are related to a foreign laboratory. Besides the CNRS has strong links and several partnerships with higher education and companies

The CNRS is composed of 10 different institutes, among all them, the "Institute for Engineering and Systems Sciences" and the "Institute of Information Sciences and their interactions" are the ones the LAAS is part of.

LAAS

The "Laboratoire d'Analyse et d'Architecture des Systemes" was created in 1968 by Jean Lagasse. The first goal of this research unit was to build a laboratory of automatics and its spatial applications which was, at this time, a tremendous international concern. From then, the notion of "system" was more appropriated for the research which were being conducted, over the years the LAAS has become what it is today.
LAAS Scientific research is distributed into eight main departments: Critical Information Processing,

Networks and Communications, Robotics, Decision and Optimization, MicroNanosystems RF and Optics, Nano-Engineering and Integration, Energy Management and MicroNanoBio Technologies.

I did my internship in the Robotic department.

Robotic Department

The Robotics (ROB) theme area conducts research along several axes involving perception, motion, manipulation, decision-making, communication and interaction between the robot and its environment: the other robots, humans and ambient intelligence systems.
The main research topics are:

- Environment perception and modeling
- Navigation, localization, motion planning and control
- Manipulation planning and control
- Natural, artificial and virtual motion
- Autonomous decision making, temporal planning, learning
- Control architectures, embedded systems, robustness and fault tolerance
- Human-robot multi-modal and decisions interaction
- Multi-robot cooperation

Antonio Franchi's Group

My supervisor at LAAS was Dr. Antonio Franchi, a Permanent Researcher at LAAS. Obtained Professional Habilitation (HDR) from Toulouse Institute of Technology in 2016. In 2010, He obtained his PhD in "Analysis and Control of Dynamical Systems" at "La Sapienza" university in Rome. He is also an IEEE Senior Member and is an associated editor of the IEEE Transactions on Robotics, and a co-chair of the IEEE RAS Technical Committee on Multi-Robot Systems. His research interests include Multi-robot systems, Human/robot bilateral shared control, Human-in-the-loop, Swarm Teleportation, Cooperative Control and Estimation, Communication, Synchronization, Path Planning, Distributed Algorithms, Exploration, Localization, Pursuit-evasion, Patrolling, Calibration. [2]

Thesis Organization

The Thesis comprise of the following chapters (apart from the Introduction chapter):

[2] Antonio Franchi's homepage:`http://homepages.laas.fr/afranchi/robotics/?q=node/1`

- Literature Review: we discuss the recent research about decentralized cooperative manipulation, and a description about the theory behind the decentralized estimation algorithm under test.
- Experimental Setup: we describe the hardware components and the adopted software architecture in the experiment, followed by a description of the steps undertaken to perform the experiment, and to provide the signals needed for the implementation and validation.
- Kinematic Phase: we go through the theory and implementation of the kinematic phase stages in detail, showing the obtained results and the discussions.
- Dynamic Phase: we describe the theory and implementation of the dynamic phase stages in detail, showing the obtained results and the discussions.
- Conclusion and Future Work: we summarize the main outcomes of this thesis work and possible recommendations for further work.

Chapter 2

Literature Review

Overview

This chapter starts with briefly presenting recent and related work on decentralized cooperative estimation and manipulation, highlighting the mean characteristics. Then focuses on the decentralized estimation and cooperative control algorithm under study, starting with the mathematical representation of a rigid body being grasped by multiple manipulators, going through the proposed decentralized control low and finally the mathematical formulation of the estimation problem.

Related Work

In [9] a decentralized control structure was developed in order to extend operational space methodologies for fixed-base manipulators to mobile manipulation systems. used the concepts of **virtual** linkage and the augmented object to solve the coordination problem between manipulators. The augmented object model is obtained by combining the dynamics of the individual mobile manipulators and the object, while the virtual linkage is a way to model the internal forces associated with multi-grasp manipulation. In this method, the needed measurements are the forces at the grasp points, however, accurate knowledge about the load parameters is required.

In quite a recent work in [10], fully decentralized and recursive approach of online identification of unknown kinematic and dynamic parameters for cooperative manipulation of a rigid body. Each robot must able to measure the linear and angular velocities at its contact point, the acceleration at the contact point and the force and torque applied by the robot to the rigid body. The estimated quantities are: rigid body transformation, the object mass, the center of mass and the inertia tensor in sensor frame. The identification is addressed in 3D and can be used for adaptive control.

On another framework, some works have addressed the decentralized cooperative manipulation by adopting a leader follower scenario. one is by [11]. they have designed a leader-follower formation in which the leader is aware of the object's desired trajectory and implements it via an impedance control law, and the follower estimates it by observing the object's motion and imposes a similar impedance law. the algorithm was designed for 3D and it adopts **load sharing** among the robots according to their specific payload capabilities. In this method, both agents use solely their own force, position and velocity measurements.However, the geometric and inertial parameters of the mobile manipulators as well as of the object are considered known.

A similar idea is in the work by [12], which is known as Force-Amplifying N-Robot Transport

System **(Force-ANTS)**, where the leader applies a small force to guide the rest of the group, and the follower robots amplify the effect of the force by synchronizing their forces to the leader's force, this is done by locally measuring the motion of the object at their attachment points. No explicit communication among the robots is required, instead, The robots instead use sensors that measure the motion of the object itself to coordinate their actions. The leader (either a robot or a human) is the only one who knows the desired trajectory. The leader steers the group towards the destination by adjusting its own force, while the follower robots reinforce the leader's intention by aligning their forces with the leader's. This method is suitable for moving an object that is too heavy for a single robot, and is designed for planar motion (2D). However each robot must know the object mass, the number of robots, the kinetic (static) and viscous friction coefficients.

In [13] a decentralized cooperative manipulation methodology for mobile manipulators using quaternions to represent the object orientation was developed. No measurement of the contact force/torques is required in this method, in addition, the method enable load sharing among the robot arms, for different power capabilities. However each agent must know object's dynamic and kinematic parameters, on another hand they have developed an adaptive version where the agents' and the object dynamic parameters are considered unknown.

Another related work is the work of [14] in which the authors developed a Distributed cooperative parameter estimation and manipulation of an unknown object. The robots cooperatively estimate the object's kinematic and dynamic parameters by properly moving the object or by applying a proper sequence of wrenches in such a way to determine a parameter of the load at each step. The algorithm is developed for mobile manipulators in 3D. No explicit communication is required among the robots. the only information they are assumed to share is the cardinality N of the team. The estimated parameters are used in a distributed cooperative algorithm aimed at controlling the object pose

Another algorithm where the object is assumed to be unknown is the work by [15] which have obtained experimental results on a successful transportation by using multiple custom-made aerial manipulators (hexa-copters), with 2 Degrees-of-Freedom arms. The algorithm implements an online parameter estimation to obtain the parameters of the common payload such as mass and moment of inertia, same as the above method, without the need of multi-axis force/torque sensors.

The authors in [16] were able to design a decentralized adaptive controller that allows teams of agents to perform collaborative manipulation without communication or prior knowledge of object parameters, the proposed algorithm can work in 2D or 3D, but at the moment it was experimentally validated in 2D using planar robots.

The method adopts Adaptive Model Reference Control strategy and Lyapunov method to prove the stability of the controller.

The method assumes that each agent can measure the linear velocity of the center of mass, and the angular velocity of the object. Moreover each agent must be able to determine its orientation with respect to the body frame.

The authors in [17] have developed a distributed cooperation control to grasp and transport

an object. The method takes into account time delays and switching network typologies. The method takes into account the redundancy of the manipulators for decoupling the task and null-space motion, the mobile bases can be controlled in the null-space to maintain a formation, achieve velocity consensus, and ensure collision avoidance, while the motion of the end effector was considered to be the primary task of each manipulator.

Force measurements at the end-effectors are necessary for the algorithm. The work was validated via simulation on a network of three non-holonomic mobile manipulators.

Joel M. Esposito in [18] developed a decentralized Cooperative Manipulation with a Swarm of Mobile Robots. The method uses an online consensus estimate of the swarm's configuration. However the knowledge of object mass and inertia, moreover, the desired acceleration is essential for the control to follow a variable velocity trajectory.

The results were verified by simulation.

Alessandro Marino in [19] uses the results on distributed control of Lagrangian systems to cooperative manipulation of a rigid body tightly grasped by manipulators mounted on mobile platforms.

The method exploits consensus theory to distributively estimate the full state of the system and the object dynamics to estimate the squeezing wrenches, and a local adaptive control law. The local motion of the robot and the Force/Torque measurements are needed to be available by each robot.

The estimation algorithm problem formulation

The following section reviews in detail the decentralized estimation algorithm proposed by, Antonio Franchi, Antonio Petitti, and Alessandro Rizzo (see [2], and [1] with Donato di Paola for the proposed control algorithm).

Robots Modeling

n mobile manipulators on a plane, can be modeled as an undirected graph communication graph $\mathcal{G} = \{\mathcal{I}, \mathcal{E}\}$ which is undirected. Moreover we assume that $\mathcal{G}$ is connected. The neighbors of a robot i are defined as $\mathcal{N}_i = \{j \in \mathcal{I} : (i, j) \in \mathcal{E}\}$.

Each robot i, is able to: exert a force $f_i \in \mathbb{R}^2$, through a contact points $C_i \in B$, and assumed to produce null torque. In addition, each robot i measures only the velocity $mat\dot{h}bfp_{ci}$ of C_i.

The load B, subject to the forces $f_1, f_2, ..., f_n$, can be described by the following differential equations (see grasping chapter in [20]):

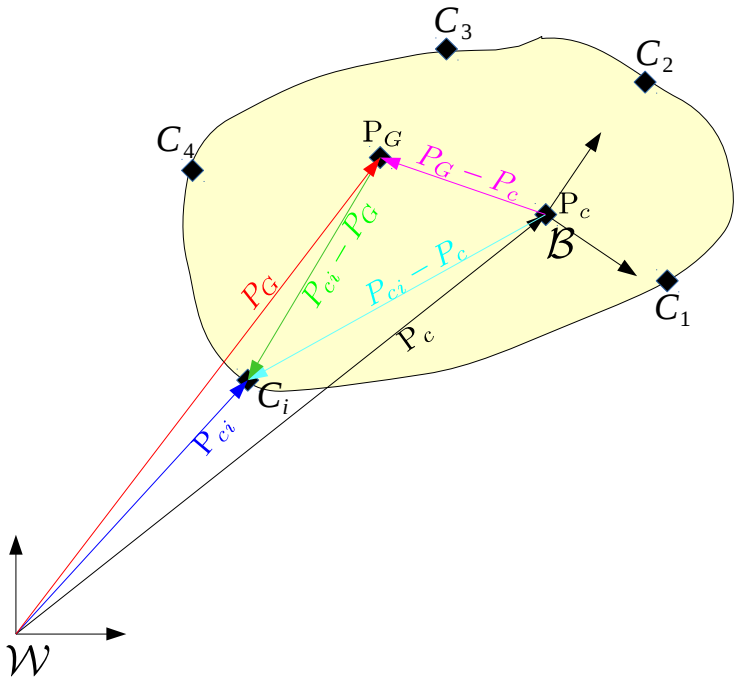


Figure 2.1: Schematic of a rigid body showing the contact points C_i where $i = 1, 2, 3..., n$, their position $\mathbf{p}_{ci}$ the CoM position $\mathbf{p}_c$ and geometrical center of contact points $\mathbf{p}_G$

$$\begin{cases} \dot{\mathbf{v}}_c = \dfrac{1}{m}\sum_{i=1}^{n} \mathbf{f}_i \\ \dot{\omega} = \dfrac{1}{J}\sum_{i=1}^{n} (\mathbf{p}_{ci} - \mathbf{p}_c)^{\perp^T} \mathbf{f}_i \end{cases} \tag{2.1}$$

Where $m \in \mathbb{R}$ is the mass of the object, $\mathbf{v}_c \in \mathbb{R}^2$ is the velocity of the center of mass (CoM) of the object, $J \in \mathbb{R}$ is the moment of inertia of the body, $\omega \in \mathbb{R}$ is the rotational rate of the object, $\mathbf{p}_c \in \mathbb{R}^2$ is the position of the CoM and $\mathbf{p}_{ci} \in \mathbb{R}^2$ is the position of the contact point C_i, for $i = 1...n$.

Also the symbol $(.)^{\perp}$ represents a rotation of $\pi/2$, for example $\mathbf{v}^{\perp} = \mathbf{Q}\mathbf{v} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} v^x \\ v^y \end{pmatrix} = \begin{pmatrix} -v^y \\ v^x \end{pmatrix}$.

The approach decomposes the term $\mathbf{p}_{ci} - \mathbf{p}_c$ into $\mathbf{p}_{ci} - \mathbf{p}_G$ and $\mathbf{p}_G - \mathbf{p}_c$, graphical representation is shown in figure 2.1.

Where $\mathbf{p}_G = \frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_{ci}$ is the geometric center (the centroid position) of the contact points.

The dynamics of the systems becomes as:

$$\begin{cases} \dot{\mathbf{v}}_c = \dfrac{1}{m}\displaystyle\sum_{i=1}^{n} \mathbf{f}_i \\ \dot{\omega} = \dfrac{1}{J}\displaystyle\sum_{i=1}^{n} (\mathbf{p}_{ci} - \mathbf{p}_G)^{\perp^T} \mathbf{f}_i + \dfrac{1}{J}(\mathbf{p}_G - \mathbf{p}_c)^{\perp^T} \displaystyle\sum_{i=1}^{n} \mathbf{f}_i \end{cases} \tag{2.2}$$

Where the motivation to the decomposition of $\mathbf{p}_{ci} - \mathbf{p}_c$ will be appear as we go through the algorithm.

The control objective is to let the load velocity $\mathbf{v}_c(t)$ and the angular velocity rate $\omega(t)$ follow a given **desired trajectory** $\mathbf{v}_c^d$ and $\omega^d(t)$ using **only** the available information (local applied forces and local velocities), and a **decentralized control law** that is able to achieve the above objective, without the need of:

- All-to-all communication.
- A central processor that computes the control actions
- The knowledge of the controlled quantities $\mathbf{v}_c(t)$, $\omega(t)$.
- The knowledge of The internal state of the object $\mathbf{p}_{ci} - \mathbf{p}_c$, and.
- The knowledge of the parameters of the dynamical system.

Proposed Control Law [1]

Given a load trajectory $\mathbf{v}_c^d$ and $\omega^d(t)$, each robot i, where $i = 1, ..., n \geq 2$, applying the following control law:

$$\mathbf{f}_i = \frac{m}{n}\mathbf{u}_c + \frac{Ju_\omega - m(\mathbf{p}_G - \mathbf{p}_c)^{\perp^T}\mathbf{u}_c}{\sum\limits_{i}^{n} \|\mathbf{p}_{ci} - \mathbf{p}_c\|^2}(\mathbf{p}_{ci} - \mathbf{p}_c)^{\perp} \tag{2.3}$$

Where:

$$\mathbf{u}_c = \dot{\mathbf{v}}_c^d + k_v(\mathbf{v}_c^d - \mathbf{v}_c)u_\omega = \dot{\omega}^d + k_\omega(\omega^d - \omega) \tag{2.4}$$

Plugging $\mathbf{f}_i$ into the load dynamics, and exploiting $\sum\limits_{i=1}^{n}(\mathbf{p}_{ci} - \mathbf{p}_G) \perp = 0$ we get:

$$\dot{\mathbf{v}}_c = \dot{\mathbf{v}}_c^d + k_v(\mathbf{v}_c^d - \mathbf{v}_c)\dot{\omega} = \dot{\omega}^d + k_\omega(\omega^d - \omega) \tag{2.5}$$

Which implies the convergence of the linear CoM velocity $\mathbf{v}_c(t) \to \mathbf{v}_c^d(t)$ and the rotational rate $\omega \to \omega^d(t)$

The control law at 2.3 implies that the quantities m, J, $\sum_i^n \|\mathbf{p}_{ci} - \mathbf{p}_c\|^2$, $\mathbf{v}_c$, ω, $\mathbf{p}_{ci} - \mathbf{p}_G$, $\mathbf{p}_G - \mathbf{p}_c$, must be locally known by each robot in order for the control to be decentralized.

The above argument stimulates the design of a decentralized algorithm that each robots estimates the quantities mentioned above, while each robot can only:

- Locally measure the velocity $\mathbf{v}_{ci}$ of the contact point $\mathbf{C}_i$.
- Locally control the applied force $\mathbf{f}_i$.
- Communicate with its one-hop neighbors $\mathcal{N}_i$ (i.e. without the need of an intermediate router between each robot and its neighbors).

Overview of the Estimation Algorithm [2]

Figure 2.2 shows a block diagram of various stages of the estimation algorithm, it contains two main parts, the Kinematic and Dynamic part.

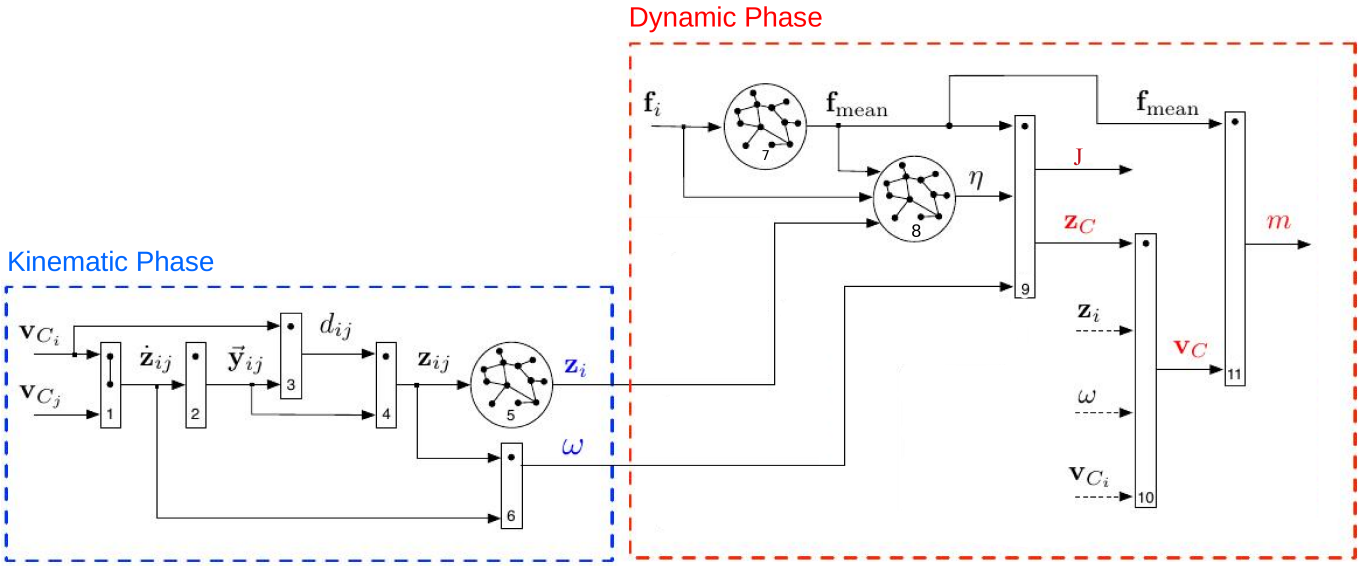


Figure 2.2: Overview of the distributed algorithm. In the kinematic phase, only the velocity measurements and the rigid body kinematics are used. After this phase, the estimates of the time-varying quantities $\mathbf{z}_i(t)$ and $\omega(t)$ (in blue) become available to each robot i. On the other hand in the dynamical phase, the knowledge of the forces and the rigid body dynamics are used as well. After this phase, the quantities $J, \mathbf{z}_C(t), \mathbf{v}_C$ and m (in red) become available to each robot i.

The Kinematic part takes measurements of velocities of contact points $\mathbf{v}_i$ and $\mathbf{v}_j$ where $j \in \mathcal{N}_i$ (a neighbor of i. With the use of rigid body kinematics, it estimates the quantities $\mathbf{z}_i = \mathbf{p}_{ci} - \mathbf{p}_G$, and ω.

The Dynamic part uses the outcomes of the Kinematic part in addition to the force inputs $\mathbf{f}_i$ and $\mathbf{v}_{ci}$, utilizing rigid body kinematics and dynamics delivers the estimates of the quantities J, $\mathbf{v}_c$, $\mathbf{z}_c$ and m.

In the next chapter we describe the experimental setup, then the theory and implementation of the two phases in detail, showing the obtained results and the discussions.

Chapter 3

Experimental Setup

This chapter describes the hardware components used to perform the experiment, and how they are interconnected, as well as the adopted software architecture, followed by a description of the steps undertaken to perform the experiment, and finally reporting how the parameters and states mentioned in the mathematical model (in subsection 2.3.1) are obtained or derived.

Hardware

A block diagram showing hardware components of ROSPO and their connections is in figure 3.1.

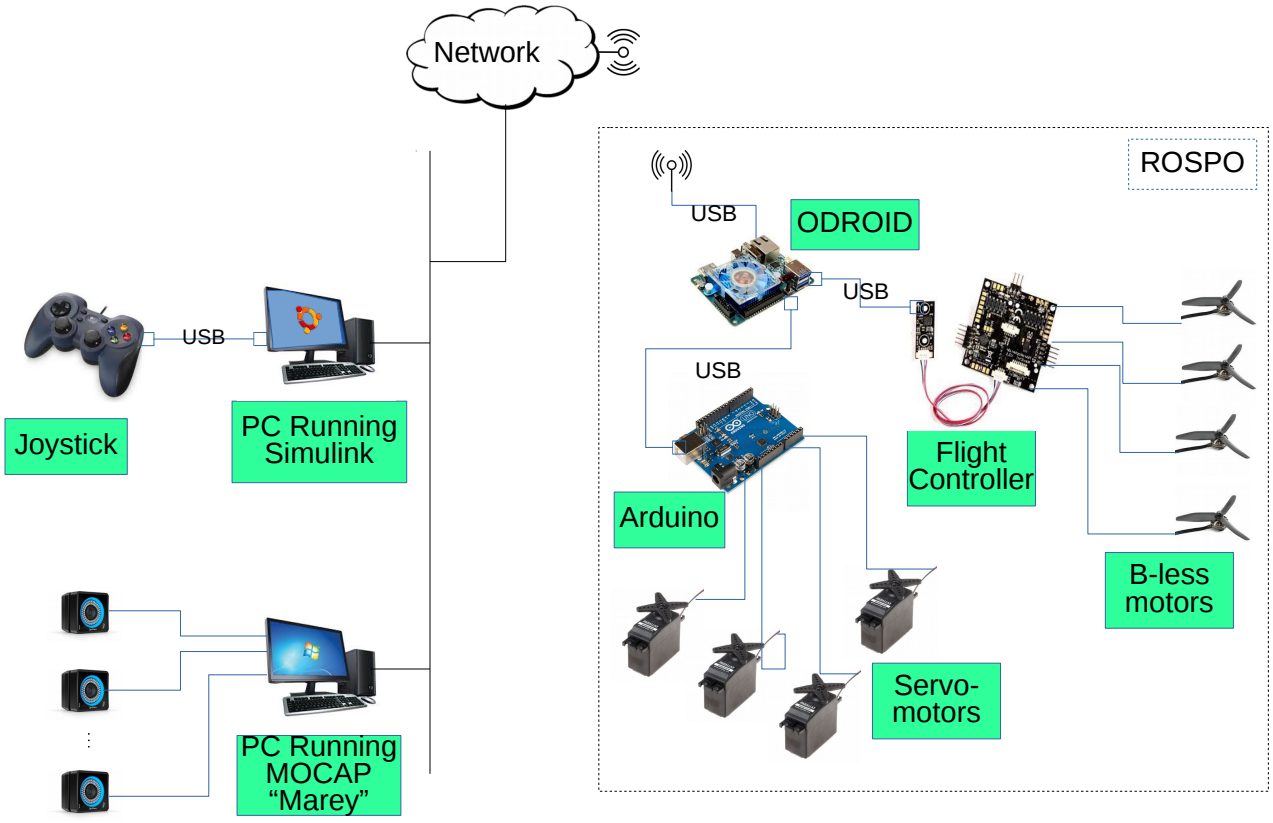


Figure 3.1: ROSPO hardware.

The platform contains 4 brush-less motors that are controlled with a control board (Flight Ctrl 2.1) by Mikrokopter [1] containing an Inertia Measurement Unit (IMU), four Brush-less Controller (BL-

[1] http://www.mikrokopter.de/en/home

Ctrl 2.0) with proprietary firmware and four Brush-less motors (ROXXY 2827-35). The platform also contains 4 servo-motors that are controlled by an Arduino board that receives serial commands from the on-board computer (ODRIOD), the latter is equipped with a wi-fi modem in order to be connected to the network.

Motion Capture The **Mocap**, is an Optitrack system from NaturalPoint[2], it consists of S250e cameras and prime17W cameras and one pc station running windows and the Optitrack software "Motive". In figure 1.6 a part of ROSPO platfom in which the markers that are detected by the cameras can be seen.

Software Architecture

OpenRobots Package

OpenRobots is a software that is being developed at LAAS/CNRS to tackle the study and design of autonomous machines [3].

robotpkg is a compilation framework and packaging system for installing robotics software developed by the robotic community. It also contains packages for some general, third-party open-source software that the robotics software depends on and that is not commonly packaged by major UNIX distributions.

Software Components

Figure 3.2 shows a block diagram of the software components used to perform the experiment.

OpenRobots software packages used in the experiment (Figure 3.2) are shown below

- **POCOLIBS** (Middleware) is a real-time communication library between software components (called GenoM components).
- **GenoM** (The Generator of Modules) is a tool to encapsulates software functions inside independent components. **genom3-pocolibs** is a template that provides a GenoM template for generating Pocolibs based components.
- **MATLAB/Simulink** is used to access control GenoM3 components using (MATLAB Genomix) package.
- tk3-mikrokopter package contains the brush-less Electronic Speed Controller (ESC) code compatible with BL-Ctrl board and the autopilot proper compatible with Flight-Ctrl boards.

[2] `http://www.naturalpoint.com/optitrack/`
[3] `https://www.openrobots.org`

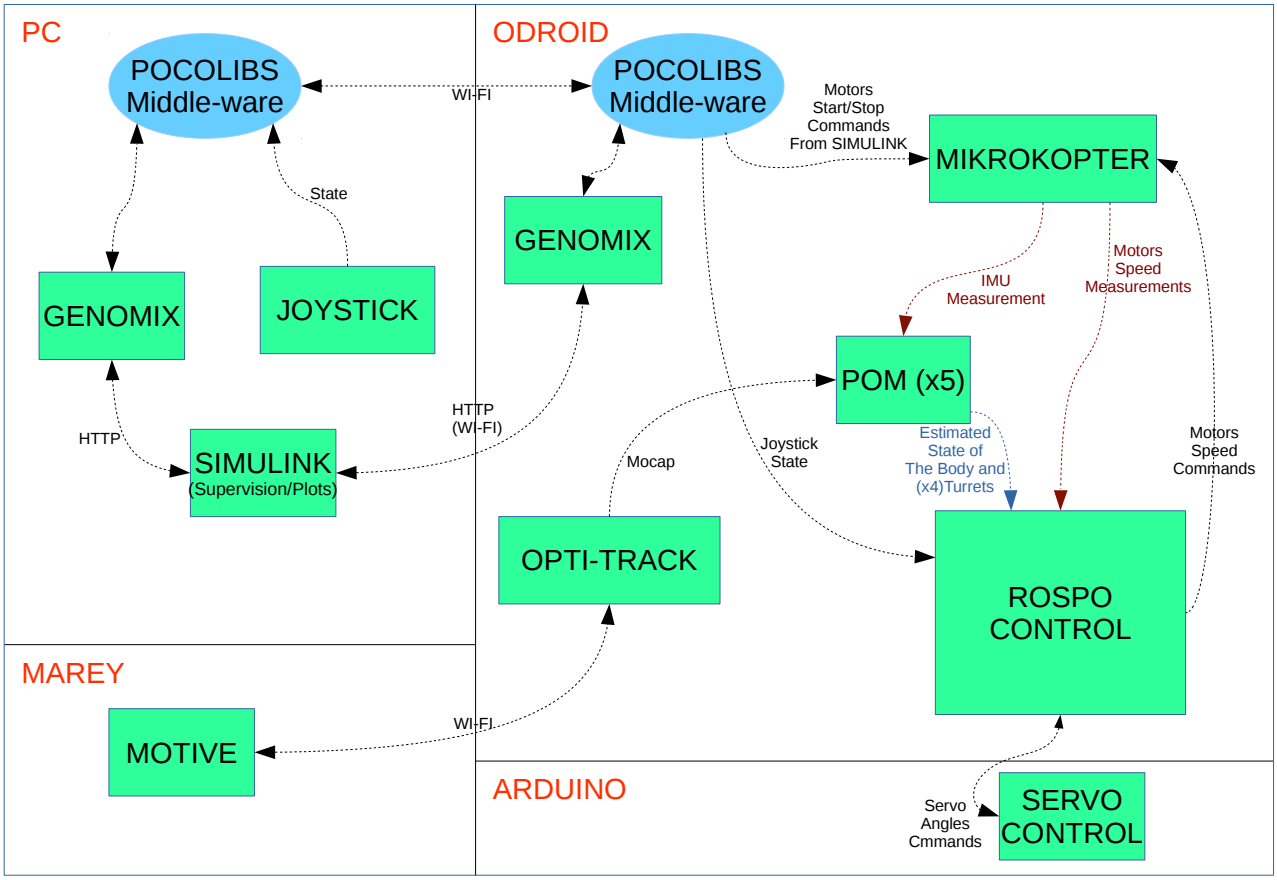


Figure 3.2: Software Components used in the experiment setup, POCOLIBS as the middle-ware to "glue" the software components in the PC, the onboard computer "ODROID" and on the desktop "MAREY" that runs the Motion Capture System.

- **mikrokopter-genom3** is a low level hardware controller for tk3-mikrokopter quadrotors. This component takes a desired wrench (forces + torques) control input in body frame and computes the propellers velocities accordingly. It communicates with the tk3-mikrokopter software running on board.
- **Optitrack-genom3** exports motion capture data from the Optitrack system.
- **pom-genom3** collects position/velocity/acceleration measurements from other components, and generate a fused state estimation from these sources.
- **joystick-genom3** is a GenoM3 component that exports the readings of any connected joystick device.
- The component "**ROSPO Control**" is and application specific GenoM3 component, it generates Brush-less motors speed commands based on the Joystick state, logs in text files the estimated state of the body and the 4 turrets from "pom-genom3", the brush-less motors speed measurements and the commands sent to the servo-control software in Arduino.

.

The Procedure

The experiment was performed by applying a force trajectory to the brush-less motors and the servo-motors based on the command of a user that operates the platform with a joystick, at the same time, the state of the platform is recorded, and saved in log files.

Then the data in the log files is applied to the algorithm implementation in Simulink in a workstation. Since we testing only the estimation algorthm in that stage and not the control part, There is no need to implement the estimation algorithm on-board. Further, this helped to test the algorithm as much as we needed without the need to perform a lot of experiments.

The tasks mentioned above are implemented in the GenoM3 component "ROSPO Control", which generate the force trajectory and process the state of the Joystick was made in Simulink and converted to C using an auto-generation tool.

The on-board computer (ODRIOD), contains linux operating system and can be accessed via Secure-Shell (SSH) protocol through the wi-fi connection. This enables controlling the execution of the components remotely from the PC station, besides operating ROSPO platform with the joystick.

Computing reference values for comparison

The variables obtained from measurements are

- the orientations of the platform and the 4 turrets represented in quaternions[4].
- The linear velocities of the platform and the 4 turrets in [meters/seconds], x and y components $\mathbf{v}_c$ and $\mathbf{v}_{ci}$.
- Angular velocity of the platform ω.
- The positions of the platform and the 4 turrets in [meters] $\mathbf{p}_{ci}$.
- Speeds measurements of the 4 brush-less motors, along with the speed sent from the software ω_{ti}.

It is useful to note that the state of the platform is estimated from the date coming from Optitrack and the data measured by the IMU, While the states of the turrets are obtained only from the Optitrack, Which makes it more noisy.

All the above vector quantities are obtained in world (inertial) frame.

[4]Obtained by pom-genom3

Orientations of the Platform and the Turrets

The orientations of the platform and the 4 turrets are represented in quaternions q_x, q_y, q_z and q_w, then they are transformed to yaw angles using the following formula.

$$Yaw = \arctan 2(2(q_w q_z + q_x q_y), 1 - 2(q_y^2 + q_z^2))$$

$\mathbf{z}_{ij}$, $\mathbf{z}_i$ and $\mathbf{z}_C$

$\mathbf{z}_{ij}$ is computed from the turret positions $\mathbf{p}_{ci}$, $i = 1, ..., 4$ and the position of the CoM of the platform $\mathbf{p}_c$ (see figure 2.1). The same applies to $\mathbf{z}_i$ and $\mathbf{z}_C$

Input Forces $\mathbf{f}_i$

The applied forces are computed from the speed measurements of the brush-less in [Hz] motors and the orientation of the turrets in [rad], also the orientation of the platform. The magnitude of the force can be obtained from the brush-less speed ω_t by the following relation.

$$|f| = k_{\omega_t} \omega_t^2$$

Where k_{ω_t} is the thrust coefficient. The value for the used propellers is $6.5 \times 10^{-4} [N/Hz^2]$.

Then the force magnitude along with the orientation of the turret represent the force in Polar coordinates, and can be easily converted to Cartesian coordinates.

Finally, the obtained force is represented in the body frame, therefore it is transformed to world frame to be suitable with the theory. This is done by multiplying it with the transformation matrix $R(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix}$, where ψ is the orientation of the platform in [rad].

The Mass and the Inertia of the Platform

The Mass of the platform was computed from mass measurements of the various parts of the platform, and the inertial was estimated by approximating these parts to the basic shapes whose formulae are known, with the latter and the part's distance from the center of mass of the platform, the rotational inertia around the center of mass of the platform is computed.

The data of the mass and the inertia computation is shown in table no. 3.1.

Communication Topology

(See section 1.7 for recap). The adopted undirected graph topology is shown in figure 3.3, which represents the most "sparse" connected graph topology of a network of 4 agents. All the results

Table 3.1: The Inertias of Various Different Parts of ROSPO platform

Part	Mass (g)	shape	Distance from CoM [cm]	w [cm]	dp [cm]	len [cm]	Inertia around CoM [kg m^2]
Box chassis	250						
MK board	70						
Arduino Board	40						
Odroid Board	50						
Electronics Box (Includes all Boards)	410	Cuboid	0	18	18	10.5	0.002214
Battery	243	Cuboid	10	6.8	2.8	10	0.002539512
Turret 1	470	Cuboid	42	10	5	9	0.0833975833
Turret 2	470	Cuboid	39	10	5	9	0.0719765833
Turret 3	470	Cuboid	41	10	5	9	0.0794965833
Turret 4	425	Cuboid	41	10	5	9	0.0718852083
Bar 1	383.52	Cuboid	25	2	94	2	0.05222264
Bar 2	383.52	Cuboid	25	2	94	2	0.05222264
Bar 3	187.68	Cuboid	47	2	46	2	0.044774192
Bar 4	187.68	Cuboid	15	2	46	2	0.00753848
Bar 5	187.68	Cuboid	15	2	46	2	0.00753848
Bar 6	187.68	Cuboid	47	2	46	2	0.044774192
Bar 7	122.4	Cuboid	36	2	30	2	0.01678512
Bar 8	122.4	Cuboid	37	2	30	2	0.01767864
Bar 9	122.4	Cuboid	7	2	30	2	0.00152184
Bar 10	122.4	Cuboid	0	2	30	2	0.00092208
Bar 11	122.4	Cuboid	35	2	30	2	0.01591608
Bar 12	122.4	Cuboid	37	2	30	2	0.01767864
Wheel 1	152	Solid cylinder	20	2.3			0.006120204
Wheel 2	152	Solid cylinder	20	2.3			0.006120204
Wheel 3	152	Solid cylinder	20	2.3			0.006120204
Wheel 4	152	Solid cylinder	20	2.3			0.006120204
Total							**0.6155633103**

shown in the next chapters are with adopting this topology unless otherwise specified.

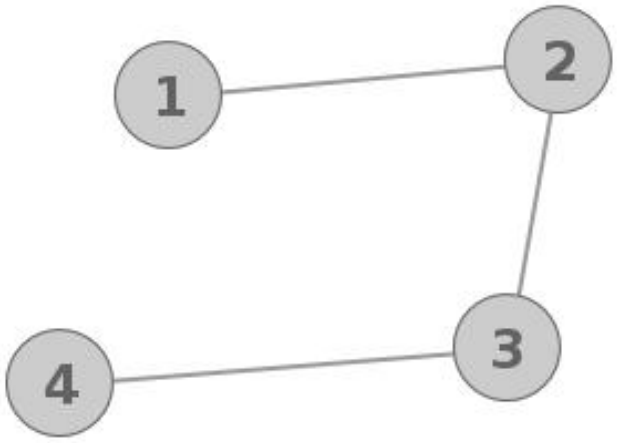


Figure 3.3: The adopted communication topology of the robotic network

Chapter 4

Kinematic Phase

Overview

This chapter goes through the theory and implementation of the Kinematic Phase stages in detail, showing the obtained results and the discussions.

The kinematic phase receives as inputs $\mathbf{v}_{ci}$, $\mathbf{v}_{cj}$ and $\hat{\mathbf{z}}_j$ where $j \in \mathcal{N}_i$ (the neighbors group), and provides $\hat{\mathbf{z}}_i$ and $\hat{\omega}$ exploiting rigid body kinematics, linear on-line estimation and a consensus-based centroid estimation algorithm.

The Inputs to the Kinematic Phase

The following figure, (4.1 shows the measurements of $\mathbf{v}_{ci}, \quad i = 1, 2, 3, 4$, along with calculated versions to show the differences between them. The calculated versions are obtained utilizing rigid body kinematics and the measurement of ω and $\mathbf{v}_c$. The reading suffer from a fair amount of noise and deviations from the calculated versions at various time intervals. Noted that the measurement of $\mathbf{v}_{c2}$ is a bit worse than the other agents.

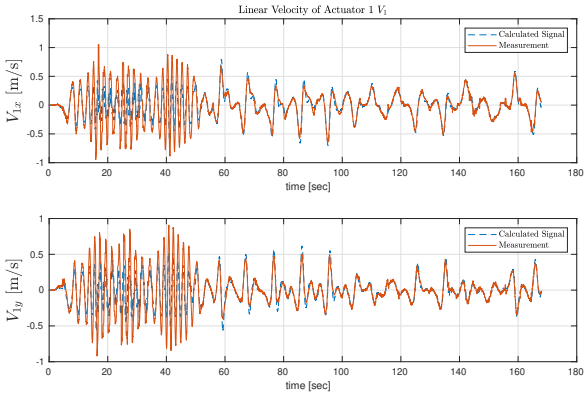
$\dot{\mathbf{z}}_{ij}$ Estimation (Stage 1)

$\dot{\mathbf{z}}_{ij}$ is computed from:

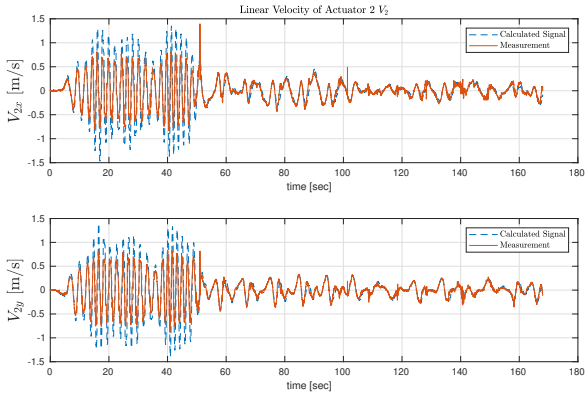
$$\dot{\mathbf{z}}_{ij} = \mathbf{v}_i - \mathbf{v}_j$$

For every $j \in \mathcal{N}_i$.

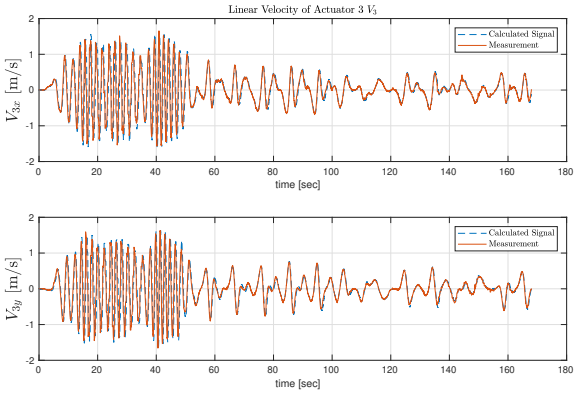
$\dot{\mathbf{z}}_{ij}$ is the rate of change of the distance between the contact point i and contact point j. All the obtained quantities are in World frame, so we expect so get a value as long as the body is rotating ($\|\mathbf{v}_i - \mathbf{v}_j\| \neq 0$)



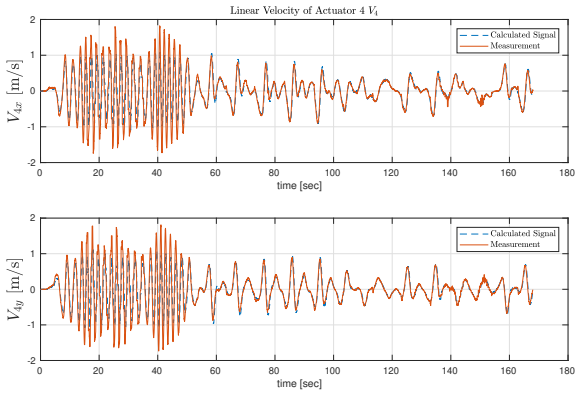
(a) Contact point 1 velocity: Reading VS Measurement



(b) Contact point 2 velocity: Reading VS Measurement



(c) Contact point 3 velocity: Reading VS Measurement



(d) Contact point 4 velocity: Reading VS Measurement

Figure 4.1: Velocity of Contact point between each robot and the manipulated object

$\mathbf{y}_{ij}$ Estimation (Stage 2)

$$\mathbf{y}_{ij} = \mathbf{Q}\mathbf{z}_{ij}/\|\mathbf{z}_{ij}\|\frac{\mathbf{Q}(\dot{\mathbf{p}}_{ci} - \dot{\mathbf{p}}_{cj})}{\|\dot{\mathbf{p}}_{ci} - \dot{\mathbf{p}}_{cj}\|} \tag{4.1}$$

Where $Q = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$.

A lower limit for the norm of $\dot{\mathbf{z}}_{ij}$ is set instead of exactly zero. $\mathbf{y}_{ij}$ is not updated as long as $\|\dot{\mathbf{z}}_{ij}\|$ goes below the limit "$\dot{\mathbf{z}}_{ij} < Th_1$"

d_{ij} Estimation(Stage 3)

Problem Formulation

$$\dot{\mathbf{z}}_{ij} = d_{ij}\dot{\mathbf{y}}_{ij}, \tag{4.2}$$

The x component of the vector equation was feed into a Recursive Least Square Estimation to obtain an Estimate of d_{ij}. Then the average of the two estimates is considered.

for the linear system described in (4.2), Which can be written as follows:

$$\dot{\mathbf{y}} = \frac{1}{d_{ij}}\mathbf{u}$$

On-line Estimator: Recursive Least Square

This algorithm was applied to in stages 3 and 11 (see figure 2.2)

Then the linear system is discretized using (Backward Euler Method) as follows:

$$\begin{aligned} \frac{y(t) - y(t-1)}{T_s} &= \frac{1}{d_{ij}}u(t-1) \\ y(t) - y(t-1) &= \frac{T_s}{d_{ij}}u(t-1) \\ h(t) &= \theta u(t-1) \end{aligned} \tag{4.3}$$

Where $t = 1, 2, 3, 4, ...$

The adopted Recursive Least-Square algorithm:

$$\begin{aligned} \mathbf{S}(t) &= \mathbf{S}(t-1) + u(t-1)^2 \\ K &= u(t-1)/\mathbf{S}(t) \\ e &= h(t) - (\hat{\theta}(t-1) * u(t)) \\ \hat{\theta}(t) &= K * e + \hat{\theta}(t-1) \\ \hat{d_{ij}} &= \frac{T_s}{\hat{\theta}} \end{aligned} \tag{4.4}$$

The value of $\hat{d_{ij}}$ is saturated not to exceed [-30, 30].

The end of the estimation is determined when the estimation (of both components) reaches a steady-state. That is, when the difference between the current value of the estimation and the previous value does not exceed a threshold for a certain amount of time. From the experiments a threshold of $0.002m$ and a time of 10000 samples were found suitable to provide an acceptable estimation.

In the previous experiment the threshold was $0.005m$ and the time was 5000 samples, Which puts more constraints on the accuracy of the estimation before selecting a value for the next stages.

Selecting Trajectory for $\|\mathbf{d}_{ij}\|$ Estimation

Since d_{ij} changes the sign whenever the difference $\mathbf{v}_i - \mathbf{v}_j$ (or we could say the sign of ω) changes the sign. So it is preferable to select a motion trajectory that keeps the sign of the $\mathbf{v}_i - \mathbf{v}_j$ unchanged for a sufficient amount of time.

Therefore, at the beginning of the algorithm the turrets must apply forces to the platform in order to keep the angular velocity ω in a constant sign until the estimation of $\hat{d_{ij}}$ is complete.

The rest of the trajectory was performed simply by setting constant rotational speeds of the turrets with the following orientations $\theta_1 = 0^o$, $\theta_2 = 0^o$, $\theta_3 = 0^o$, $\theta_4 = 0^o$, this will allow the platform to rotate around its center with small amount of linear translations.

The linear estimator is albe to obtain an estimate with estimation error within the specified threshold (see the previous sub-section), thanks to the fact that we are certain of the model structure in 4.2, the only uncertainty is introduced by the noise in the input velocities.

d_{ij} Sign Ambiguity

To solve the issue, two linear systems were implemented, one with the positive d_{ij} and the other with negative d_{ij}.

Sign Tracking Algorithm: Complementary Filters

For stage 3, two linear systems were implemented, one with the positive d_{ij} and the other with negative d_{ij}.

The system can be described as the following:

$$\begin{aligned} \dot{\mathbf{y}} &= \pm\alpha\mathbf{u} \\ y_y &= y + \mu_y \\ y_u &= u + \mu_u \end{aligned} \tag{4.5}$$

$$\begin{aligned} \dot{\hat{y}}_1 &= \alpha y_u + k_p(y_y - \hat{y}_1) \\ \dot{\hat{y}}_2 &= -\alpha y_u + k_p(y_y - \hat{y}_2) \end{aligned} \tag{4.6}$$

Any of the equations above can be written in the frequency domain as follows:

$$\hat{y}_{1or2}(s) = \frac{k_p}{s + k_p} y_y(s) + \frac{s}{s + k_p}(\pm\alpha)\frac{y_u(s)}{s} \tag{4.7}$$

Equation (4.7) shows that the Complementary Filter can serve as a high-pass filter to prevent the drift resulting from the integration of $y_u(s)$ as well as a low-pass filter that attenuates the high frequency noise affecting the measurements.

Where the computation of $\mathbf{y}_{ij}$ stops and the current value is saved if the magnitude of $\dot{\mathbf{z}}_{ij}$ is less than a certain threshold $||\dot{\mathbf{z}}_{ij}|| < Th_1$, this is because the effect of the noise is higher at this region.

This stage was found more effective by using both components of equation (4.5). The implementation is shown in figure 4.2.

Results of the Sign Tracking Algorithm

If an error in the d_{ij} sign tracking stage causes a wrong tracking of the sign, the wrong sign of d_{ij} will lead to a flipped sign of the next stages (for example ω will be computed with a flipped sign).

The following figure (4.3) shows the tracking of the sign of ω using the Complementary Filters shown above, using the values of thresholds $Th_1 = \sqrt{0.002}$. The plots show the sign tracking using one component (x and y), and by adding both results.

In 4.3, the ability of the accumulated signal (in black), to follow the sign of the angular velocity measurement, shows that it represents the correct sign of d_{ij}.

$\mathbf{z}_{ij}$ Estimation (Stage 4)

Computing $\mathbf{z}_{ij}$ is done by simply multiplying $\mathbf{y}_{ij}$ with d_{ij}

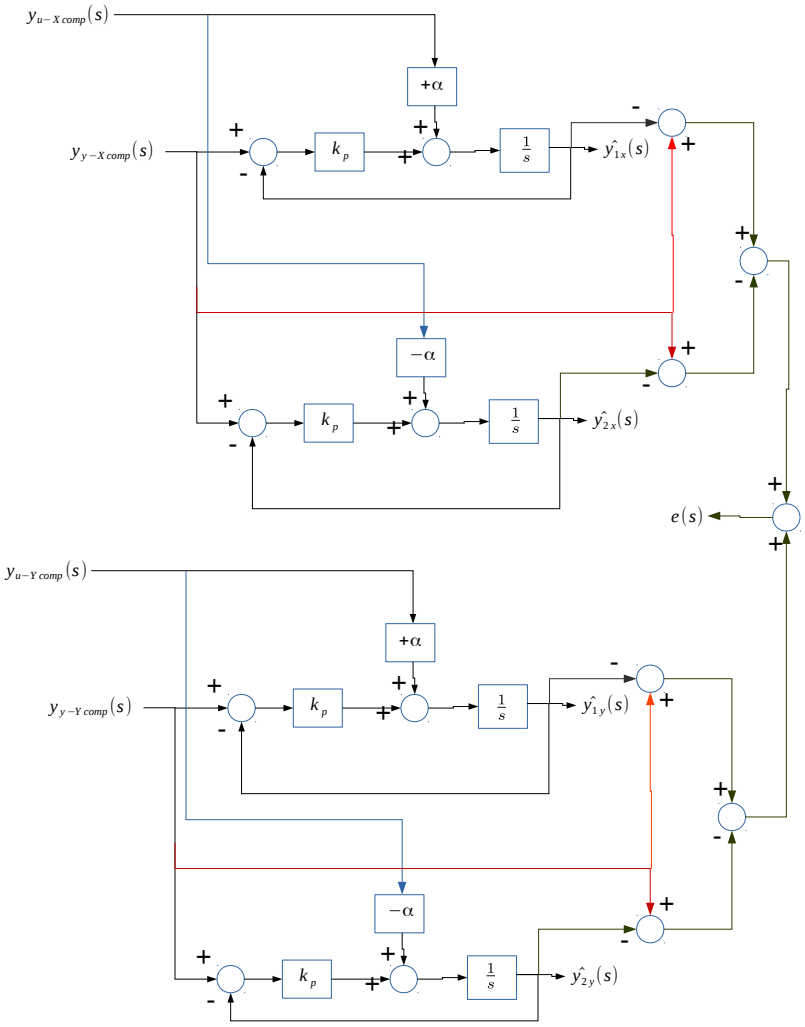


Figure 4.2: The implementation of Complementary Filters to estimate the correct sign of d_{ij}

Centroid Algorithm: $\mathbf{z}_i$ Estimation (Stage 5)

Overview of The Centroid Algorithm

$\mathbf{z}_i$ is estimated using distributed centroid estimation algorithm from relative positions $\mathbf{z}_{ij}$. Based on the work of [21], that they proposed a distributed technique for estimating the centroid of a network of agents from noisy relative measurements. Figure 4.4 shows

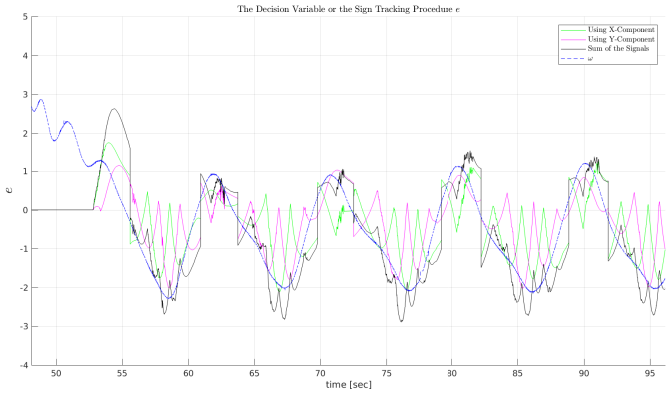


Figure 4.3: Angular Velocity Estimation Sign Estimation using $\mathbf{v}_{c1}$ and $\mathbf{v}_{c2}$

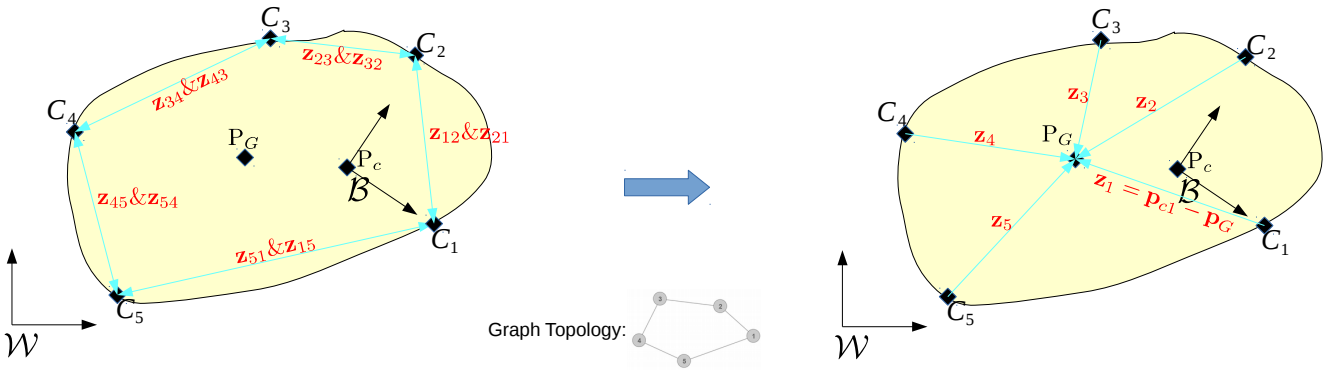


Figure 4.4: Estimating the centroid of a network of agents from relative measurements

Where $\mathbf{z}_{ij} = \mathbf{p}_{ci} - \mathbf{p}_{cj}$ is the relative position between agent i and agent j, and $\mathbf{z}_i = \mathbf{p}_{ci} - \mathbf{p}_G$ is the position of agent i relative to the centroid.

$$\hat{\mathbf{z}}_i(t+1) = \mathcal{W}_{ii}\hat{\mathbf{z}}_i(t) + \sum_{j \in \mathcal{N}_i \cup i} \mathcal{W}_{ij}(\hat{\mathbf{z}}_j(t) + \hat{\mathbf{z}}_{ij}(t))$$

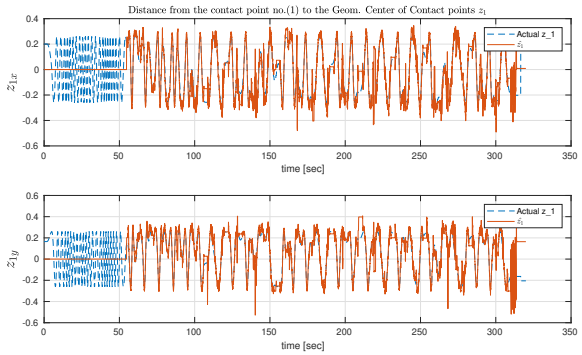
Exploiting the current estimates of $\hat{\mathbf{z}}_{ij}$ and the estimated states $\hat{\mathbf{z}}_j$ of the neighbors group ($j \in \mathcal{N}_i$).

$\mathcal{W}_{ij}$ are the Metropolis weights of the undirected version of the communication graph. They are defined as:

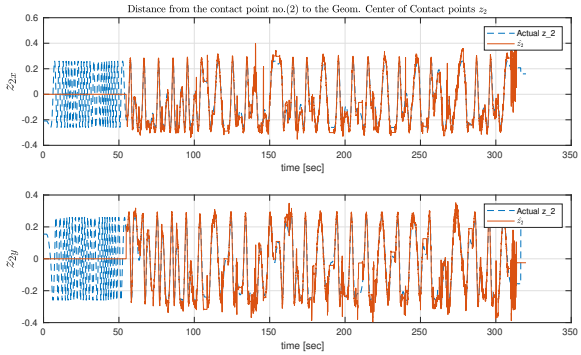
$$W_{ij} = \begin{cases} \frac{1}{1+max\{d_i,d_j\}} & if\{i,j\} \in \} \\ 1 - \sum_{\{i,k\} \in \}} W_{ik} & if i = j \\ 0 & otherwise \end{cases}$$

Results of the Centroid Algorithm

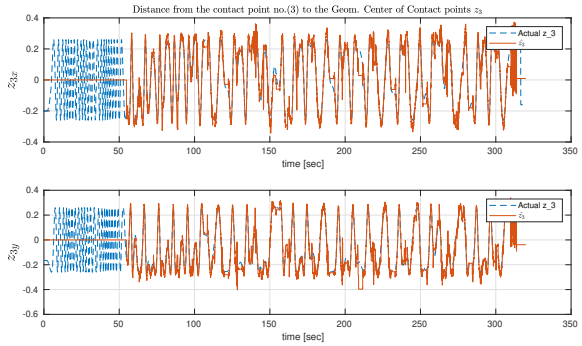
The results are shown by figure 4.5.



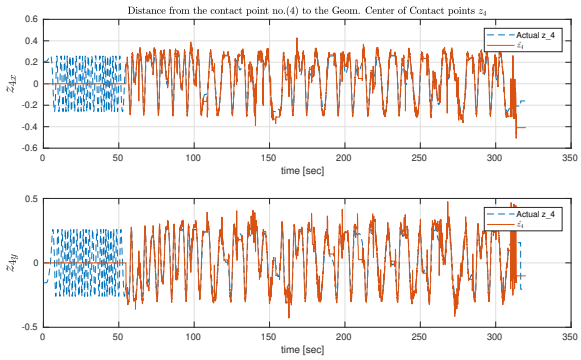
(a) $\mathbf{z}_1$ Estimation by agent1



(b) $\mathbf{z}_2$ Estimation by agent2



(c) $\mathbf{z}_3$ Estimation by agent3



(d) $\mathbf{z}_4$ Estimation by agent4

Figure 4.5: Centroid Estimation algorithm

It is intuitive to assume that the communication network topology is a major factor in the quality of estimation (see 7), the more sparse the network the less the estimation quality is. This is actually the case here, and it is more relevant with the presence of errors in previous stages (the estimation of $\hat{\mathbf{z}}_{ij}$), figure 4.6 shows the estimation of $\hat{\mathbf{z}}_1$ with to topologies for communication, the first is the one we are adopting in this thesis (see section 3.5), and the other one is all-to-all communication which is the most dense topology that can be used. From that it is safe to assume that also increasing the number of agents improves the quality of estimation.

ω Estimation (Stage 6)

The angular rate of the object can be computed locally from the quantities $\dot{\mathbf{z}}_{ij}$ (from refStage 5) and $\mathbf{z}_{ij}$ (from refStage 1) exploiting the rigid body kinematics.

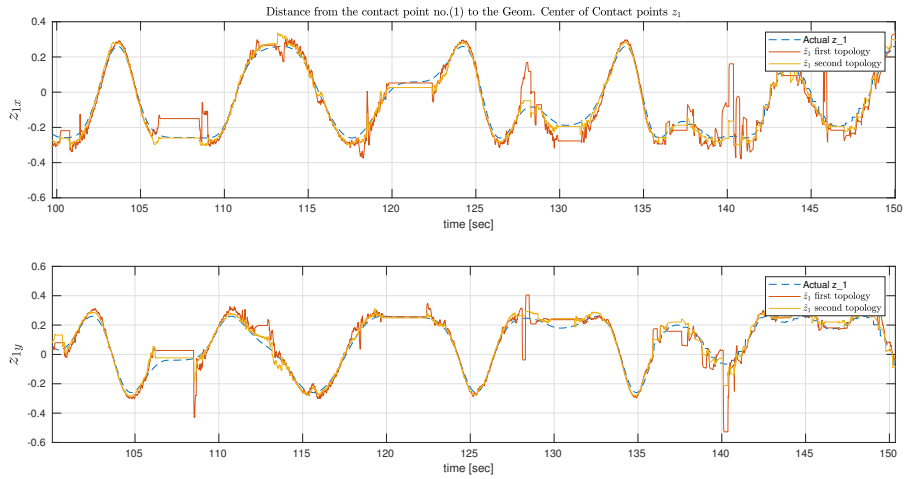


Figure 4.6: Z_4 Estimation by the agent 4

$$\omega \mathbf{z}_{ij} = -\dot{\mathbf{z}}_{ij}^{\perp}, \tag{4.8}$$

Multiplying both sides by $\mathbf{z}_{ij}^{6}$ we get ω

$$\omega = -(\mathbf{z}_{ij}^{T}\dot{\mathbf{z}}_{ij}^{\perp})(\mathbf{z}_{ij}^{T}\mathbf{z}_{ij})^{-1}$$

Each robot i will then have as many estimates of ω as the number of its neighbors $j \in \mathcal{N}_i$, These estimates can be averaged to produced more accurate results.

Obtained Estimation of ω

The estimation of the 4 agents along with the measurement of ω is shown in figure 4.7, a zoomed plot of the same is shown in figure 4.8.

The estimations of ω by the 4 agents shows that the sign tracking algorithm is following the direction reverse of $\hat{\mathbf{y}}_{ij}$, which is reflected in smoothness in the subsequent stages ($\hat{\mathbf{z}}_i$ and $\hat{\omega}$). Also the fact that each agent has several estimates of $\hat{\omega}$ (depending on the topology) and it computes the average among them, gives better estimates because it attenuates outliers.

Discussions about the Kinematic Phase

- The noise and deviations in the inputs affect the quality of the estimation considerably. mostly, the computation of the unit vector $\hat{\mathbf{y}}_{ij}$ consists of a division by the magnitude of $\hat{\dot{\mathbf{z}}}_{ij}$ the effect is more severe when the latter is almost 0.

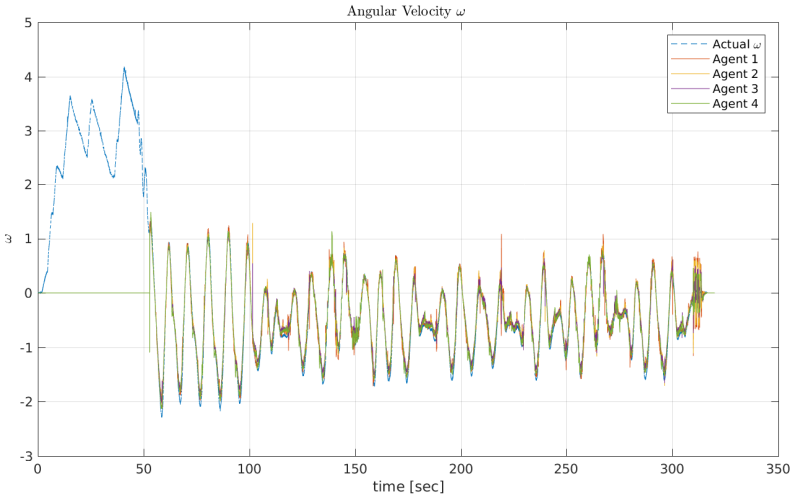


Figure 4.7: Angular Velocity Estimation by the 4 agents

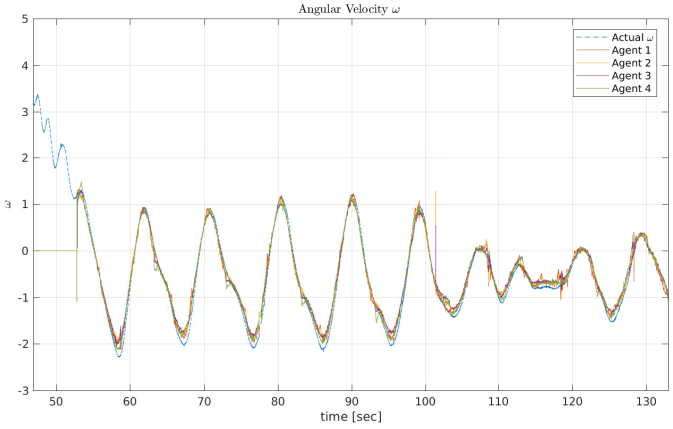


Figure 4.8: Zoomed View: Angular Velocity Estimation by the 4 agents

- Stages 3, 5 and 6 have the ability to attenuate the noise effect to some extent. Stage 3 particular in tracking the sign using the complementary filter.
- Stges 4, and 5 ($\hat{\omega}$ and the centroid algorithm) can produce better result if the communication network is more dense and/or increases the number of agents.

Chapter 5

Dynamic Phase

Overview

This chapter goes through the theory and implementation of the Dynamic Phase stages in detail, showing the obtained results and the discussions.

The Dynamic Phase receives as inputs the outcomes of the Kinematic Phase, $\hat{\mathbf{z}}_i$ and $\hat{\omega}$ and $\mathbf{f}_i$, and also exchanges some signals with the neighboring agents. and provides $\hat{\mathbf{z}}_c$, $\hat{m}$, $\hat{J}$ and $\hat{\mathbf{v}}_c$. It uses non-linear state observation, dynamic average consensus algorithms and linear on-line estimation as well.

The Inputs to the Dynamic Phase

Along with the estimates of the Kinematic Phase which are shown in the previous chapter, The Applied forces are also inputs to the Dynamic Phase. They are shown in figure no 5.1.

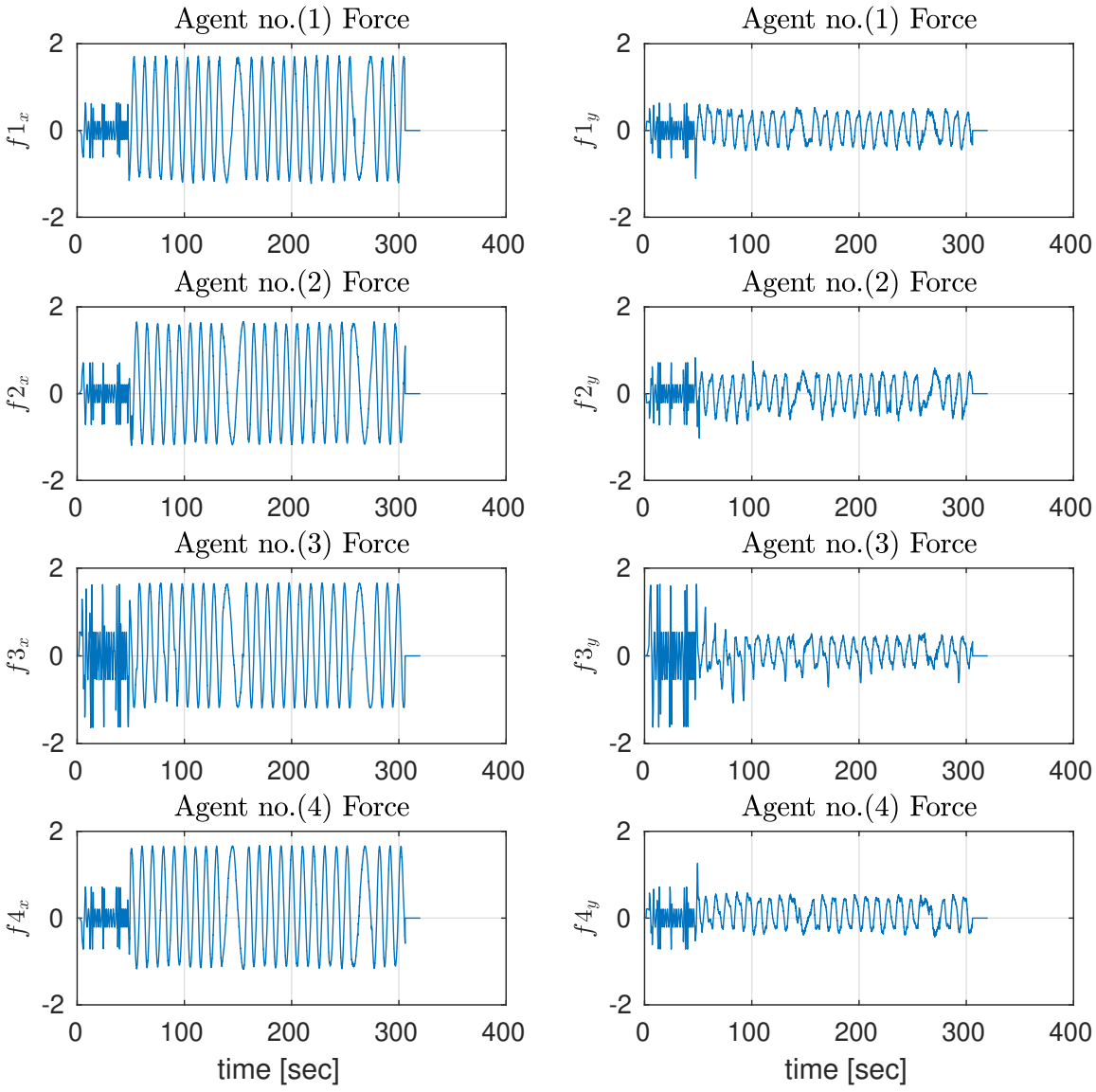


Figure 5.1: Applied Forces by the 4 Actuators

Rotational Dynamics

Recalling that $\mathbf{z}_c = \mathbf{p}_G - \mathbf{p}_c$ is a time-varying quantity that contains the information about the CoM position. The rotational dynamical equation (**??**) can be re-formulated using the following identities, the first is decomposing the local force $\mathbf{f}_i(t)$ in two parts

$$\mathbf{f}_i(t) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{f}_i(t) + \Delta\mathbf{f}_i(t) = \mathbf{f}_{\text{mean}}(t) + \Delta\mathbf{f}_i(t) \tag{5.1}$$

and recalling

$$\sum_{i=1}^{n} \mathbf{z}_i^{\perp T} = 0 \quad \text{and} \quad \sum_{i=1}^{n} \Delta\mathbf{f}_i = 0. \tag{5.2}$$

We can then write the rotational dynamics of the platform, using (5.1) and (5.2) as follows

$$\begin{aligned}\dot{\omega} =& \frac{1}{J}\left(\sum_{i=1}^{n} \mathbf{z}_i^{\perp T}\right)\mathbf{f}_{\text{mean}} + \frac{n}{J}\mathbf{z}_C^{\perp T}\mathbf{f}_{\text{mean}}(t) + \frac{1}{J}\sum_{i=1}^{n}\mathbf{z}_i^{\perp T}\Delta\mathbf{f}_i \\ &+ \frac{1}{J}\mathbf{z}_C^{\perp T}\sum_{i=1}^{n}\Delta\mathbf{f}_i + \frac{1}{J}\sum_{i=1}^{n}\tau_i \\ =& \underbrace{\frac{1}{J}\mathbf{z}_C^{\perp T} n\mathbf{f}_{\text{mean}}}_{\frac{1}{J}\mathbf{z}_C^{\perp T}\tilde{\mathbf{f}}} + \underbrace{\frac{1}{J}\sum_{i=1}^{n}\mathbf{z}_i^{\perp T}\Delta\mathbf{f}_i}_{\frac{1}{J}\eta_1} + \underbrace{\frac{1}{J}\sum_{i=1}^{n}\tau_i}_{\frac{1}{J}\eta_2},\end{aligned}$$

- $\tilde{\mathbf{f}} = n\mathbf{f}_{\text{mean}}$,
- $\eta_1 = \sum_{i=1}^{n} \mathbf{z}_i^{\perp T}\Delta\mathbf{f}_i = \sum_{i=1}^{n} \mathbf{z}_i^{\perp T}\mathbf{f}_i$, and
- $\eta_2 = \sum_{i=1}^{n}\tau_i$, This term does not exist in the case of the ROSPO platform because the actuators can only exert a force on the platform.

Rewriting the rotational dynamics

$$\dot{\omega} = J^{-1}\mathbf{z}_C^{\perp T}\tilde{\mathbf{f}} + J^{-1}\tilde{\eta} \tag{5.3}$$

where $\omega(t)$ is known from stage 6 and $\tilde{\eta} = \eta_1 + \eta_2$ is locally known to each agent from stage 8 which is explained next.

Dynamic Average Consensus Algorithms: Estimation of η and $\mathbf{f}_{mean}$ (Stages 7 and 8)

Using the work of [22], Which is a discrete-time dynamic average consensus algorithm that allow a group of agents to track the average of their reference inputs.

Utilizing the communication topology specified by the Adjacency Matrix $\mathcal{A}$.

$$x_i(t+1) = x_i(t) + \sum_{j \neq i} \mathcal{W}_{ij}(x_j(t) - x_i(t)) + (r_i(t) - r_i(t-1))$$

Where x_i and r_i are the state and input of agent i respectively.

The Dynamic Average consensus algorithm is used to estimate three quantities $\eta_1 = J^{-1} \sum_{i=0}^{n} \mathbf{z}_i^{\perp} \mathbf{f}_i$ and $\eta_2 = j^{-1} \sum_{i=0}^{n} \tau_i$, and also the average of the applied forces $\mathbf{f}_{mean}$. The Dynamic Average Consensus computes the average of $\mathbf{z}_i^{\perp} \mathbf{f}_i$, $i = 1, 2, 3, ..., n$, i.e. $\frac{\eta}{n}$. And n is assumed to be know by each agent. This assumption is safe at this stages because there exist consensus-based algorithms that obtain it.

Results of the Dynamic Average Consensus in Stage 7

The estimation of η (stage 7) by the 4 agents is shown in figure 5.2. A zoomed version is shown is 5.3. The estimation of the average applied force $\mathbf{f}_{mean}$ is show in figure 5.4.

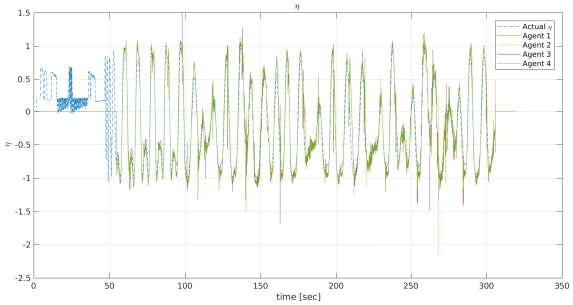


Figure 5.2: η Estimation by the 4 agents

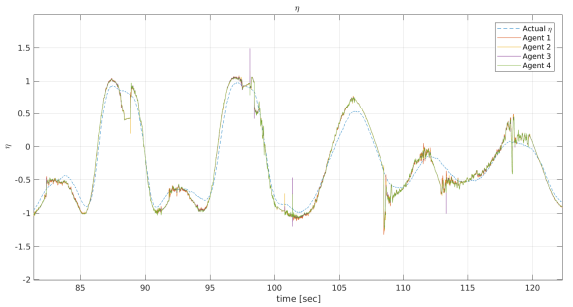


Figure 5.3: Zoomed view: η Estimation by the 4 agents

The results in both stages showed that the dynamics of the estimation is far faster then the change of the rate of change of the averages, this comes in the expense of making the estimation more sensitive to noise. The figures show differentiation effect (a spike) at various instants. To reduce that it could be useful to modify the algorithm to be less sensitive, (which I did not find a way to

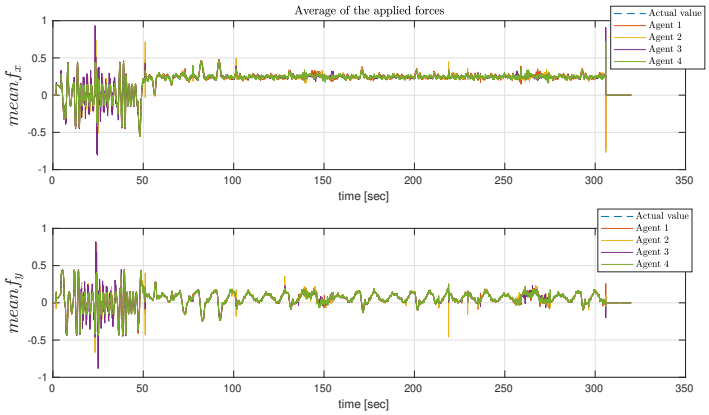


Figure 5.4: Average value of the applied forces Estimation by the 4 agents

do it), or to filter/smooth the inputs to the stages, (I have tried off-line median filter which showed effectiveness but it is not applicable in the algorithm because it has to be applied on-line.

Nonlinear State Observation: Estimation of $\mathbf{z}_C$ and J (Stage 9)

Going back to (5.3), ω, J^{-1} and $\mathbf{z}_C(t)$ can be selected as state variables and $\tilde{\mathbf{f}}$ and $\tilde{\eta}$ are the inputs. In order to complete (5.3) with the dynamics of $\mathbf{z}_C(t)$ we recall that $\mathbf{z}_C$ is a constant-norm vector, rigidly attached to the object, hence

$$\dot{\mathbf{z}}_C = \omega\, \mathbf{z}_C^{\perp}. \tag{5.4}$$

Furthermore since we do not know how J changes, and anyway in practical scenarios they change, they will change gradually, then we assume, for the design of the observer, that $\dot{J} = 0$. Combining (5.3) and (5.4), we obtain the nonlinear system

$$\begin{aligned} \dot{x}_1 &= -x_2 x_3 \\ \dot{x}_2 &= x_1 x_3 \\ \dot{x}_3 &= x_4 x_1 u_2 - x_4 x_2 u_1 + x_4 u_3 \\ \dot{x}_4 &= 0 \\ y &= x_3 \end{aligned} \tag{5.5}$$

where $(x_1\ x_2)^T = \mathbf{z}_C = (z_C^x\ z_C^y)^T$, and $x_4 = J^{-1}$ are the unknown part of the state vector, $x_3 = \omega$ is the measured part of the state vector and, consequently, can be considered as the system output, and $(u_1\ u_2)^T = \tilde{\mathbf{f}}$, $u_3 = \tilde{\eta}$ are known inputs. The following subsections overview the observability conditions from the literature for this class of non-linear systems and applies the definition to the system in 5.5. Followed by the theoretical proof of convergence and a demonstration of the outcome

of this stage.

Nonlinear System Observability Overview

Considering a general nonlinear system with a single output

$$\begin{cases} \dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})u \\ y = h(\mathbf{x}) \end{cases}, \tag{5.6}$$

Where $f : \mathbb{R}^n \to \mathbb{R}^n, g : \mathbb{R}^n \to \mathbb{R}^n$, assuming that $f(.), g(.)$ are sufficiently smooth and that $h(0) = 0$. We define the following notation [23]

- $\mathbf{x}_u(t, \mathbf{0}x_0)$: represents the solution of (5.6) at time t originated by the input u and the initial state x_0.
- $y(\mathbf{x}_u(t, \mathbf{0}x_0))$: represent the output y when the state $\mathbf{x}$ is $\mathbf{x}_u(t, \mathbf{0}x_0)$.

Clearly

$$y(\mathbf{x}_u(t, \mathbf{0}x_0)) \equiv h(\mathbf{x}_u(t, \mathbf{0}x_0))$$

A pair of states (x_0^1, x_0^2) s said to be distinguishable [**?**] if there exists an input function u such that

$$y(\mathbf{x}_u(t, \mathbf{0}x_0^1)) \equiv y(\mathbf{x}_u(t, \mathbf{0}x_0^2))$$

Local Observability can be defined as follows: The state space realization (5.6) is said to be (locally) observable at $\mathbf{x}_0 \in \mathbb{R}^n$ if there exists a neighborhood U_0 of $\mathbf{x}_0$ such that every state $\mathbf{x} \neq \mathbf{x}_0 \in \Omega$ is distinguishable from $\mathbf{x}_0$. It is said to be locally observable if it is locally observable at each $\mathbf{x}_0 \in \mathbb{R}^n$ [23]

This means that (5.6) is locally observable in a neighborhood $U_0 \subset \mathbb{R}_n$ if there exists an input $u \in \mathbb{R}$ such that

$$y(\mathbf{x}_u(t, \mathbf{0}x_0^1)) \equiv y(\mathbf{x}_u(t, \mathbf{0}x_0^2)) \quad \forall t \in [0, t] \quad \Leftrightarrow \quad \mathbf{x}_0^1 = \mathbf{x}_0^2$$

Considering an unforced nonlinear system of the form

$$\begin{cases} \dot{\mathbf{x}} = f(\mathbf{x}) \\ y = h(\mathbf{x}) \end{cases}, \tag{5.7}$$

Also $f : \mathbb{R}^n \to \mathbb{R}^n, g : \mathbb{R}^n \to \mathbb{R}^n$. Looking for Observability conditions in a neighborhood of the origin $\mathbf{x} = 0$, the state space realization (5.7) is locally observable in a neighborhood $U_0 \subset D$ containing the origin, if [23]

$$rank \left(\begin{bmatrix} \nabla h \\ \vdots \\ \nabla L_f^{n-1} h \end{bmatrix} \right) = n \qquad \forall \mathbf{x} \in U_0 \tag{5.8}$$

Observability Analysis

The theorem in (5.8) was applied to the nonlinear system in (5.5) in order to assess the Observability of the system.

The computation of the Observability matrix for (5.5) requires a great effort of hand-writing. The matrix contains a large amount of terms that it is difficult to compute the determinant of the matrix to assess whether it has full rank or not. Therefore MATLAB's tool called Symbolic Math was used to compute the determinant and the expression is as follows

$$det(O) = -u_3 x_3^3 x_4^2 (u1^2 + u2^2) \tag{5.9}$$

The Observability depends on the values of the states x_3 and x_4 in addition to the inputs. The system is considered observable if x_3 and x_4 and u_3 and at lease one of u_1, u_2 are not equal to 0.

Designing an observer

Consider the following dynamical system

$$\begin{aligned} \dot{\hat{x}}_1 &= -\hat{x}_2 x_3 + u_2(y - \hat{x}_3) \\ \dot{\hat{x}}_2 &= \hat{x}_1 x_3 - u_1(y - \hat{x}_3) \\ \dot{\hat{x}}_3 &= \hat{x}_1 u_2 - \hat{x}_2 u_1 + k\hat{x}_4 u_3 + k_e(y - \hat{x}_3) \\ \dot{\hat{x}}_4 &= k u_3 (y - \hat{x}_3), \end{aligned} \tag{5.10}$$

where $k_e > 0$. If $y(t) \not\equiv 0$, $u_3(t) \not\equiv 0$ and $(u_1(t)\ u_2(t))^T \not\equiv \mathbf{0}$, then (5.10) is an asymptotic observer

of a modified version of (5.5), in which the following change of variable is done $\hat{x}_4 \to \frac{x_4}{k}$, $\hat{x}_1 \to x_1 x_4$, $\hat{x}_2 \to x_2 x_4$, *i.e.*, defining $\hat{\mathbf{x}} = (\hat{x}_1\ \hat{x}_2\ \hat{x}_3\ \hat{x}_4)^T$ and $\mathbf{x} = (x_1 x_4\ x_2 x_4\ x_3\ x_4/k)^T$, one has that $\hat{\mathbf{x}}(t) \to \mathbf{x}(t)$ asymptotically, which in turns also implies that $(\hat{x}_1/\hat{x}_4\ \hat{x}_1/\hat{x}_4\ k\hat{x}_4) \to (x_1\ x_2\ x_4)$ asymptotically.

Proof of Convergence

Define the error vector as $\mathbf{e} = (e_1\ e_2\ e_3\ e_4)^T = ((x_1 x_4 - \hat{x}_1)\ (x_2 x_4 - \hat{x}_2)\ (x_3 - \hat{x}_3)\ (x_4/k - \hat{x}_4))^T$; after some algebra, the error dynamics is given by

$$\dot{\mathbf{e}} = \begin{pmatrix} 0 & -x_3 & -u_2 & 0 \\ x_3 & 0 & u_1 & 0 \\ u_2 & -u_1 & -k_e & k u_3 \\ 0 & 0 & -k u_3 & 0 \end{pmatrix} \mathbf{e} \tag{5.11}$$

Define the following candidate Lyapunov function: $V(\mathbf{e}) = \frac{1}{2}\mathbf{e}^T\mathbf{e}$, whose time derivative along the system trajectories is

$$\begin{aligned} \dot{V}(\mathbf{e}) =& e_1(-e_2 x_3 - u_2 e_3) + e_2(x_3 e_1 + u_1 e_3) + \\ & e_3(u_2 e_1 - u_1 e_2 + k u_3 e_4 - k_e e_3) - e_4 k u_3 e_3 \\ =& - e_1 e_2 x_3 - e_1 u_2 e_3 + e_2 x_3 e_1 + \\ & e_2 u_1 e_3 + e_3 u_2 e_1 - e_3 u_1 e_2 - k_e e_3^2 + k u_3 e_3 e_4 \\ & - k u_3 e_3 e_4 \\ =& - k_e e_3^2, \end{aligned} \tag{5.12}$$

which is negative semi-definite. Since $\dot{\mathbf{V}} = 0$ as soon as $e_3 = 0$ and e_3 can be of any value, we need to study the invariant set imposing $e_3 \equiv 0$, which implies, in particular, that $e_3 = 0$, $\dot{e}_3 = 0$, and $\ddot{e}_3 = 0$.

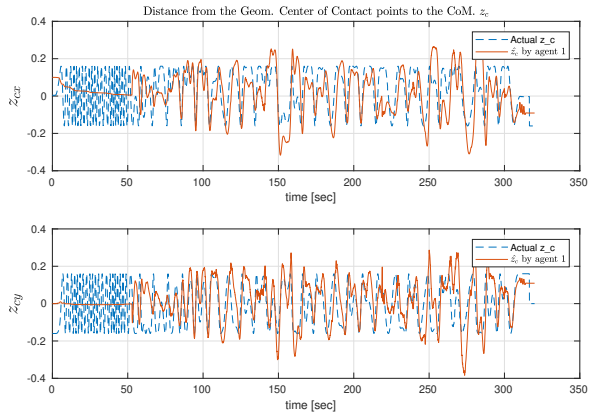
Considering, for simplicity the case in which the inputs are step-wise constant, the last three equation result in the following linear system:

$$\begin{pmatrix} u_2 & -u_1 & u_3 \\ -x_3 u_1 & -x_3 u_2 & 0 \\ -x_3^2 u_2 & x_3^2 u_1 & 0 \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ e_4 \end{pmatrix} = \mathbf{E} \begin{pmatrix} e_1 \\ e_2 \\ e_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{5.13}$$

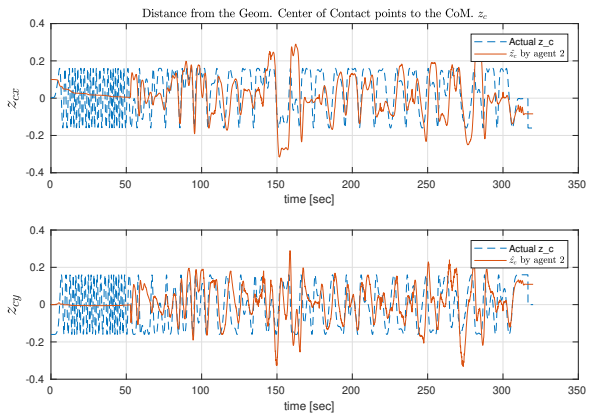
The determinant of $\mathbf{E}$ is $-u_3 x_3^3 (u_1^2 + u_2^3)$, if the assumptions of the theorem are satisfied then $\mathbf{E}$ is non-singular and therefore the only trajectory of the system that ensures $\dot{V} = 0$ is $\mathbf{e} = \mathbf{0}$. Therefore, this non-linear observer is an asymptotic observer 5.5.

Results of Nonlinear State Observation

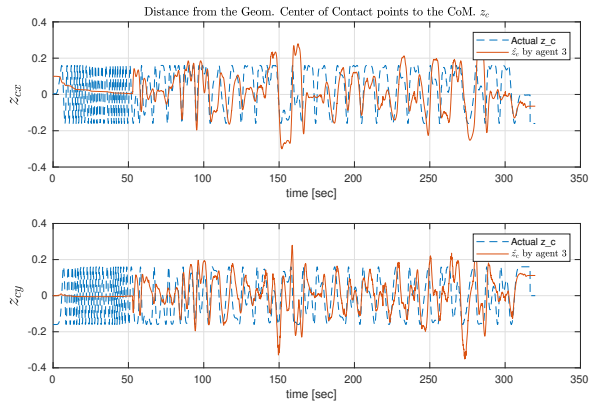
Figures 5.5 shows the estimtion of $\hat{\mathbf{z}}_c$ by agents 1, 2, 3, and 4 respectively, and figure 5.6 shows the estimation of $\hat{J}$ of all 4 agents, the estimate is plotted along with the computed version.



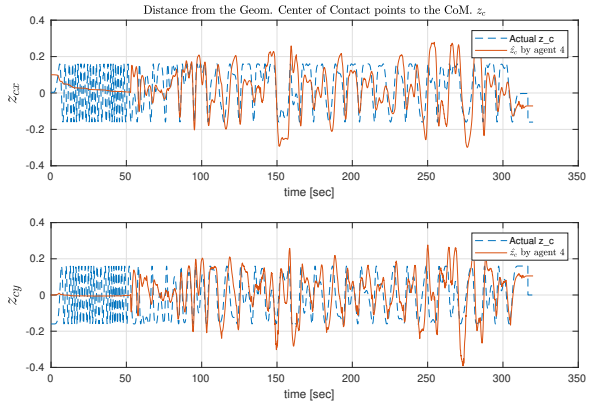
(a) $\mathbf{z}_C$ Estimation by agent1



(b) $\mathbf{z}_C$ Estimation by agent2



(c) $\mathbf{z}_C$ Estimation by agent3



(d) $\mathbf{z}_C$ Estimation by agent4

Figure 5.5: $\mathbf{z}_C$ Estimation

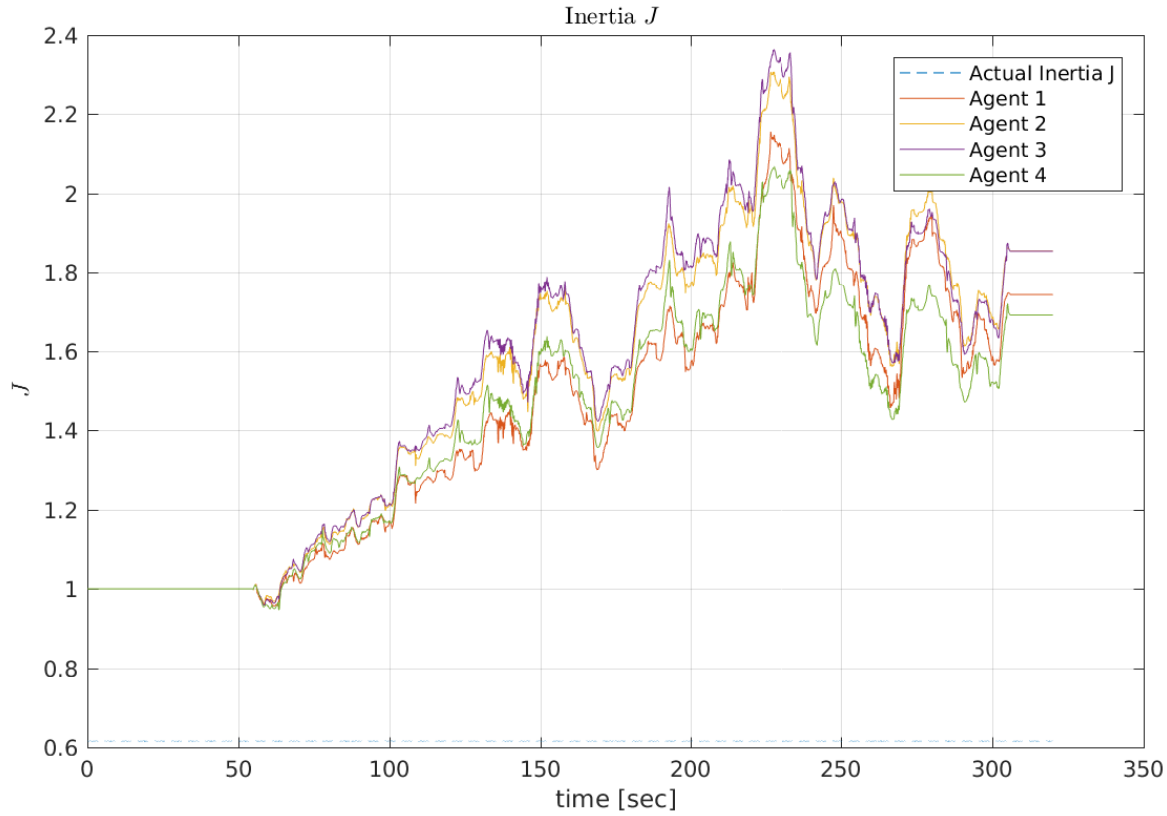


Figure 5.6: Platform Inertia Estimation by the 4 agents

Figure 5.7 shows the estimation of one of the agents using the estimated signals coming from the previous signals.

The nonlinear observer is at stage 9, all its inputs and outputs are taken from previously estimated quantities. To to study the effect of the accumulated errors on the performance of the observer, the observer was implemented using the data coming directly with the measurements (or computed). Figure 5.8 shows the the plot of the results in both cases, and for more insight, figure 5.9, shows the abs error ($|\mathbf{z}_C - \hat{\mathbf{z}}_C|$) also in both cases.

Recalling that the inputs of the observer are: system inputs $\mathbf{f}_{mean}$, and $\eta = \sum_{i=1}^{n} \mathbf{z}_i^{\perp T} \mathbf{f}_i$, and system output ω.

Therefore the observer's performance is affected if the input and output are affected. However, the observer performance is not optimum even with the use of better data.

This can be due to the following reasons

- The presence of friction in the wheels.The mathematical model does not describe accurately the dynamics of the physical system. In particular, the presence of the friction in the wheels, although it was reduced by attaching the wheels closer to the CoM to reduce the moment of the friction forces. It still affects the dynamics in a nonlinear way.

- The inputs forces are not directly measured but computed from the propeller speed and the orientation angle of each actuator, then applying The speed-thrust quadratic low with the use

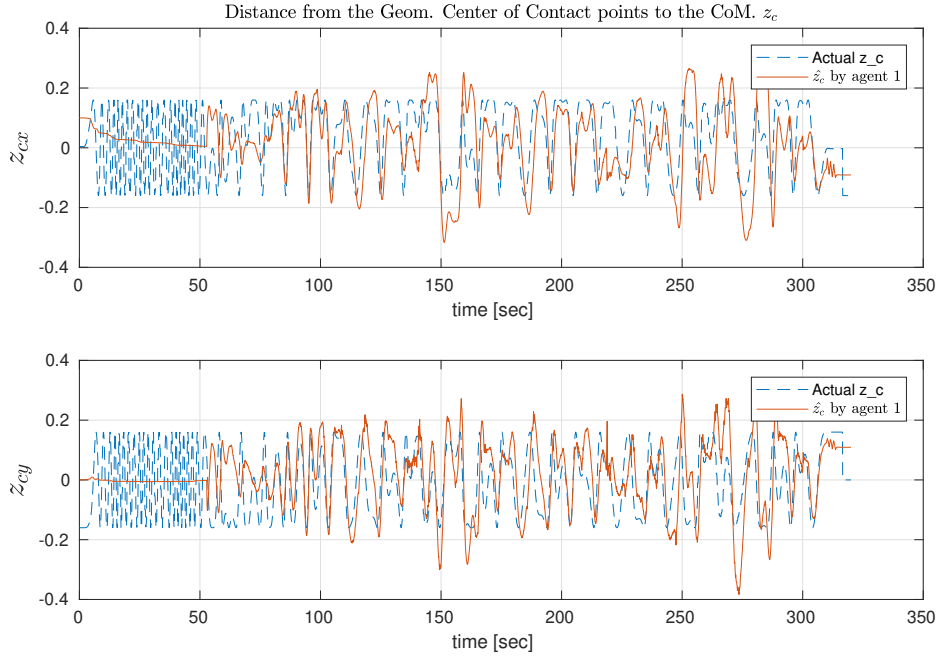


Figure 5.7: $\mathbf{z}_C$ Estimation by one of the agents. The input and output to the observer are taken from the estimated signals of the previous stages

of the constant, and finally applying a transformation from body-frame to world-frame (see the Implementation chapter).

The effect of non-linearity is studied by arranging the rotational dynamics in the form $\dot{\mathbf{y}} = \frac{1}{J}\mathbf{u}$. assuming $y = \omega$ and $u = \sum_{i=1}^{n} \mathbf{z}_i^{\perp T} \mathbf{f}_i + \mathbf{z}_C^{\perp T} \sum_{i=1}^{n} \mathbf{f}_i$.

Next MATLAB System Identification tool was used to fit the "computed signals y and u to the above linear system with. The input and output are shown in figure 5.10, and the simulated output against the measurement is shown in figure 5.11. The "best fit" criteria in MATLAB System Identification toolbox was 52.21%. This result indicates the presence of non-linearity in the physical dynamics which was not captured by the linear model. It also shows how model uncertainties affect the estimation quality.

Moreover, the obtained linear model that achieves best-fit has the transfer function of $\frac{y(s)}{u(s)} = \frac{0.7367}{s+0.2243}$, and in differential equation form $\dot{y} + 0.2243y = 0.7367$. Which is a linear model with viscous friction $0.2243[N.s/rad]$, and inertia $J = \frac{1}{0.7367} = 1.3574$ Which is close to the value obtained by the nonlinear observer.

Gain Tuning: The Effect of k_{e1} and k_{e2}

To understand the effect of the gain k_{e2}, the system in (5.5), is implemented in Simulink with the following specifications:

- Automatic Solver with Fixed Step.

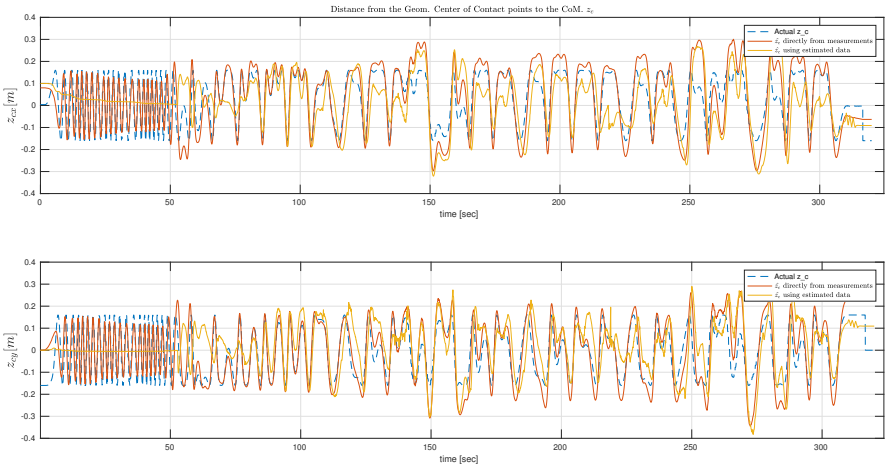


Figure 5.8: $\mathbf{z}_C$ Estimation by one of the agents. Comparing taking the data from the measurements directly with using the estimated data from previous stages.

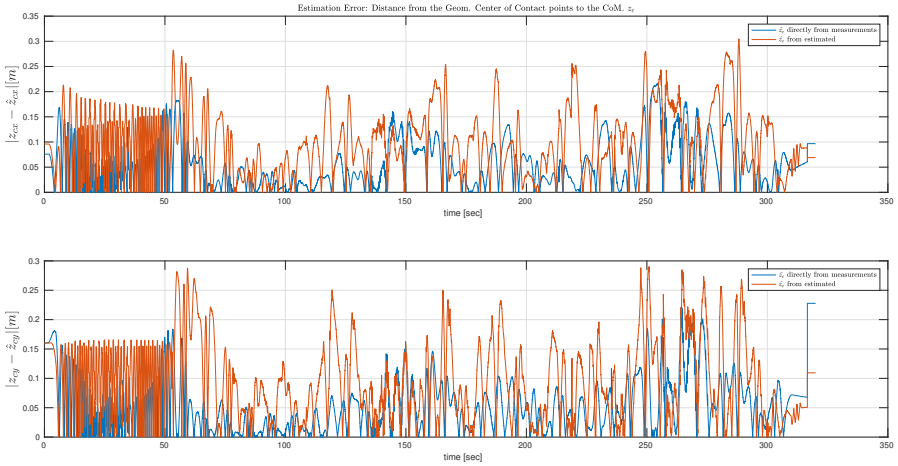


Figure 5.9: $\mathbf{z}_C$ Absolute error of $\mathbf{z}_C$ estimation by one of the agents. Comparing taking the data from the measurements directly with using the estimated data from previous stages.

- Sampling time = 0.002 seconds.
- Type of integration method: Backward Euler
- The input to the system are shown in figure 5.12.
- The initial state of the system $\mathbf{x} = (0\ 1\ 0\ 0.1)$

Figure 5.13 shows the state estimation of the observer when fixing the value of k_{e1}: $k_{e1} = 1$ and k_{e2} is varied between the values $k_{e2} = 1, 15 and 20$.

From figure 5.13, in case of $k_{e2} = 1$ the convergence of x_4 is extremely slow that it is for the duration of the simulation, it is not able to converge to the preset value $x_4 = 0.1$, while increasing k_{e2} accelerate the convergence but also makes the response more aggressive.

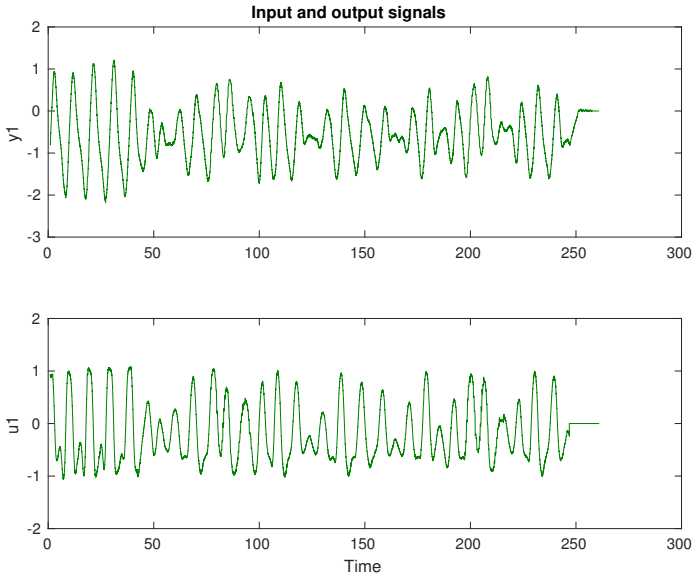


Figure 5.10: the input u and the output y for linear system identification applied to the experimental data.

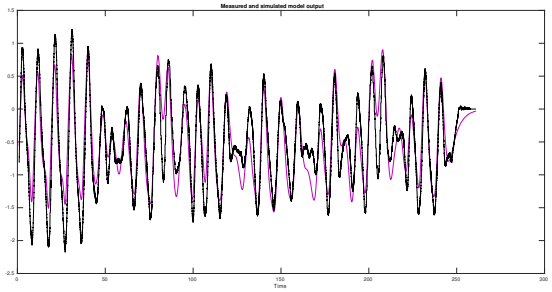


Figure 5.11: The output measurement y against the simulated output of the identified linear system.

It was noted that the speed of response depends on the amplitude u_3, for the selected amplitude $0.1Nm$ the response becomes very slow. This motivates using k_{e2} as a gain for u_3 to speed up the convergence, since u_3 is an internal signal in the overall model that depends on $\mathbf{f}_i$ and $\mathbf{z}_i$.

The effect of k_{e2} can be seen by looking at (5.10), in particular: $\dot{\hat{x}}_3$ and $\dot{\hat{x}}_4$. k_{e2} increases the effect of e_3 on the change of $\dot{e}_4$.

Increasing the gain k_{e1} reduces the speed of convergence of the states $\hat{x}_1, \hat{x}_2 and \hat{x}_4$. The reason is because it increases the effect of the term $(e_3 = x_3 - \hat{x}_3)$ on the rate of change of $\hat{x}_3$, which reduces e_3 itself, recalling that e_3 is the factor that modifies the values of the other estimated states.

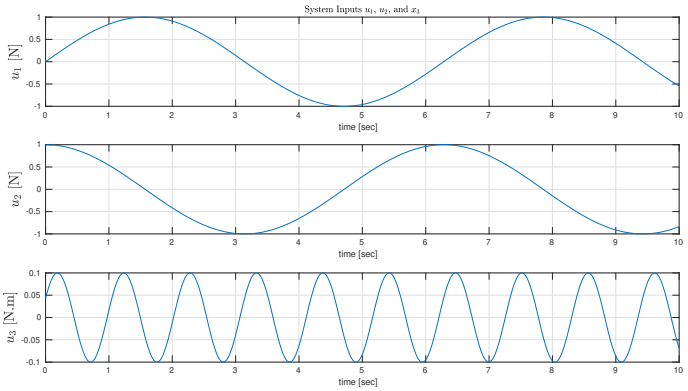


Figure 5.12: Inputs to the System

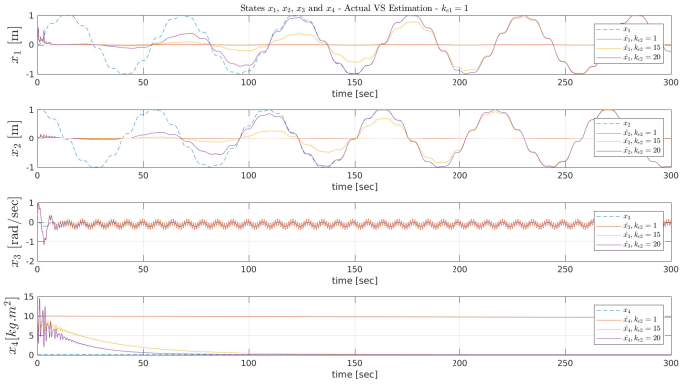


Figure 5.13: States Estimation Using Nonlinear Observer

$\mathbf{v}_C$ Calculation (Stage 10)

Utilized Kinematic Relations

The velocity of the center of mass $\mathbf{v}_C(t)$ is estimated locally by each agent i using the rigid body constraint

$$\frac{\mathrm{d}}{\mathrm{dt}}(\mathbf{p}_C - \mathbf{p}_{C_i}) = \omega(\mathbf{p}_C - \mathbf{p}_{C_i})^{\perp},$$

which can be rewritten as

$$\mathbf{v}_C = \mathbf{v}_{C_i} - \omega(\mathbf{p}_C - \mathbf{p}_{C_i}) = \mathbf{v}_{C_i} - \omega(\mathbf{p}_C - \mathbf{p}_G + \mathbf{p}_G - \mathbf{p}_{C_i}) = \mathbf{v}_{C_i} - \omega(\mathbf{z}_c + \mathbf{z}_i)$$

whose right-hand-side elements are all known since:

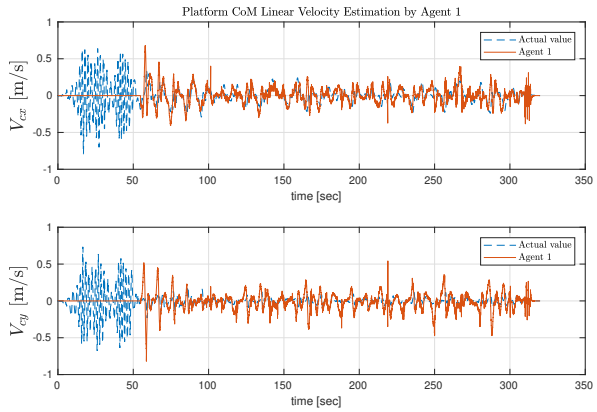
- $\mathbf{v}_{C_i}(t)$ is locally measured by agent i

- $\omega(t)$, $\mathbf{z}_C(t)$, and $\mathbf{z}_i(t)$ are known by each agent i thanks to stages 6, 11 and 5, respectively.

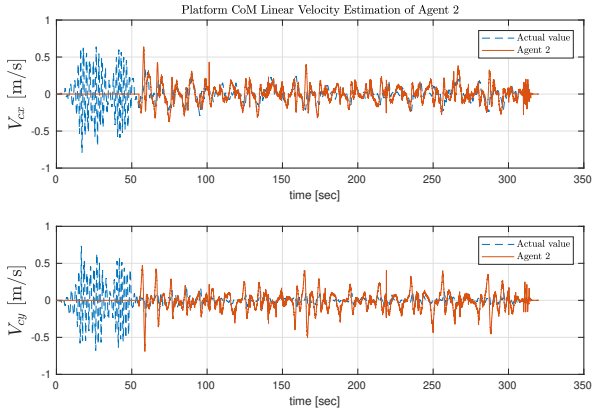
The results are shown in the following subsection.

Results and Comments

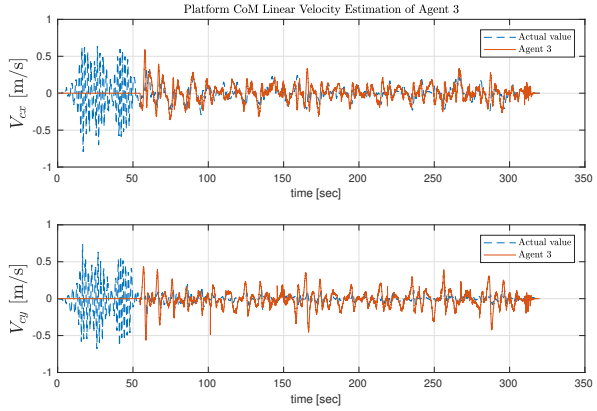
Figure 5.14 shows the plot of $\hat{\mathbf{v}}_c$ estimated by each agent, along with the measurement.



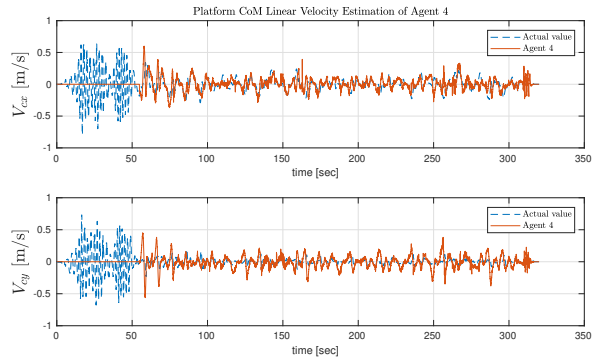
(a) Estimation by agent1



(b) Estimation by agent2



(c) Estimation by agent3



(d) Estimation by agent4

Figure 5.14: CoM Linear Velocity $\mathbf{v}_C$ Estimation

From the figures we can see that estimation error is large. It is evident that the accumulation of errors throughout the previous stages affect deeply this particular stage. Because stage 10 itself is not expected to introduce uncertainty because it is mainly an arithmetic equation.

It is useful to mention that ROSPO was arrange to maximize the distance $(\mathbf{p}_{ci} - \mathbf{p}_c)$ and in turn maximize $\mathbf{z}_c$ and $\mathbf{z}_i$ respectively $(\mathbf{z}_c \approx 18 centemeters)$, to assess their effect on $\mathbf{v}_c$. While in previous experiments $\mathbf{z}_c$ and $\mathbf{z}_i$ were quite small $(\mathbf{z}_c \approx 2 centemeters)$, therefore, $\hat{\mathbf{v}}_c$ was depending only on $\mathbf{v}_{ci}$ which is a measurement, and $\hat{\omega}$ which did suffer from considerable error.

Nonetheless, recalling that the the state of ROSPO is obtained from fusion between the IMU and Motion Capture measurements. Both of the measurements are based on a CoM location that

depends on the person's judgment. The method I have used to determine the actual location of CoM is prone to human error and there could be a possibility that the measurement suffers from some errors because of that. But the time of the internship was over before I was able to verify that.

Estimation of the Mass (Stage 11)

According to the theory, the estimation of the mass in each agent is a linear estimation problem shown in (5.14) since the system is similar to the one in (4.2), which, similarly, can be solved by using the recursive Least Square algorithm.

Re-writing (2.1) as

$$m\dot{\mathbf{v}}_c = n\mathbf{f}_{mean}, \tag{5.14}$$

Assuming n is assumed to be known. Further, $\mathbf{f}_{mean}$ is distributively estimated from $\mathbf{f}_i$ using dynamic average consensus algorithm (stage 7) 5.4, and also $\mathbf{v}_c$ is known locally by each robot i from (stage 10) 5.6.

Since The friction on the wheels is not negligible, the viscous friction was added to the linear system and was expected to improve the quality of the estimation of the mass, the modified system is shown in 5.15.

$$m\dot{\mathbf{v}}_c = n\mathbf{f}_{mean} + \beta_v\mathbf{v}_c, \tag{5.15}$$

$$\begin{aligned} \frac{y(t)-y(t-1)}{T_s} &= \frac{n}{m}u(t-1) + \frac{\beta v}{m}y(t-1) \\ y(t)-y(t-1) &= \frac{T_s n}{m}u(t-1) + \frac{Ts\beta v}{m}y(t-1) \\ y(t) &= \theta_1 y(t-1) + \theta_2 u(t-1) \end{aligned} \tag{5.16}$$

Where $\theta_1 = 1 + \frac{T_s\beta_v}{m}$ and $\theta_2 = \frac{T_s n}{m}$
And the similarly to Stage 3, the following Recursive Least Square can be used:

$$\begin{aligned} \mathbf{S}(t) &= \mathbf{S}(t-1) + \phi(t)\phi(t)^T \\ K &= \phi(t)/\mathbf{S}(t) \\ e &= y(t) - (\phi(t)^T\theta(t-1)) \\ \theta(t) &= K * e + \theta(t-1) \end{aligned} \tag{5.17}$$

Where $t = 1, 2, 3, ...$, $\phi = [y(t-1) \quad u(t-1)]$ and $\mathbf{S}$ can be initialized to $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

Results of the Linear Estimator

The mass estimation by agent 1 is shown in figure 5.15, the other agents produced similar results so there is no need to show them.

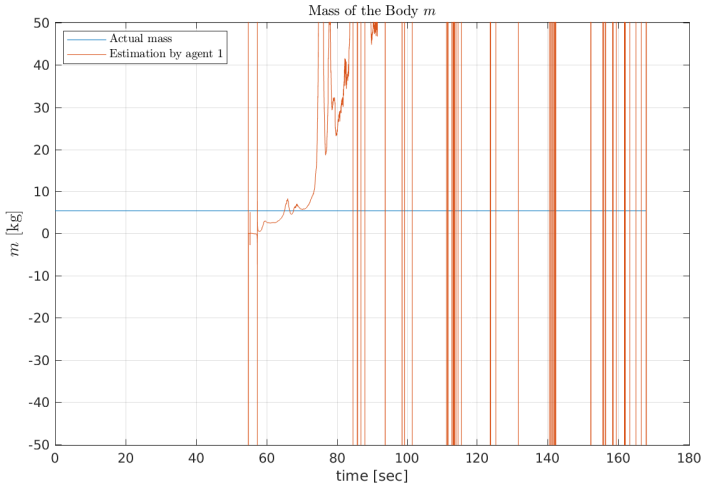


Figure 5.15: Body Mass Estimation by agent 1

As shown in the figures the estimation of the mass is not stable. This is due to the presence of friction in the wheels. To go around this issue a static friction model was included:

Friction Non-linear Model

Modeling the friction in the wheels: The static friction forces were also included in addition to the linear viscous friction to obtain a more realistic model.

$$m\dot{\mathbf{v}}_c = \begin{cases} \beta_v \mathbf{v}_c & if \quad n\mathbf{f}_{mean} + \beta_v \mathbf{v}_c - m\dot{\mathbf{v}}_c \leqslant K \\ n\mathbf{f}_{mean} + \beta_v \mathbf{v}_c - K sign(\mathbf{v}_c) & otherwise \end{cases} \tag{5.18}$$

Where $\beta_v \leq 0$ is the coefficient of viscous friction, and $K > 0$ is a constant that can be determined as follows.

By conducting an experiment of applying the forces of the turrets to the body only in x-direction or in y-direction (in Body Frame). The forces to be applied gradually until the body starts moving. The forces applied that made the body move can be used to compute the constant K. The outcome of the experiment is shown in figures 5.16 and 5.17.

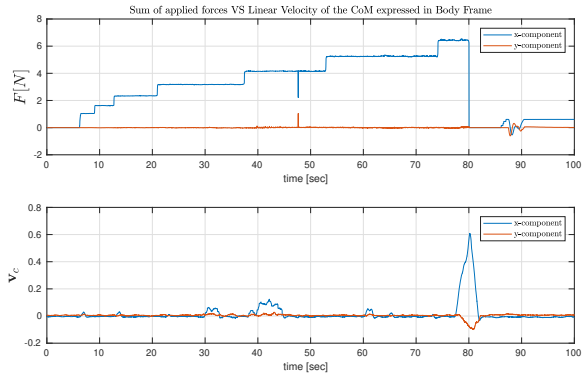


Figure 5.16: Sum of the applied forces and the Linear velocity of the platform expressed in Body Frame - moving in x-direction

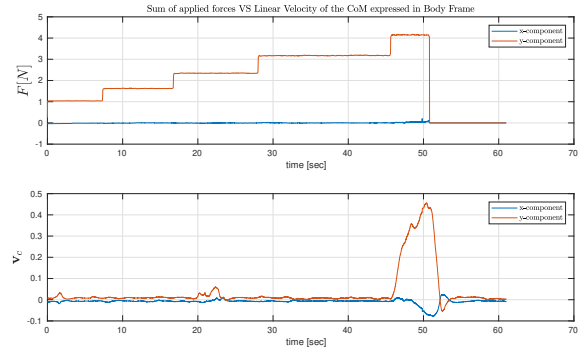


Figure 5.17: Sum of the applied forces and the Linear velocity of the platform expressed in Body Frame - moving in y-direction

For the force trajectory that was applied to the estimation algorithm, transforming the quantity $n\mathbf{f}_{mean}$ to Body Frame, showed that the amount of force required to move the platform against the

friction forces in x or y directions was not exceeded yet we saw movements on $\mathbf{v}_c$ plots, which shows that the model used does not describe the friction in the wheels properly.

In summary, the static friction is very high that large applied forces are needed to keep move the platform.

Applying large forces was not possible because of the low space in which the platform can move into, and other technical issues such as the presence of a limit on the electrical currents absorbed by the brush-less motors, and the Motion Capture is not able to precisely measure the state of the platform and the turrets while moving fast.

Comments on the Effect of the Friction on the Estimated Mass

One of the aims of the thesis is to try extend the estimation algorithm to include a realistic model of the friction of the load. From the previous subsection, it might be intuitive to suggest a solution of reducing the friction in the bearings, then apply the method mentioned in the previous section (i.e. Applying coordinated forces and taking the value that moves the body), but that is not applicable in a decentralized algorithm because it requires a central node to coordinate their applied forces. Even if it is found possible to apply locally available forces to achieve the task, although it is still considered decentralized, it is not the favorable direction to proceed.

Nevertheless, I was not able to reduce the friction in the wheels. The ball-bearing type is meant to be used to transform heavy objects, and is not designed to have minimum friction coefficients.

A possible solution (after finding a way to reduce the friction coefficient) is to use Hammerstein (or Weiner) model structures, which consist of a linear block, and an input (or an output) nonlinearity which is described by a polynomial model. Then apply a recursive estimation method to obtain the parameter values.

Discussions about the Dynamic Phase

- Due to model uncertainties in rotational and linear dynamics, complete experimental validation was not possible.
- Regarding the rotational dynamics, the friction in the wheels was reduced to some extent.
- If we change the model structure the non-linear observer will not guarantee achieving convergence, and we will need to change the observer and possibly changing the approach.
- Regarding the linear dynamics the friction was very high that prevented from obtaining a correct estimate even in a centralized manner.
- Judging The large error in estimating $\hat{\mathbf{v}}_c$ makes us say that it will not be suitable for the decentralized control algorithm. Simply because $\hat{\mathbf{v}}_c$ is not only used in the control law, but because its a controlled variable.

Chapter 6

Conclusion and Future Work

In summary and to the best knowledge of the author...

- It is necessary to implement also the decentralized control law along with the estimation to have a definite say about its validity, this is done through observing both (estimation + control) and evaluating the performance in the overall.
- The objectives of the thesis were not fully met mainly because the model uncertainties had more impact than anticipated, besides some technical issues that made it less possible to tackle them in all stages.
- It is recommended to find a way to reduce the friction in the wheels, then try to find a suitable way of nonlinear state estimation for a model that includes nonlinear friction.
- We can adopt a nonlinear model (Hammerstein/Weiner) to describe the linear dynamics, or to adopt a Linear Variable Parameter methods in estimation, along with adaptive control.
- With the current setup, it is possible to test most of the estimation algorithm (except stage 11 assuming we know the mass of the system), along with the proposed decentralized control law. However on expects that the system will not behave as desired because of the presence of the friction.
- Extending the work to 3D, is a higher goal after experimental validation of the algorithm.

Chapter 7

Appendix

Graph Theory Concepts

The basic idea of a consensus algorithm is to impose similar dynamics on the information states of each vehicle. If the communication network among agents allows **continuous communication** or if the communication bandwidth is sufficiently large, then the information state update of each vehicle is modeled using a **differential equation**. On the other hand, if the communication data arrive in **discrete packets**, then the information state update is modeled using a **difference equation** [24].

An average consensus algorithm with a scalar information state is over-viewed in the next paragraphs, in which a scalar information state is updated by each agent using, respectively, a first-order differential equation and a first-order difference equation.

Assuming n agents in the team. The team's communication topology can be represented by directed graph $\mathcal{G}_n \triangleq (\mathcal{V}_n, \mathcal{E}_n)$, where $\mathcal{V}_n = \{1, ..., n\}$ is the node set and $\mathcal{E}_n \subseteq \mathcal{V}_n \times \mathcal{V}_n$ is the edge set, figure 7.1 shows three different communication typologies for three vehicles. The communication topology may be time varying due to vehicle motion or communication dropouts. The form of a continuous-time consensus algorithm is given by [24]

$$\dot{x}_i(t) = -\sum_{j=1}^{n} a_{ij}(t)[x_i(t) - x_j(t)], \quad i = 1, ..., n \tag{7.1}$$

Where $a_{ij}(t)$ is the (i, j) entry of adjacency matrix $\mathcal{A}_n \in \mathbb{R}^{n \times n}$ associated with $\mathcal{G}_n$ at time t (see Appendix A) and $\mathbf{x}_i$ is the information state of the i th agnet. Setting $a_{ij} = 0$ denotes the fact that agent i cannot receive information from agent j. A consequence of (7.1) is that the information state $\mathbf{x}_i(t)$ of agent i is driven toward the information states of its neighbors. The critical convergence question is, when do the information states of all of the vehicles converge to a common value?

Although the equation (7.1) ensures that the information states of the team agree, it does not dictate a specified common value.

Further to (7.1), consider information states with single-integrator dynamics given by

$$\dot{\mathbf{x}}_i = \mathbf{u}_i, \quad i = 1, ..., n \tag{7.2}$$

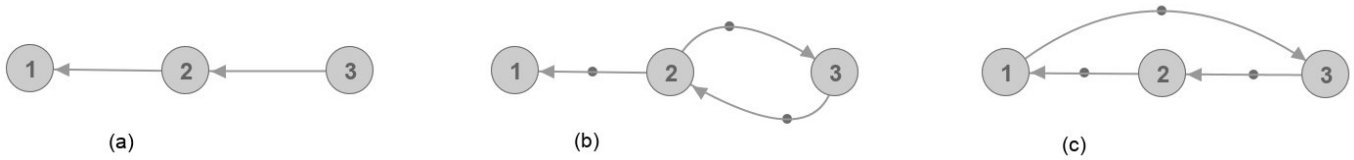


Figure 7.1: Three different communication typologies for three agents. Subplot (c) is strongly connected because there is a directed path between every pair of nodes. However, (a) and (b) are not strongly connected

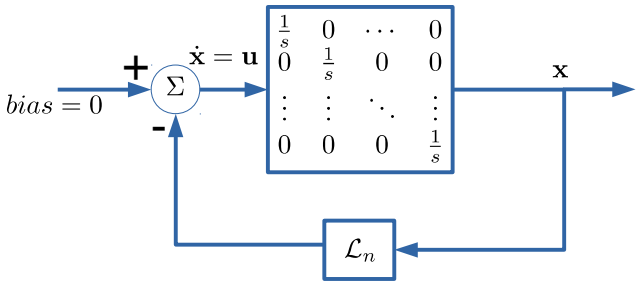


Figure 7.2: Block-diagram representation of the fundamental consensus algorithm applied on 1st order state dynamics

Where $\mathbf{x}_i \in \mathbb{R}^m$ is the information state and $\mathbf{u}_i \in \mathbb{R}^m$ is the information control input of the ith agent. A continuous-time consensus algorithm is given by

$$\mathbf{u}_i = \sum_{j=1}^{n} a_{ij}(t)(\mathbf{x}_i - \mathbf{x}_j), \quad i = 1, ..., n \tag{7.3}$$

$a_{ij}(t)$ is explained above. Equation (7.3) is written for one agent i. It can be written for all agents in compact form

$$\mathbf{u} = -\mathcal{L}_n \mathbf{x} \tag{7.4}$$

Equations (7.2) and (7.4) together make a closed-loop as shown in figure no **??** [25].

The consensus algorithm (7.3) is distributed in the sense that each agent needs only information from its neighbors.

From now on, we will consider only fixed communication typologies, which implies that the Adjacency $\mathcal{A}_n$ and the Laplacian $\mathcal{L}_n$ matrices are time-invariant.

From 7, the properties of the Laplacian $\mathcal{L}_n$ are directly related to the associated graph $\mathcal{G}_n$. Further, based on the Laplacian $\mathcal{L}_n$, the consensus algorithm converges if and only if the graph $\mathcal{G}_n$ is connected [24]. To prove that we can rewrite (7.3) and (7.4) as a single line: $\dot{\mathbf{x}} = -\mathcal{L}_n \mathbf{x}$, which is

a matrix differential equation that has the **explicit solution** of the consensus dynamics as follows [24]

$$\mathbf{x}(t) = e^{-\mathcal{L}_n t}\mathbf{x}_0 \tag{7.5}$$

Since $\mathcal{L}_n$ is a symmetric matrix, it can be nationalized by an orthogonal matrix U, such that $UU^T = I$. Therefore, $L = U\Lambda U^T$ where $\Lambda = diag(\lambda_i)$. Exploiting the latter we get $e^{-U\Lambda U^T t} = Ue^{-\Lambda t}U^T$.

Then we get the state evolution as $\mathbf{x}(t) = Ue^{-\Lambda t}U^T\mathbf{x}_0$. Which can be written as follows

$$\mathbf{x}(t) = \mathbf{u}_1\mathbf{u}_1^T e^{-\lambda_1 t}\curvearrowleft_0 + \sum_{i=2}^{n}\mathbf{u}_i\mathbf{u}_i^T e^{-\lambda_i t}\mathbf{x}_0 \tag{7.6}$$

Regarding the first term of (7.6) From the properties of $\mathcal{L}_n$, it's known that $\lambda_1 = 0$ and $u_1 = \frac{\mathbf{1}}{\sqrt{n}}$ Thus (7.6) becomes

$$\mathbf{x}(t) = \frac{(\mathbf{1}^T\mathbf{x}_0)\mathbf{1}}{n} + \sum_{i=2}^{n}\mathbf{u}_i\mathbf{u}_i^T e^{-\lambda_i t}\mathbf{x}_0 \tag{7.7}$$

Regarding the second term of (7.7), if $\mathcal{G}$ is connected, then $\lambda_2 \neq 0$ and $\lambda_n \geq ... \geq \lambda_2 > 0$, therefore the term will decay with time because the presence of negative exponents ($\lim_{t\to\infty}\mathbf{x}(t) = \frac{(\mathbf{1}^T\mathbf{x}_0)\mathbf{1}}{n}$). the latter term means that each agent state $\mathbf{x}_i, \quad \forall i$ is $\frac{\mathbf{1}^T\mathbf{x}_0}{n}$ the average of the initial state $\curvearrowleft_0$.

The result is that all the agent states $\mathbf{x}_i$ converge towards a common value, that is, the average of the inital state $\curvearrowleft_0$.

The rate of convergence is directly related to the value of λ_2 (called the degree of connectivity of the graph). This is evident from (7.7). The value of λ_2 (smallest eigenvalue in the sum) dictates the rate of the asymptotic decay of the sum of exponential function, accordingly if λ_2 is large, the exponential sum will decay faster. Moreover, λ_2 is a measure of how dense the graph is.

All the above discussion was related to **undirected graphs** in which we could use the symmetry property of $\mathcal{L}_n$. In the case of directed graphs, $\lim_{t\to\infty}\mathbf{x}(t) = \frac{(\mathbf{1}^T\mathbf{x}_0)\mathbf{1}}{n}$ is true when the directed graph is **balanced**. a directed graph is called balanced if, for every vertex, the in-degree equals the out-degree.

Bibliography

[1] A. Petitti, A. Franchi, D. Di Paola, and A. Rizzo, "Decentralized motion control for cooperative manipulation with a team of networked mobile manipulators," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 441–446, IEEE, 2016. (document), 1.4, 2.3, 2.3.2

[2] A. Franchi, A. Petitti, and A. Rizzo, "Distributed estimation for cooperative mobile manipulation," *CoRR*, vol. abs/1602.01891, 2016. (document), 1.4, 1.1, 2.3, 2.3.3

[3] M. D. Zivanovic and M. Vukobratovic, *Multi-arm cooperating robots: dynamics and control*, vol. 30. Springer Science & Business Media, 2006. 1.1

[4] R. R. Ma and A. M. Dollar, "On dexterity and dexterous manipulation," in *2011 15th International Conference on Advanced Robotics (ICAR)*, pp. 1–7, June 2011. 1.1

[5] M. Hvilshøj and S. Bøgh, ""little helper"—an autonomous industrial mobile manipulator concept," *International Journal of Advanced Robotic Systems*, vol. 8, no. 2, p. 15, 2011. 1.2

[6] T. R. Kurfess, *Robotics and automation handbook*. CRC press, 2004. 1.3

[7] A. Franchi, "Decentralized estimation and control for cooperative mobile manipulation." Presentation, 2016. 1.3

[8] A. Franchi, A. Petitti, and A. Rizzo, "Distributed estimation of the inertial parameters of an unknown load via multi-robot manipulation," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pp. 6111–6116, IEEE, 2014. 1.4

[9] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, and A. Casal, "Coordination and decentralized cooperation of multiple mobile manipulators," *Journal of Field Robotics*, vol. 13, no. 11, pp. 755–764, 1996. 2.2

[10] T. Fan, H. Weng, and T. Murphey, "Decentralized and recursive identification for cooperative manipulation of unknown rigid body with local measurements," *arXiv preprint arXiv:1709.01555*, 2017. 2.2

[11] A. Tsiamis, C. K. Verginis, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Cooperative manipulation exploiting only implicit communication," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 864–869, IEEE, 2015. 2.2

[12] Z. Wang and M. Schwager, "Force-amplifying n-robot transport system (force-ants) for cooperative planar manipulation without communication," *The International Journal of Robotics Research*, vol. 35, no. 13, pp. 1564–1586, 2016. 2.2

[13] C. K. Verginis, M. Mastellaro, and D. V. Dimarogonas, "Robust quaternion-based cooperative manipulation without force/torque information," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 1754–1759, 2017. 2.2

[14] A. Marino, G. Muscio, and F. Pierri, "Distributed cooperative object parameter estimation and manipulation without explicit communication," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 2110–21116, IEEE, 2017. 2.2

[15] H. Lee and H. J. Kim, "Constraint-based cooperative control of multiple aerial manipulators for handling an unknown payload," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 2780–2790, 2017. 2.2

[16] P. Culbertson and M. Schwager, "Decentralized adaptive control for collaborative manipulation," 2.2

[17] G.-B. Dai and Y.-C. Liu, "Distributed coordination and cooperation control for networked mobile manipulators," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 6, pp. 5065–5074, 2017. 2.2

[18] J. M. Esposito, "Decentralized cooperative manipulation with a swarm of mobile robots," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 5333–5338, IEEE, 2009. 2.2

[19] A. Marino, "A decentralized adaptive control for tightly connected networked lagrangian systems," in *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, pp. 4656–4661, IEEE, 2017. 2.2

[20] D. Prattichizzo and J. C. Trinkle, *Grasping*, pp. 955–988. Cham: Springer International Publishing, 2016. 2.3.1

[21] R. Aragues, L. Carlone, C. Sagues, and G. Calafiore, "Distributed centroid estimation from noisy relative measurements," *Systems & Control Letters*, vol. 61, no. 7, pp. 773–779, 2012. 4.7.1

[22] M. Zhu and S. Martínez, "Discrete-time dynamic average consensus," *Automatica*, vol. 46, no. 2, pp. 322–329, 2010. 5.4

[23] R. Hermann and A. Krener, "Nonlinear controllability and observability," *IEEE Transactions on automatic control*, vol. 22, no. 5, pp. 728–740, 1977. 5.5.1, 5.5.1

[24] W. Ren and R. W. Beard, *Distributed consensus in multi-vehicle cooperative control*. Springer, 2008. 7.1, 7.1

[25] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007. 7.1