On Aerial-Ground Co-manipulation:
The Tele-MAGMaS Project and Prospects of Vision Based State Estimation

An internship report submitted in partial fulfillment of the requirements for the degree of MASTER ARIA-ASI

by

Subodh MISHRA

under the guidance of

Dr. Antonio Franchi

Automatic Control, Robotics and Applied Informatics

École Centrale de Nantes

Certificate

It is certified that the work contained in this internship report entitled "On Aerial-Ground Co-manipulation: The Tele-MAGMaS Project and Prospects of Vision Based State Estimation" by "Subodh MISHRA" has been carried out under my supervision.

Dr. Antonio Franchi

August 2017

Permanent CNRS Researcher (CR1)

Robotics & InteractionS, LAAS-CNRS, Toulouse

Abstract

Name of the student: **Subodh MISHRA** Student No.: **150475K**

Degree for which submitted: **MASTER ARIA** Specialization: **Automatic Control**

Internship Title:

On Aerial-Ground Co-manipulation: The Tele-MAGMaS Project and Prospects of Vision Based State Estimation

Supervisor: **Dr. Antonio Franchi**

Month and year of report submission: **August 2017**

In this report I have presented the work I was involved with, during my internship at LAAS-CNRS, Toulouse. In the first three months of my stay, I was involved with the integration of various subsystems like the ground manipulator, the aerial manipulator, the haptic interface, the visualizer etc.for taking part in the KUKA Innovation Awards 2017 and the details about the overall project and these subsystems are well enumerated in the first few chapters of the report. My work revolved around the software integration and the details about the robotic software architecture used in the Tele-MAGMaS project is explained in a chapter dedicated to it. After the end of the competition I focused my interest and efforts in searching for alternatives to the motion capture system used for state estimation of the flying robot. To this end, a market survey of available vision sensors was done and the most suitable one was procured. The final part of my report describes the use of Visual SLAM for state estimation and the integration of a state of the art V-SLAM algorithm called the ORB_SLAM2 with the procured vision sensor.

Acknowledgements

I extend my sincerest gratitude to Dr. Antonio Franchi of LAAS-CNRS, Toulouse for hosting me at the Robotique et InteractionS (RIS) team for 6 months as an intern. I had an immensely productive time at LAAS-CNRS and got an opportunity to get my feet wet in the area of research that may potentially be a significant part of my doctoral work. Besides these, the environment at LAAS-CNRS is conducive for doing phenomenal research, thanks to the workaholic researchers, research engineers, graduate students and interns. It was great to be part of such a team of motivated people, striving hard to do extraordinary work.

I am also deeply thankful to the authorities at Centrale Nantes who allowed me to move to Toulouse for my end of study internship. Without their support this might not have been possible.

I am indebted to Nicolas for helping me at each and every step, right from the day I entered the LAAS-CNRS premises. I learnt a lot by working with him. His ways of finding quick intuitive solutions to seemingly daunting problems is something I will always try to inculcate within me. I am also thankful to Davide for allowing me to use the OTHex aerial robot and helping me to learn fly it.

I am also thankful to Anthony for always taking time to answer my emails in order to solve software related issues. Thanks to his help, I now know more about software architectures for robotics than I ever did. His patience in dealing with everyone's query is praiseworthy.

I will be failing in my duty if I don't thank Shubham for helping me with experiments and Andrea for all the valuable inputs he gave me about Visual SLAM and Visual Odometry.

I want to take this opportunity to thank Quentin for being a great friend and for staying late into the evenings to help me with 3-D printing some parts. The coffee sessions with him helped me relax and rejuvenate.

Last, but most importantly, I am thankful to my parents back in India who have instilled in me a love for science and a spirit of inquiry and inquisitiveness. But for there support and blessings, I wouldn't have come this far.

Contents

List of Figures

List of Tables

Abbreviations

Tele-MAGMaS	Human-in-the-loop **M**ulti–robot **A**erial–**G**round **M**anipulator **S**ystem
UAV	**U**nmanned **A**erial **V**ehicle
UGV	**U**nmanned **G**round **V**ehicle
USAR	**U**rban **S**earch **A**nd **R**escue
CoM	**C**enter **o**f **M**ass
MoCap	**Mo**tion **Cap**ture
VTOL	**V**ertical **T**ake **O**ff and **L**anding
DoF	**D**egrees **o**f **F**reedom
TiltHex	**Tilt**ed **Hex**arotor
OTHex	**O**pen **T**ilted **Hex**arotor
OS	**O**perating **S**ystem
ROS	**R**obot **O**perating **S**ystem
RGB-D	**R**ed **G**reen **B**lue- **D**epth
IMU	**I**nertial **M**easurement **U**nit
SfM	**S**tructure **f**rom **M**otion
SLAM	**S**imulataneous **L**ocalization **A**nd **M**apping
V-SLAM	**V**isual-**SLAM**
VO	**V**isual **O**dometry
DH	**D**enavit **H**artenberg
UDP	**U**ser **D**atagram **P**rotocol
TCP	**T**ransmission **C**ontrol **P**rotocol
IP	**I**nternet **P**rotocol
GenoM	**Gen**erator **o**f **M**odules

ROS	**R**obot **O**perating **S**ystem
USB	**U**niversal **S**erial **B**us
LBR	**L**eicht**b**au**r**oboter (German for lightweight robot)
iiwa	**i**ntelligent **i**ndustrial **w**ork **a**ssistant
HTTP	**H**yper **T**ext **T**ransfer **P**rotocol
GPS	**G**lobal **P**ositioning **S**ystem
LiDAR	**Li**ght **D**etection **A**nd **R**anging
IR	**I**nfra-**R**ed
ORB	**O**riented FAST and **r**otated **B**RIEF
FAST	**F**eatures from **A**ccelerated **S**egment **T**est
BRIEF	**B**inary **R**obust **I**ndependent **E**lementary **F**eatures
KF	**K**alman **F**ilter
EKF	**E**xtended **K**alman **F**ilter
BA	**B**undle **A**djustment

Chapter 1

Introduction

1.1 Project Overview

This master thesis report is based on the on-going research on Cooperative Manipulation using a fleet of Aerial and Ground Robots at LAAS-CNRS, Toulouse. Because of the variety of robots involved, the constraints of the workspace, the limitations of actuators, the size and shape of the manipulandum, hardware and software issues, etc. co-operative manipulation is one of the most challenging problems in the field of mobile robotics.

Cooperative Aerial Ground Manipulation requires expertise in all the domains of Robotics like mechanical design, control systems engineering, system modelling, state estimation, signal processing, programming and software engineering, etc. . Although cooperative manipulation using only Aerial or only Ground Robots is not new, the use of both Aerial and Ground Manipulators is a novelty that commends itself.

This project is called the **Tele-MAGMaS** which is the acronym for Human-in-the-loop **M**ulti–robot **A**erial–**G**round **Ma**nipulation **S**ystem . The Tele-MAGMaS concept is composed of three constituent elements:

1. A ground robot manipulator, the master manipulator.
2. A fleet of multirotor UAVs to aid the ground robot manipulator.
3. A human operator remotely supervising the entire system with an haptic feedback device.

The Aerial Robots help the ground manipulator by:

1. Augmenting its achievable task-space.
2. Reducing body vibrations in the manipulandum in case of long and flexible object, thereby reducing the torques extered on the joints of the ground manipulator.

In complex and unpredictable scenarios like Urban Search and Rescue (USAR), a human operator is required in the field to tele-operate the system and/or to command minor course correction in the trajectories of the robots involved. An human operator/supervisor is highly recommended because the objective of the USAR missions is to save precious human lives.

1.2 Motivation and Objectives

USAR is a field where the significance of planning, control, and human robot interaction is of utmost importance. A human in loop can react faster to unforeseen situations. At times, the manipulandum in a site may not be small enough to be grasped from its center of mass (CoM) by the ground manipulator and lifting it from just one end may result in severe vibrations in the manipulandum which may force the ground manipulator to operate in the verge of its joint torque limits but these oscillations can be avoided if the other end of the manipuladum is supported by an aerial robot. Also, if a large object is grasped from the center then the reachable taskspace of the ground manipulator will reduce. The Aerial Robots involved are VTOL systems, which can be quadrotors, helicopters or any generic multirotor system. This heterogenous multi-robot team operating in remote site/disaster scene, will be controlled/supervised as whole by a human operator through a multi degree of freedom actuated haptic device. The human operator is provided with haptic device which can send/receive signals to/from the multi robot system. It can be either used to command the fleet of robots or to get situational awareness or both.

1.3 Intended Use Case

Every year, there are countless number of human and natural disasters all over the world. Most of the times there are victims trapped under debris and the ruins or in a building and getting them out safely, without causing loss of lives of the rescue workers and the victims

is a challenging operation. Oftentimes, in addition to the victims, the rescue workers have also lost their lives. Thanks to the advancement of technology, the number of such deaths have reduced significantly. Human robot teaming in USAR is a solution to save more human lives. The world trade center disaster was the first successful known of the human-robot teaming in USAR [1].

So far, the core of the rescue robotics has been mobile manipulators and only recently VTOL-UAVs have been used in USAR for visual inspection. However, the potential capability of VTOLs is beyond visual inspection and sensing. They can be utilized for aerial transportation or to act on their environment using passive tools, or robotic manipulator installed on them, and making them fully mature aerial robots. A fleet of VTOL-UAVs cooperatively manipulating an object has been presented in [2].

When dexterous manipulation by VTOL-UAV is not needed, a lightweight passive arm is preferred over an active arm manipulator for the following reasons:

1. Using a 1-DoF gripper mechanism, the robot can use lightweight passive arm for holding and moving objects.
2. A light weight passive arm has a small payload footprint, thus less energy is required and the flight time increases significantly.
3. A passive arm has a simpler mechanical structure compared to an actuated one , so, instead of designing a special vehicle to carry a robotic arm, one can keep the existing platform and integrate the simple passive arm to it.
4. The flying robot used by us is fully actuated and can track a 6-DoF pose, hence the full actuation of the flying platform can compensate for the passivity of the arm.

Thus, using passive arms is much more cost efficient in terms of design, development and maintenance but it also depends on the type of flying platform employed.

However, ground manipulators (stationary or mounted on a mobile robot) usually have significantly higher payload capacities than typical small sized VTOL-UAVs, thus ground manipulators will remain the core of rescue robotics. The goal of this project is to use VTOL-UAVs to help the robot manipulator to overcome the aforementioned issues of: limited reachable taskspace and ill effects of vibrations on the ground robot. In industrial settings also, the ground robot manipulator could benefit from the extra dexterity and capabilities induced by a team of flying manipulators. The vibrating manipulandum may

exert high torques on the ground manipulator and the aerial manipulator acts as a flying companion by limiting these vibrations, thereby limiting the joint torques on the ground manipulator.

Rescue robotics has always involved a human operator for driving and controlling of different functionalities of the robot through visual feedback or haptic feedback. The haptic feedback gives the human operator situational awareness of the remote site and also helps the operator to send control signals.

1.4 Objectives

This ambitious project has the following objectives:

A: To design new bilateral control feedback strategies in order to effectively realize a working Tele-MAGMaS system, that will be able to:

- A.1 Compensate the limited joint torques of the ground robot manipulator.
- A.2 Expand the accessible workspace of the ground robot manipulator in case of tool-operation.
- A.3 Reduce vibrations in case of large flexible objects.

B: To build a new generation of lightweight passive gripper for small sized aerial robots. This objective consists of:

- B.1 Finding best connection mechanism between the passive gripper and the aerial vehicle which can be a fixed connection, a universal joint, a spherical joint, or a compliant spherical joint, depending on the application.
- B.2 Finding the best shape, mechanical structure, and material for building the passive lightweight grippers, which could also change depending on the application.
- B.3 Finding the best gripping mechanism depending on the application, considered solution are a unilateral contact, an active vacuum gripper, or a single degree of freedom gripper.

C: To provide the necessary infrastructure in order to practically exploit Tele-MAGMaS in real situations.

D: To prepare the scientific and practical basis to go beyond tele-operating the Heterogenous-Multi-Robot system as whole, and using shared controlled strategies, in which the Robot team will utilize some level of autonomy along with the human operator.

1.5 Resources at LAAS

To accomplish the goal of this complex project a variety of hardware and associated software is necessary. Each of the component has its own software and there has to be a middleware to relay information from one system to another and to create a common interface for message passing. The variety of robots involve the use different software which makes this project highly complex from both hardware and software point of view.

The list of hardware used is as follows:

- KUKA LBR iiwa 14 R820 with flexFellow carrier.
- Open Tilt Hex (OTHex) VTOL.
- omega.6 Haptic device.
- Vision Sensor System.
- OptiTrack: Motion Capture Tracking system for indoor testing.

And the list of software used is as follows:

- KUKA Controller: KUKA sunrise OS or KUKA Fast Research Interface.
- ROS: as a middleware.
- Matlab/Simulink: is used to implement part of the control scheme.
- Middleware independent GenoM3 software developed at LAAS-CNRS.
- GenoM3 components for various subsystems.
- Force dimension sdk: to control the haptic device.
- Optpp: An open source tool used for solving optimization probems.
- Motive: software used for indoor position by OptiTrack motion capture system.
- V-Rep for visualization and simulation.

Chapter 2

Literature Survey

"Cooperative Aerial-Ground Manipulation" is a challenging research problem. In fact, cooperative manipulation is very common place in assembly lines of manufacturing industries like automobiles, heavy electrical equipments, defence and space industry, etc. Some of the oldest references on cooperative manipulation are in the field of cooperative robotic assembly. An approach to control multiple robot arms engaged in cooperative manipulation and assembly operations is shown in [3]. The technique uses transducers to measure constraint reactions created in the resulting closed-chain multirobot configuration by relative trajectory errors between manipulators. A structural analysis formulation is then employed to determine the magnitude of these errors, followed by active control compensation to eliminate them.

Although a number of Robotics labs around the world have presented their research on cooperative manipulation, their work has primarily involved homogeneous robots which makes their control and planning simple as compared to the case when heterogenous team of robots is used. Controlling a group of ground or aerial mobile robots is simpler when compared to controlling a fleet comprising both aerial and ground robots. Some good references on cooperative manipulation using ground mobile robots are [4] and [5] . In [4] the authors present two control methodologies: a partially decentalized controller, in which each robot computes the control law locally, but it has an a priori knowledge of the physical parameters of the load, and a fully decentralized approach in which each robot does not have any information about the load. In both the cases the robots do not have access to any information about the state of the other robots in the team. On the topic of cooperative aerial manipulation, [6], [7] and [8] are compelling first reads.

The authors of [6] have shown that autonomous micro aerial robots can operate in three-dimensional unstructured environments, and offer many opportunities for environmental monitoring, search and rescue, and first response and also discuss the deployment of large numbers of aerial robots, focusing on the control and planning problems with applications to cooperative manipulation and transport, construction, and exploration and mapping. The kinematics of cooperative transport of payloads suspended by multiple aerial robots with cables is shown in [7]. It also presents case studies with an equilateral triangle payload and a general payload for demonstration. The flying hand, presented in [8], is a robotic hand consisting of a swarm of UAVs, able to grasp an object where each UAV contributes to the grasping task with a single contact point at the tooltip, mimicking the finger behaviour. The swarm of robots is tele-operated by a human hand whose fingertip motions are tracked, e.g., using an RGB-D camera. The cooperative use of a UAV and a UGV for turning a lever in an industrial setting is presented in [9]. This work presents the results of efforts to build a heterogeneous robotic system capable of executing complex disaster response and recovery tasks. The authors of [9] explore high level task scheduling and mission planning algorithms that enable various types of robots to cooperate together, utilizing each others strengths to yield a symbiotic robotic system.

The Tele-MAGMaS project involves use of an aerial robot and a robot arm mounted on a mobile platform to cooperatively manipulate a long flexible rod with a human in loop to supervise and control the complete system. LAAS-CNRS has developed core competencies to successfully coordinate and complete this project which involves cooperation of IRISA-Rennes, University of Siena and SNU-South Korea. Researchers from the above institutions have described the successful design and testing of different tele-operation schemes for VTOL swarms in free flight ([10] and [11]). A unified frame-work that allows letting the group of UAVs autonomously control its topology in a safe & stable manner and suitable incorporation of some skilled human operators in the control loop is presented in [10]. This way, the human's superior cognitive capabilities and precise manual skills can be exploited as a valid support for the typical autonomy of a group of UAVs. A novel decentralized control strategy for bilaterally tele-operating heterogeneous groups of mobile robots from different domains (aerial, ground, marine, and underwater) is proposed in [11]. By using a decentralized control architecture, the group of robots, which is treated as the slave side, is made able to navigate in a cluttered environment while avoiding obstacles, inter robot collisions, and following the human motion commands. Simultaneously, the human operator acting on the master side is provided with a suitable force feedback informative of the group response and of the interaction with the surrounding environment. A hierarchical control framework for multiple cooperative quadrotor-manipulator systems is

proposed in [12], which allows to endow the common grasped object with a user-specified desired behavior (e.g., trajectory tracking, compliant interaction, etc.). A new aerial tool operation system consisting of multiple quadrotors connected to a tool by spherical joints to perform tool operation tasks is proposed in [13]. This work shows that the attitude dynamics of each quadrotor is decoupled from the tool dynamics, so that one can consider the quadrotors as thrusters and control the tool by adjusting the orientation and magnitude of these thrusters.

Successful design and testing of different bilateral tele-operation schemes for a single VTOL in contact with moving and fixed object to see to what extent a VTOL-UAV equipped with a passive tool is reliable in generating different desired force vectors is presented in [14] and [15]. The problem of a quadrotor that physically interacts with the surrounding environment through a rigid tool is considered in [14]. The authors present a theoretical design that allows to exert an arbitrary 3D force by using a standard near-hovering controller that was originally developed for contact-free flight control. A novel teleoperation framework for aerial robots that physically interact with the environment is presented in [15]. This framework allows to tele-operate the robot both in contact-free flight and in physical contact with the environment in order, e.g., to apply desired forces on objects of the environment. The framework is build upon an impedance-like indirect interaction force controller that allows to use standard under-actuated aerial robots as force effectors. Haptic feedback from the master side enables the user to feel the contact forces exerted by the robot on the environmnet.

Very promising results of experiments of cooperative manipulation using a KUKA Light Weight Robot and a Quadrotor UAV at LAAS-CNRS, have been presented in the paper [16] which was published in ICRA 2017. In this paper the authors lay the foundation of the first heterogeneous multi-robot system of the Multiple Aerial-Ground Manipulator System (MAGMaS) type. A MAGMaS consists of a ground manipulator and a team of aerial robots equipped with a simple gripper manipulating the same object. The idea is to benefit from the advantages of both kinds of platforms, i.e., physical strength versus large workspace. This paper forms the basis of the Tele-MAGMaS project on which this internship report is based. The work presented in [16] lucidly elucidates the pros and cons of homogenous multi robot systems and presents the idea of using both Aerial and Ground manipulators. Aerial manipulators augment the workspace of the robot and ground manipulators have higher payload carrying capacity. The idea is to combine the advantages of both the classes of robots. The work presented in [16] has two fold contribution, firstly, it presents the derivation of the model of a robotic system composed of a manipulator and

a fleet of aerial robots and secondly it proposes a nonlinear control scheme, that considers existing uncertainties and constraints of the system.

Since much of the testing and experimentation for the Tele-MAGMaS project was done indoors, the optitrack motion capture system was used for estimating the pose of the UAV and other objects in the flying arena. The secondary objective of the internship is to use a vision sensor on board the UAV, instead of using the motion capture for pose estimation of the UAV. The Motion Capture (MoCap) system works seamlessly indoors but it is not usable outdoors. Even indoors, the MoCap system requires a higher bandwidth and it practically makes it impossible to completely operate the UAVs wirelessly. What follows next is a brief review of state of the art methodologies for using vision based sensors for pose estimation, i.e. position and orientation of bodies. The choice of sensors for pose estimation is of crucial importance and it depends on a variety of factors. For flying robots meant for aerial manipulation, the sensors must be lightweight to provide room for the weight of the manipulandum. Laser scanners provide reliable measurements but are rather heavy for flying robots. Besides, they also consume more power when compared to passive sensors like cameras. Time-of-flight cameras are light and work outdoors but have a limited resolution. Cameras are lightweight and provide high resolution, hence, they are suitable for an application like ours.

The estimation of pose of a mobile robot with a vision based sensor is related to the problem of Structure from Motion (SfM), Visual-Simulataneous Localization and Mapping (V-SLAM) and Visual Odometry. VO is a subset of V-SLAM and V-SLAM is a subset of SfM. According to [17], SfM is more general than VO and tackles the problem of 3D reconstruction and 6DOF pose estimation from un-ordered image sets. VO is a particular case of SfM and focuses on estimating the 3D motion of the camera sequentially (as a new frame arrives) and in real time. VO is V-SLAM before closing the loop and it is a building block of V-SLAM. While VO aims at local consistency of the trajectory, V-SLAM aims at global consistency of the trajectory and the map. The choice between VO and V-SLAM purely depends on the trade-off between performance and consistency. VO trades off consistency for real-time performance, without the need to keep track of all the previous history of the camera. Since, we are concerned about real time implementation of SfM, we will explore the possibility of using various open sourced and out of the box VO and V-SLAM algorithms in the Tele-MAGMaS project.

Vision sensors are of three basic types, monocular, stereo and RGB-D. For doing VO or V-SLAM using monocular camera, there are a lot of open source packages available, most of them have a ROS wrapper as well, which makes their integration even simpler.

They are, PTAM (Parallel Tracking and Mapping), DSO (Direct Sparse Odometry), LSD SLAM (Large Scale Direct Monocular SLAM), ORB SLAM, Semi-direct monocular Visual Odometry (SVO SLAM). For performing VO or V-SLAM using stereo and RGB-D cameras, packages like OpenCV RGBD Odometry (Visual Odometry based on RGB-D images), Dense Visual Odometry and SLAM (DVO_SLAM) for RGB-D Cameras, ORB_SLAM2 for RGB-D and Stereo cameras, Robust Visual Inertial Odometry (rovio) for Stereo Cameras, etc. can be used. An experimental evaluation of several RGB-D VO methods is given in [18].

As discussed in [19], V-SLAM algorithms can be implemented by using just a cheap and easily available monocular camera, but, as the depth is not observable from just one camera, the scale of the map and estimated trajectory is unknown. Besides, the system needs to make an initial map of the environment which cannot be obtained from just one view. In addition, monocular V-SLAM suffers from scale drift and often fails when performing rotations [19]. Using Stereo or RGB-D sensors ameliorates these problems. Although the fundamental working of Stereo and RGB-D based algorithms are same, the hardware used is different. In terms of technical complexity a Stereo Camera is simpler but in RGB-D cameras, a better estimate of depth can be obtained. There are a large number of RGB-D cameras available, but in general they cannot be used outdoors because the Infra-Red pattern emitted by the projector is washed away by sunlight. The Embedded Infrared Assisted Stereovision 3D Imaging System with Color Camera Technology exploits the positive attributes of both Stereo and RGB-D systems. It consists of an infrared laser projection system, two infrared cameras, a color camera and a fisheye camera. The depth video stream is generated with active stereo vision technology assisted by the infrared laser projector and the two infrared imaging sensors. Color data is provided by a color camera. This techonlogy is marketed and promoted by Intel and it is called the RealSense SLAM.[1]

[1] `https://software.intel.com/sites/products/realsense/slam/`

Chapter 3

The Ground Manipulator

In this Chapter the details about the ground manipulator is presented. A brief description of the system parameters, its software architecture and a brief idea about its control methodology is elucidated.

Since LAAS-CNRS was one of the finalists of the KUKA-Innovation Awards 2017, KUKA provided us a KUKA LBR iiwa 14 R820 with flexFellow carrier. LBR stands for "Leichtbauroboter" (German for lightweight robot) and iiwa is the acronym for "intelligent industrial work assistant". The number "14" in the name of the robot refers to its payload which is 14 Kg and the number "820" refers to the maximum working radius in the horizontal plane which is 820 mm (Figure 3.5). This robot is a flexible and versatile solutions in industrial assembly operations. There is an option of Human Robot Collaboration (HRC) which enables special safety modes of operation and there is no need to put any fence around the robot to ensure the safety of the human operator.

3.1 Description of the LBR iiwa

The major subsystems of the robot is shown in Figure 3.1. Refering to the labelling in Figure 3.1, the following are the subsystems:

1. Connecting cable to the smartPAD.
2. KUKA smartPAD control panel.
3. Manipulator.

4. Connecting cable to KUKA Sunrise Cabinet, the robot controller.

5. KUKA Sunrise Cabinet robot controller.

The KUKA LBR iiwa 14 R820 with the flexFellow can be seen in Figure 3.2. The flexFellow is a mobile base which makes it easier for the operator to displace it from one place to another without much duress.

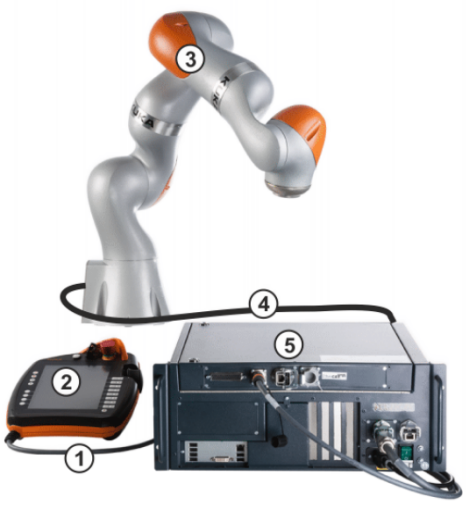

FIGURE 3.1: Overall System of KUKA LBR iiwa 14 R820: 1. Cable connecting to smartPAD, 2. KUKA smartPAD control panel, 3. Manipulator, 4. Connecting cable to KUKA Sunrise Cabinet, 5. KUKA Sunrise Cabinet robot controller

FIGURE 3.2: KUKA LBR iiwa 14 R820 with flexFellow

The KUKA LBR iiwa 14 R820 is a lightweight robot and has 7 revolute joints. The motor drive unit and the current carrying cables are all inside the robot body. Each joint has suitable sensors that provide the required signals for robot control. The joints also have sensors which let the user know if the robot is performing under its safety envelope, for examples, joint angle and torque limits, temperature limits, etc.. The main assemblies of the KUKA LBR iiwa 14 R820 can be found in the Figure 3.3. The part labelled 1 refers to the In-line wrist, the components marked 2 refer to joint modules and 3 refers to the base frame of the robot.

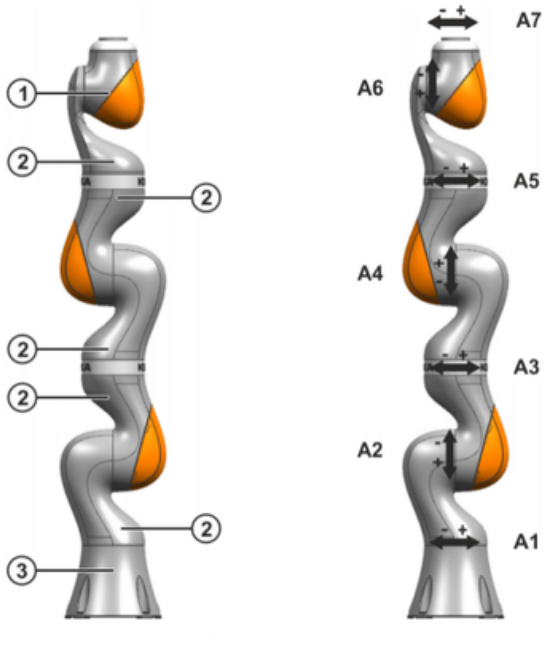

FIGURE 3.3: KUKA LBR iiwa 14 R820 Main assemblies and robot axes

The following is a brief description of the In-line wrist, joint module and base frame.

1. **In-line wrist:** The robot is fitted with a 2 axis in-line wrist. The motors are located in axes A6 and A7.

2. **Joint module:** The joint module consist of an aluminum structure. The drive units are situated inside these modules. In this way, the drive units are linked to one another via aluminum structures.

3. **Base frame:** The base frame is the base of the robot. It constitutes the interface for the connecting cables between the robot, the controller and the energy supply system.

3.2 System Parameters

Key parameters of the robot are the link length, as listed in Tables 3.1 and 3.2, because these parameters are necessary for deriving the kinematic Jacobian of the robot which is required for designing controllers for the robot both in the joint- and the task-space.

From	To	Distance(mm)
Base	Joint 2	360
Joint 2	Joint 4	420
Joint 4	Joint 6	400
Joint 6	Media Flange	187

TABLE 3.1: Dimensions of KUKA LBR iiwa 14 R820

Joint	Range of Motion (°)	Speed ($°/s$)	Torque (N m)
1	±170	85	320
2	±120	85	320
3	±170	100	176
4	±120	75	176
5	±170	130	110
6	±120	135	40
7	±175	135	40

TABLE 3.2: Joint, Speed and Torque Limits of KUKA LBR iiwa 14 R820

The working envelope of the robot is shown in Figures 3.4 and 3.5. These workspace limits were kept in mind while designing trajectory generator for KUKA LBR iiwa 14 R820.

To describe translational and rotational relationships between adjacent links, the DH parametrization is used. The DH representation results in a 4×4 homogenous transformation matrix representing each link's coordinate system. Thus, through sequential transformations, the end effector (EE) coordinates can be transformed and expressed in the base coordinates which make up the inertial frame of the dynamic system.

The DH parameterization described is explained in detail in [20].

Every joint coordinate frame is determined and established on the basis of three rules:

1. The z_{i-1} axis lies along the axis of mothon of the i^{th} joint.
2. The x_i axis is normal to the z_{i-1} axis and pointing away from it.
3. The y_i axis completes the right-handed coordinate system as required.

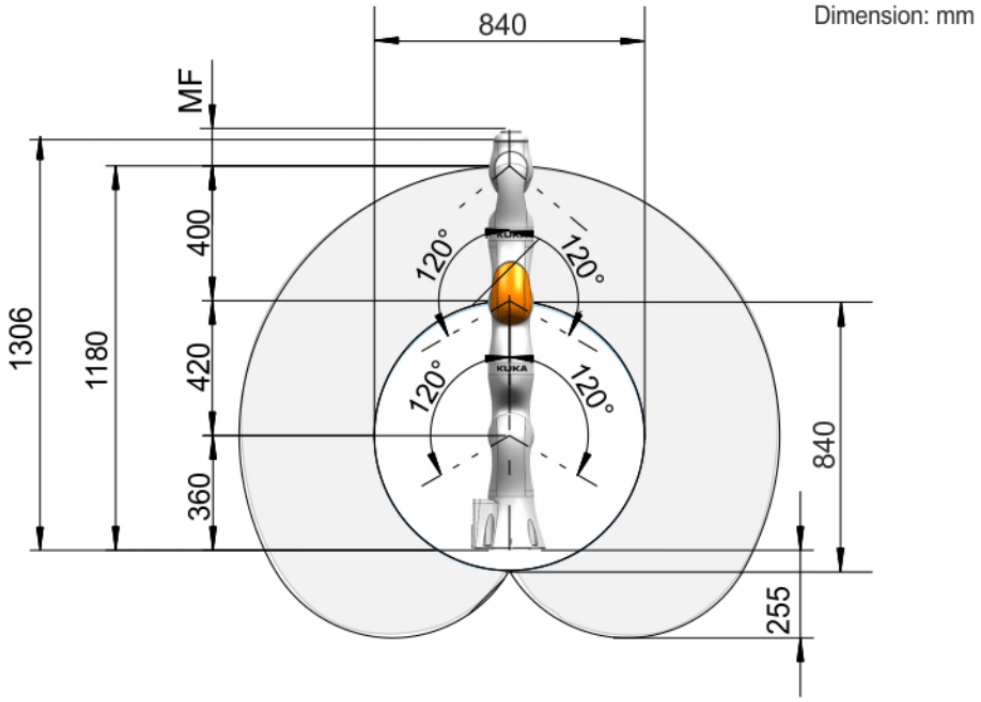

FIGURE 3.4: KUKA LBR iiwa 14 R820 working envelope, side view, MF=187 mm [Touch Electric Media Flange]

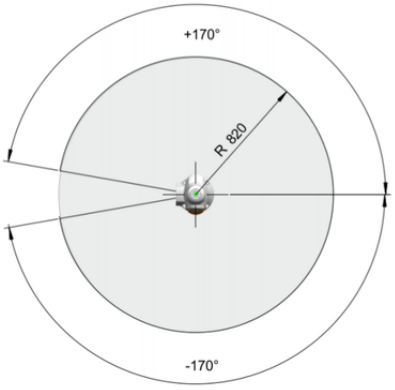

FIGURE 3.5: KUKA LBR iiwa 14 R820 working envelope, top view

The DH representation of a rigid link depends on four geometric parameters associated with each link. These four parameters completely describe any revolute or prismatic joint.

The parameters are:

- θ_i is the joint angle from the $\mathbf{x}_{i-1}$ axis to the $\mathbf{x}_i$ axis about the $\mathbf{z}_{i-1}$ axis (using the right-hand rule).

d_i is the distance from the origin of the $(i-1)th$ coordinate frame to the intersection of the $\mathbf{z}_{i-1}$ axis with the $\mathbf{x}_i$ axis along the $\mathbf{z}_{i-1}$ axis.

a_i is the offset distance from the intersection of the $\mathbf{z}_{i-1}$ axis with the $\mathbf{x}_i$ axis to the origin of the ith frame along the $\mathbf{x}_i$ axis (or the shortest distance between the $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$ axes).

α_i is the offset angle from the $\mathbf{z}_i$ axis to the $\mathbf{z}_i$ axis about the $\mathbf{x}_i$ axis (using the right-hand rule).

The Table 3.3 gives the DH representation of the KUKA LBR iiwa 14 R820. The DH parameters are required to find transformations between joints and is fundamental to calculation of the kinematic Jacobian matrix.

Joint i	θ_i	d_i	a_i	α_i
Joint 1	θ_1	d_1	0	-90°
Joint 2	θ_2	0	0	90°
Joint 3	θ_3	d_3	0	90°
Joint 4	θ_4	0	0	-90°
Joint 5	θ_5	d_5	0	-90°
Joint 6	θ_6	0	0	90°
Joint 7	θ_7	d_7	0	0°

TABLE 3.3: Denavit-Hartenberg Representation of KUKA LBR iiwa 14 R820

For the KUKA LBR iiwa 14 R820, θ_is of the Table 3.3 are joint angles and $d_1 = 0.36$ m, $d_3 = 0.42$ m, $d_5 = 0.4$ m and $d_7 = 0.187$ m.

The transformation between $(i-1)^{th}$ frame and i^{th} frame is given by:

$$ {}^{i-1}A_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

To verify the veracity of the DH parameterization table a simplified visualizer was created in MATLAB. The simplified model for joint angles = [30° 30° 0° −30° 0° 60°0°] is shown in Figure 3.6.

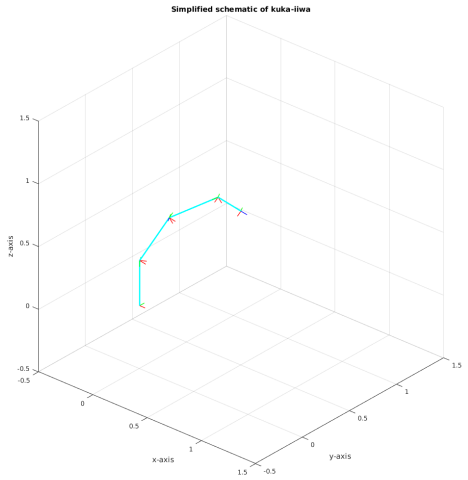

FIGURE 3.6: KUKA LBR iiwa 14 R820 simplified MATLAB model

3.3 Software Architecture

The overall software architecture of the KUKA iiwa 14 R820 is shown in Figure 3.7. The robot comes with a Sunrise cabinet which contains two computers, one running Sunrise Operating System and the other running VxWorks real time OS. A ground PC communicates with these two computers using the TCP/IP protocol or UDP protocol depending on the programming API. When programming in JAVA using the Java Editor provided by KUKA RoboticsAPI, the ground PC communicates with the computer running Sunrise OS using the TCP/IP protocol while the gripper control is communicated using UDP. When programming in C++ using the Fast Research Interface (FRI), the ground PC communicates with the computer running VxWorks using UDP. The operator can separately control the robot using the teach pendant which is connected to the computer running Sunrise OS. The manipulator is connected to the computer running VxWorks because VxWorks is a real time operating system and it allows the control of the manipulator in real time.

The hardware that incorporates the software architecture given in Figure 3.7 is shown in 3.8. As can be seen, ① is the ground PC, ② is the sunrise cabinet, ③ is the manipulator and ④ is the teach pendant.

The teach pendant is only required in the start-up phase for tasks which for practical or safety reasons cannot be carried out using programming environments, e.g. for calibration

and teaching points. After start-up and application development, the operator can carry out simple servicing work and operating tasks using the teach pendant.

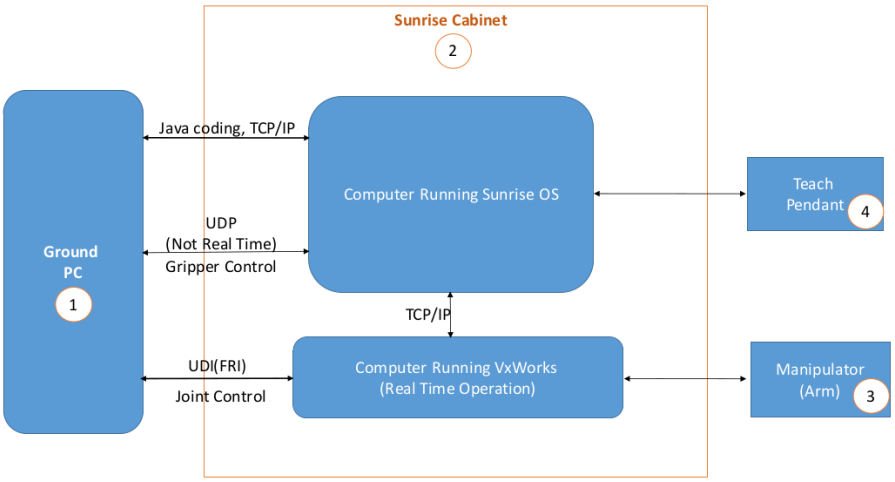

FIGURE 3.7: KUKA LBR iiwa 14 R820 software architecture

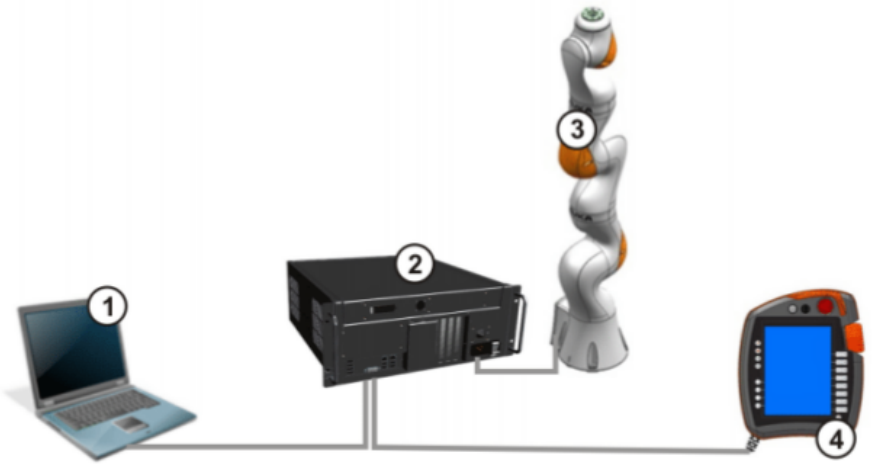

FIGURE 3.8: KUKA LBR iiwa 14 R820: Separation of operator control and programming, 1. Ground PC, 2. The Sunrise Cabinet, 3. The Manipulator and 4. The Teach Pendant (The KUKA smartPAD control panel)

In our project, the high level control laws were implemented in MATLAB/SIMULINK in the ground PC and these control inputs were communicated to the iiwa-GenoM component[1] using the genomix server which acts as a bridge between MATLAB and the GenoM component. Note that the iiwa-GenoM component is also on the ground PC. The iiwa-GenoM component is programmed in C++ and it implements joint level control and also a

[1]More information about GenoM components can be found in Chapter 6.

module to compute the inverse Kinematics so that the robot can be controlled in cartesian space. The joint level control was done using the Fast Research Interface (FRI). The low level control commands generated by the iiwa-GenoM component are sent to the computer running VxWorks in the KUKA Sunrise Cabinet via UDP communication protocol, this computer sends control commands to the manipulator.

3.4 Robot Control

Since we interact with the environment of the robot, an admittance control framework is implemented to make the motion compliant to external wrench. In order to understand the basic problem of robot control, it is useful to recall the dynamic model (Equation 3.1) whose general form for a robot with n degrees of freedom is the following:

$$\Gamma = A(q)\ddot{q} + C(q,\dot{q})\dot{q} + Q(q) \tag{3.1}$$

where,

- Γ is a vector of joint torques or forces, depending on whether the joint is revolute or prismatic respectively. In the sequel, this vector is mentioned as joint torque vector. It is the input to the robot.
- q is the joint position, $\dot{q}$ is the joint velocity and $\ddot{q}$ is the joint acceleration, each having the dimension $n \times 1$.
- $A(q)$ is $n \times n$ inertia matrix
- $C(q,\dot{q})\dot{q}$ is $n \times 1$ vector of coriolis/centrifugal torques
- $Q(q)$ is $n \times 1$ vector of gravity torques

The Control Block diagram is shown in Figure 3.9, where:

- x_{des} is the user defined trajectory.
- f_{ext} is the external disturbance.
- x_c and $\dot{x}_c$ are the compliant trajectory's position and velocity respectively.
- q_c and $\dot{q}_c$ are compiant joint position and velocity respectively corresponding to x_c and $\dot{x}_c$.

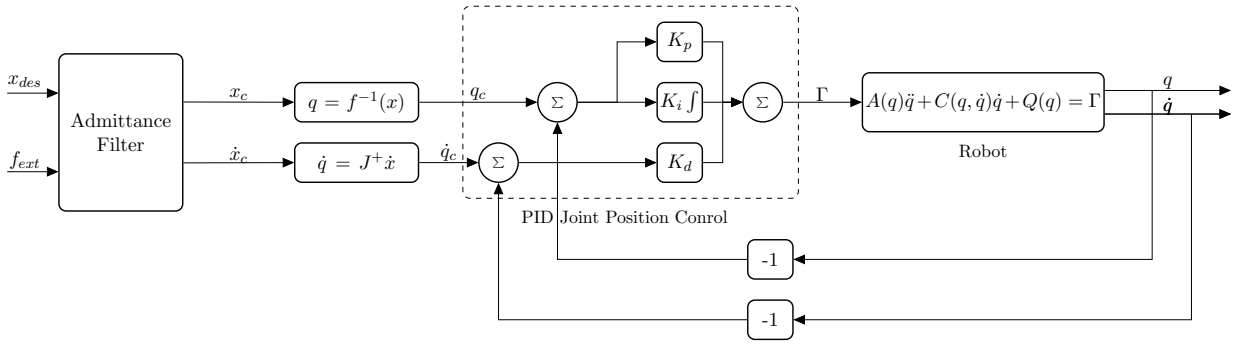

FIGURE 3.9: Robot Control: Block Diagram

- K_p, K_i and K_d are proportional, integral and derivative gains.
- J is the kinematic jacobian and J^+ is its peusdo inverse
- All other symbols have usual meaning.

As shown in the control block diagram (Figure 3.9), there is a inner PID Joint Position Controller and there is an admittance filter. The admittance filter takes the desired trajectory (x_{des}) and the external disturbance forces/torques (f_{ext}) as input and gives a complaint trajectory (x_c, $\dot{x}_c$) as output. The complaint trajectory in cartesian space is converted to a trajectory in the robot's joint space by the Inverse Geometric ($q = f^{-1}(x)$) and Inverse Kinematic ($\dot{q} = J^{+}\dot{x}$) models. These complaint joint angles and velocities are fed to the PID controller.

The equation of the PID controller is given in Equation 3.2.

$$\Gamma = K_p(q_c - q) + K_i \int (q_c - q) + K_d(\dot{q}_c - \dot{q}) \tag{3.2}$$

The admittance filter is necessary for the control of dynamic interaction between a manipulator and its environment, for eg. in problems related to object manipulation. The admittance filter reacts to interaction forces or distubances by imposing a deviation from the desired motion.

Using the notations discussed before, the equations related to the admittance filter is discussed next. Generic Admittance filter equation:

$$M(\ddot{x}_{des} - \ddot{x}_c) + D(\dot{x}_{des} - \dot{x}_c) + K(x_{des} - x_c) = -f_{ext}, \tag{3.3}$$

One can define

$$e = x_{des} - x_c, \tag{3.4}$$

and it's derivative which are describing the error between the desired trajectory and the compliant one fed to the PID controller.

The dynamics of the admittance filter can be written:

$$\ddot{x}_c = \ddot{x}_{des} + M^{-1}\left(D\dot{e} + Ke + f_{ext}\right). \tag{3.5}$$

M, D, K are the admittance parameters, respectively inertia, damping, and stiffness.

Special case of rotation gives the following expression for e and $\dot{e}$,

$$\begin{aligned} e &= \frac{1}{2}(R_{des}^T R_c - R_c^T R_{des})^{\mathrm{v}} \\ \dot{e} &= \omega_c - R_c R_{des}^t \omega_d \end{aligned} \tag{3.6}$$

where R_{des} and R_c represent the rotation, desired and compliant respectively, and $\bullet^{\mathrm{v}}$ the inverse-skew operator (extracting vector from skew-symmetric matrix) and ω_d and ω_c the rotational velocities, desired and compliant respectively. The formula for the error in rotational velocity is derived from [21].

This concludes the discussion about the Ground Manipulator, its physical parameters, software architecture and control methodology.

Chapter 4

The Aerial Manipulator

As mentioned in the Chapter 1, the aerial manipulator is used to aid the ground manipulator in manipulating long flexible objects which may vibrate and oscillate when grasped from one end. The aerial robot is used as a flying companion which helps to keep the level of joint torques exerted by a long manipulandum on the ground manipulator within safe limits.

4.1 Description of OTHex

For aerial manipulation we used a fully actuated aerial robot. Full actuation is achieved by having six tilted propellers. Flying platforms like quadrotors are underactuated. Although quadrotors can be used in aerial manipulation [16], the Tilted Hexarotor (Tilt-Hex) has the following advantages,

1. Since it is fully actuated:
   - It can track 6-DoF trajectories with the desired speed.
   - It can exert simultaneously forces and moments in any direction for physically interactive tasks.
   - It has no internal dynamics.
2. Since it has 6 propellers:
   - It has higher payload.
   - It is fault tolerant and can withstand physical damage to its propellers.

The Tilt-Hex is able to outperform the classical underactuated multi-rotors in terms of stability, accuracy and dexterity and represents one of the best choice at date for tasks requiring aerial physical interaction [22]. Proper design of propeller positions and orientations is required for attaining full actuation [23], [24], [25] and [26]. In this work, a derivative of Tilt-Hex is used and it is called the OTHex (Open Tilt-Hex). The OTHex has an open aperture on one of the sides in order to make room for the manipulandum. The Tilt-Hex cannot be used in cases where the manipulandum can come too close to the propellers, as the tilted propellers can strike against it, causing physical damage to both the flying platform and the manipulandum. The OTHex was designed to make the Tilt-Hex adaptable to aerial manipulation. The Figures 4.1 and 4.2 show the front and side views of the OTHex with the gripper arm mounted on it.

FIGURE 4.1: The OTHex with Arm, Front View

Parameter	Value	Remark
Extra Payload	2.5 Kg	
Weight	2.33 Kg	without battery
Tilt-Angles(α, β)	(35°, −10°)	determined by optimizing power loss

TABLE 4.1: Basic data related to OTHex

4.2 Modelling

The dynamic system modelling of the Open Tilt-Hex (OTHex) is presented here. The frames and the symbols are in reference to the Figure 4.3.

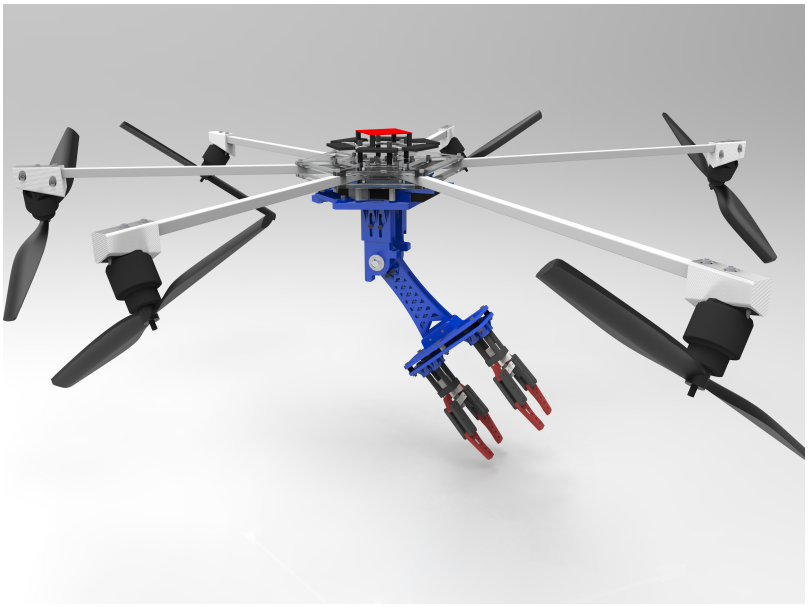

FIGURE 4.2: The OTHex with Arm, Side View

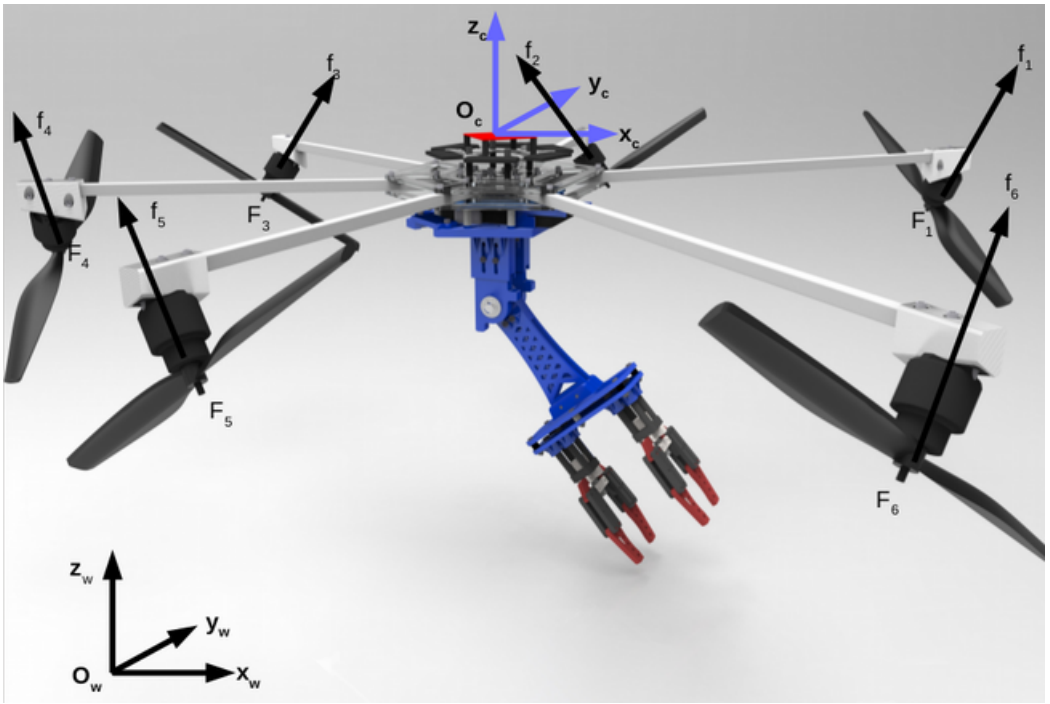

FIGURE 4.3: The OTHex with Frames, $[x_w, y_w, z_w]^T$: coordinates in world frame W, $[x_c, y_c, z_c]^T$: coordinates in OTHex's Center of Mass frame C, F_i: frame attached to each propeller, f_i: magnitude of thrust exerted by each propeller

Let us define the following Rotation matrices:

- R_C^W : Rotation matrix which transforms a point in the OTHex's Centre of Mass C frame to the world W frame.
- $R_{F_i}^C$: Rotation matrix which transforms a point in the i^{th} propeller frame F_i from to the OTHex's Centre of Mass C frame. This rotation matrix incorporates the tilt angles (α and β [1]

The i^{th} propeller rotates with angular velocity $\bar{\omega}_i$ about its axis of rotation. While rotating, the propeller exerts simultaneously a thrust force and a drag moment, In the center of mass frame, these quantities are respectively expressed as follows:

$$\begin{aligned} f_i^C(f_i, \alpha, \beta) &= f_i R_{F_i}^C(\alpha, \beta) e_3 \\ \tau_i^C(f_i, \alpha, \beta) &= (-1)^{i-1} c_f^\tau f_i R_{F_i}^C(\alpha, \beta) e_3 \end{aligned} \tag{4.1}$$

Here, $e_3 = [0,\ 0,\ 1]^T$ and $c_f^\tau > 0$ is a constant parameter which is characteristic of the type of propellers used, f_i is the magnitude of the thrust generated by each spinning propeller and it is related to the i^{th} propeller's spinning velocity $\bar{\omega}_i$ by the following quadratic relationship:

$$f_i = c_f \bar{\omega}_i^2 \tag{4.2}$$

Here c_f is another propeller dependent constant parameter. The presence of $(-1)^i$ in Equation 4.1 is from the fact that the adjacent propellers have an opposite sense of rotation in order to balance the net moment exerted on the platform. Since they spin in opposite directions, they exert drag force in opposite directions.

Summing all the thrust forces, we can compute the total force applied to the platform on its CoM, expressed in the world W coordinates:

$$f^W(\alpha, \beta, u) = R_C^W \sum_{i=1}^{n} f_i^C(f_i, \alpha, \beta) = R_C^W F_1(\alpha, \beta) u \tag{4.3}$$

Here, $u = [f_1, f_2, f_3, f_4, f_5, f_6]^T$ and $F_1(\alpha) \in \mathbb{R}^{3\times 6}$ is a α and β dependent matrix. If $\alpha = 0$ and $\beta = 0$ then it reduces to a normal planar hexarotor and $F_1 = [\mathbf{0}_{1\times 6}, \mathbf{0}_{1\times 6}, \mathbf{1}_{1\times 6}]^T$.

[1] $R_{F_i}^C = R_z R_x((-1)^{(i-1)}\alpha) R_y(\beta)$, R_z depends on the design of the Aerial Robot [23]

Summing all the moments (drag and thrust generated moment) acting on the platform, we obtain the following equation for the angular motion of the body, expressed in the CoM C-frame.

$$\tau^C(\alpha, u) = \sum_{i=1}^{n} (l_i \times f_i^C(f_i, \alpha, \beta) + \tau_i^C(f_i, \alpha, \beta)) = F_2(\alpha, \beta)u \tag{4.4}$$

Using the Newton-Euler approach, the equations of motion of the aerial platform can be then compactly written as:

$$\begin{bmatrix} m\ddot{p}_W \\ J\dot{\omega}_C \end{bmatrix} = - \begin{bmatrix} mge_3 \\ \omega_C \times J\omega_C \end{bmatrix} + \begin{bmatrix} f^W(\alpha, \beta, u) \\ \tau^C(\alpha, \beta, u) \end{bmatrix} \tag{4.5}$$

Here, J is the $\mathbb{R}^{3\times3}$ inertia matrix of the rigid body with respect the body CoM C, expressed in the frame C, m is the total mass of the platform and g is the magnitude of the acceleration due to gravity.

Using 4.3 and 4.4 in 4.5, we get:

$$\begin{bmatrix} m\ddot{p}_W \\ J\dot{\omega}_C \end{bmatrix} = - \begin{bmatrix} mge_3 \\ \omega_C \times J\omega_C \end{bmatrix} + \begin{bmatrix} R_C^W F_1(\alpha, \beta) \\ F_2(\alpha, \beta) \end{bmatrix} u \tag{4.6}$$

Defining,

$$F(\alpha, \beta) = \begin{bmatrix} R_C^W F_1(\alpha, \beta) \\ F_2(\alpha, \beta) \end{bmatrix} \tag{4.7}$$

If α is not equal to any singular configuration (like $0°$, 90°, etc. [23]) then F is full rank and it is invertible, under such circumstances the platform is fully actuated. In the OTHex we have a constant non zero α and β for all the propellers, this ensures that our platform is fully actuated.

4.3 Control

As discussed in the previous Section, the OTHex is a fully actuated flying platform, so it is possible to control all the 6-DoF independently, provided the actuation limits are not met. Let us consider the Equations 4.6 and 4.7, as α is not close to any singular value, the

matrix $F(\alpha, \beta)$ can be inverted easily and a dynamic system inversion can be performed to design a controller.

What follows next is a stepwise derivation of this dynamic feedback based controller:

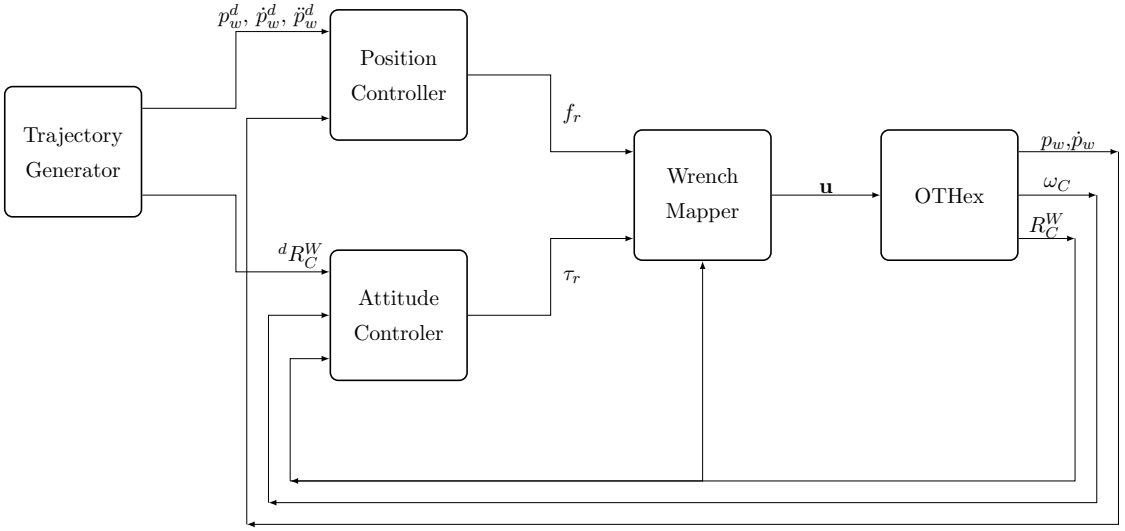

FIGURE 4.4: OTHex Control: Control block diagram of the Trajectory Generator, the Position and Attitude Controllers, the Wrench Mapper and the OTHex

1. **Position Controller:** Let us define a reference control force $f_r \in \mathbb{R}^{3\times 1}$, and compute:

$$f_r = mge_3 + m(\ddot{p}_w^d - K_p(p_w - p_w^d) - K_d(\dot{p}_w - \dot{p}_w^d)) \tag{4.8}$$

   Here, p_w^d, $\dot{p}_w^d$ and $\ddot{p}_w^d$ are the desired linear position, velocity and acceleration respectively in the world frame. $K_p > 0$ and $K_d > 0$ are the tunable proportional and derivative gain matrices.

   Because of actuator constraints, the magnitude of thrust f_i generated by every single propeller is naturally bounded by physical limits and so the f_r generated in the Equation 4.8 may not be feasible for the current orientation of the platform, under such circumstances the orientation needs to be modified by generating a compliant $f_r = f_c$. More information about this can be found in [27] but in this report the simplified case is concerned so $f_r = f_c$.

2. **Attitude Controller:** The attitude controller takes as input the desired orientation, denoted by ${}^dR_C^W \in SO(3)$ and the measured attitude state R_C^W, ω_C to compute the

reference control torque $\tau_r \in \mathbb{R}^{3\times 1}$ as follows:

$$\tau_r = \omega_C \times J\omega_C - K_R e_R - K_\omega \omega_C \tag{4.9}$$

e_R is the orientation error and can be determined by Equation 3.6 given in [21].

3. **Wrench Mapper:** The Wrench Mapper takes as input f_r and τ_r and computes the control input u as:

$$u = \begin{bmatrix} R_C^W F_1(\alpha, \beta) \\ F_2(\alpha, \beta) \end{bmatrix}^{-1} \begin{bmatrix} f_r \\ \tau_r \end{bmatrix} \tag{4.10}$$

This control u inverts the system and the control solution becomes similar to Proportional Derivative control.

The overall control block diagram is shown in Figure 4.4.

4.4 Gripper

Since the OTHex is fully actuated, we did not need a complicated multi DoF arm for simple pick and place operation. For the Tele-MAGMaS project the primary task from the aerial manipulator was to lift a bar from one end with the ground manipulator cooperatively lifting it from the other. So, we designed a gripper which opens or closes depending on whether it is at the grasping/releasing point or not and which offers some sort of compliance to rotation of the bar in the vertical plane. Hence, the gripper we designed had just one DoF about the OTHex's $y-$axis and this revolute joint was passive. The end effector was controlled by a arduino board which received command from the Control PC from it's corresponding GenoM component. The GenoM component received the control inputs from the MATLAB/Simulink code in which the high level control algorithm was running. More information about the generic software architecture can be found in the Chapter 6. The Gripper, mounted on the OTHex is shown in Figures 4.1 and 4.2.

This concludes our discussion about the Aerial Manipulator, its mathematical modeling, control and the gripper.

Chapter 5

The Haptic Device

Haptic devices are generally used to provide a sense of touch and proprioception. Haptics is the science of creating a realistic sense of touch to the user in a virtual environment. The word haptic is derived from the Greek word haptikos which means pertaining to the sense of touch. Haptic feedback is common way for acquiring spatial knowledge of the environment and object properties.

As robotic applications are gradually shifting from repetitive factory labor to everyday environments, robots must have more perception and action capabilities that makes their interaction with humans safer. Haptic devices can be used to implement the idea of virtual fencing in which the operator defines a boundary around the robot which it should not cross. When the haptic device is used, it will give a strong force feedback or a resistance to motion when the robot is near the virtual fence. It can also be used for assistiv guidance in order to guide the robot end effector (EE) to a particular point by creating a virtual cone around the goal point and when one tries to force the EE out of this cone the force feedback by the haptic device would give the notion that the EE is trying to go away from its goal.

In the project, Tele-MAGMaS, the term "Tele" refers to Tele-operation or the presence of human in loop. The roll of human is supervisory, as the presence of human in the loop ensures that the reaction to any unforeseen event is faster. We used the omega.6 haptic device by Force Dimension for tele-operation. In the next sections the functionality of omega.6 are presented.

5.1 Description of omega.6 Haptic Device

The omega.6 Haptic Device is shown in Figure 5.1. It is a pen-shaped force feedback device and provides perfect decoupling of translations and rotations.

FIGURE 5.1: The omega.6 Haptic Device

It can be used in a variety of application like:

1. Tele-Surgery, Tele-Medicine, Medical Robotics, etc.
2. Space Robotics.
3. Teleoperation consoles.
4. Virtual Simulations.
5. Physical interaction with Robots.

This device has a delta-based parallel kinematics and the rotations are decoupled from translations. Besides these there is active gravity compensation.

It supports commonly used operating systems, hence it is easy to setup and use. The software is available in haptic Software Developement Kit (SDK).

5.2 Application in Tele-MAGMaS

Though we have explored the use of this device to give us situational awareness, we used this device primarily as an input device. It was used to fly the OTHex. Thanks to the

velocity monitoring and electromagnetic damping of the omega.6 one cannot bring about sudden change in the input, and this acts as a safety feature in tasks like cooperative aerial ground manipulation.

Besides, we also intially used it to control the motion of the ground manipulator. It was used to control the velocity of the joints and to give local corrections to the position of the end effector in real time. The local corrections were necessary because the trajectory was planned off-line and could not take into account any unforeseen obstacle in the way of the end effector.

The device was connected to our control PC by a USB cable and it took commands from a s-function implemented in simulink. The s-function was written in C using the haptic SDK.

This device has a lot of applications in the Tele-MAGMaS project and integration of the force feedback capabilities is also in the pipeline. Use of force feedback for cooperative aerial manipulation has been already demonstrated at LAAS-CNRS (Figure 5.2 , [2]).

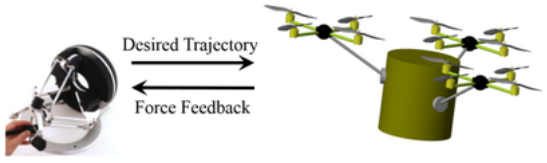

FIGURE 5.2: Bilateral Teleoperation for cooperative manipulation [2]

This concludes the discussion about the Haptic interface, its use in the Tele-MAGMaS project and the potential future uses.

Chapter 6

Software Architecture

Since, the project involves several subsystems, the software integration is an arduous task. The high-level control algorithms were written in MATLAB/Simulink environment and they run on a work station/PC. Using the middleware independent component generator GenoM (Generator of Modules) developed at LAAS-CNRS, we were able to communicate with various systems and subsystems employed in the Tele-MAGMaS project. The various constituents of the overall software architecture is described in the following sections and finally a global picture is presented where the interconnection between each of the constituents is elucidated.

6.1 Middleware

Before delving deeper into the software architecture, it is important to have a succinct idea about robotic middleware.

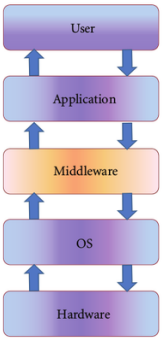

FIGURE 6.1: Middleware Layers

As illustrated in Figure 6.1 given in [28], a robotic middleware is an abstraction layer between the computer's operating system and the robotic components or the applications [28]. A developer just needs to implement the algorithm or the logic as a component (application) which can be integrated with other components. A robotic middleware offers Software Modularity, Hardware Abstraction, Architecture Abstraction, Platform Independence, Portability and communication between processes (components).

6.2 GenoM

Following the tenets of Component Based Robotic Engineering ([29], [30]), GenoM is a robotic component generator developed by the robotics team of the LAAS-CNRS. GenoM is one of the software packages developed under the Openrobots[1] project at LAAS-CNRS. GenoM3 (the current version of GenoM) is middleware independent, hence, it supports different middlewares through the notion of templates. A template is a program that describes how the formal description of the component is going to be translated into actual code. Templates also make it possible to support different programming languages. The examples of existing templates are genom3-openprs, genom3-orocos, genom3-pocolibs, genom3-ros, etc. which generate openPRS, Orocos, pocolibs, ROS based components respectively. One must note that openPRS, Orocos, pocolibs, ROS, etc. are some of the widely used robotic middlewares. For instance if one chooses a genom3-ros template then, the genom component build, will result in a ROS node and it will communicate with other nodes (components) using rostopics, rosservices and rosactions and this is what is done in the Tele-MAGMaS project. A GenoM component is a server that provides a number of services to its clients and communicates with other components through data ports. It is worth mentioning that there exists a HTTP server called genomix which is a generic interface between clients and genom components. There is a MATLAB package called matlab-genomix that interacts with the genomix HTTP server and can control GenoM3 components. It can also be used with the rosix server to control ROS nodes. Besides this, the rosix HTTP server is a generic interface between clients and ROS nodes. One can use rosix to get the data published by a rosnode in the MATLAB/Simulink environment. The GenoM3 components can also be controlled using tcl shell with the help of tcl-genomix package which interacts with genomix HTTP server and control the GenoM3 components. As already mentioned in the beginning of this chapter, all of our high level control algorithms are written in MATLAB/Simulink. So, we use the matlab-genomix package to

[1] `https://git.openrobots.org/`

interact with genomix server which provides an interface to the genom components. In our case the client is created in MATLAB environment. The scenario described in this paragraph is illustrated in Figure 6.2. The GenoM Component communicates with the hardware through device drivers using USB or Ethernet.

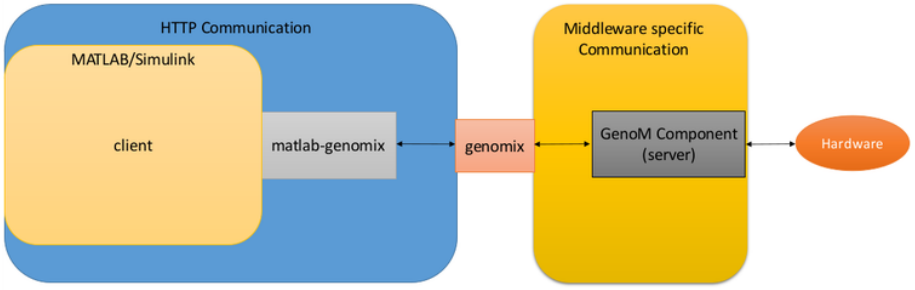

FIGURE 6.2: Client(Created in MATLAB/Simulink) and Server (GenoM Component) communicating using the genomix interface with the help of matlab-genomix package

6.3 ROS

ROS (Robot Operating System) is a flexible software for writing robotic software. Just like GenoM, it can be used to generate robotic components which implement some functionality. In the ROS framework, the components are called nodes.

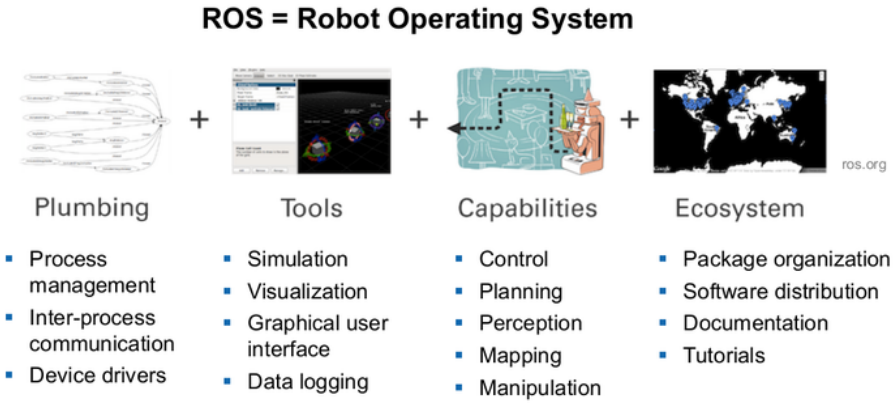

FIGURE 6.3: ROS Overview

The basic utilities provided by ROS is well summarized in the Figure 6.3. The main philosophy behind ROS is enumerated as follows:

1. Peer to peer : Individual programs communicate over defined API (ROS messages, services, etc.).

2. Distributed : Programs can be run on multiple computers and communicate over the network.
3. Multi-lingual: ROS modules can be written in any language for which a client library exists (C++, Python, MATLAB, Java, etc.).
4. Light-weight: Stand-alone libraries are wrapped around with a thin ROS layer.
5. Free and open-source: Most ROS software is open-source and free to use.

At the lowest level, ROS offers a message passing interface that provides inter-process communication and is commonly referred to as a middleware. This is of paramount importance to us because we use ROS as the middleware in the Tele-MAGMaS project. As mentioned in Section 6.2, thanks to the middleware independence of GenoM3, we use ROS as our middleware by choosing the genom3-ros template to build a GenoM component and doing so, one can launch that particular component as a ROS node which can communicate with other nodes using ROS topics, services, action library, etc.

The ROS middleware provides these facilities:

1. publish/subscribe anonymous message passing (rostopic)
2. request/response remote procedure calls (rosservice, rosaction)
3. recording and playback of messages (rosbag)
4. distributed parameter system (Parameter Server)

Since ROS is very user friendly and has a large number of users we prefer it over other middlewares. The demerit of ROS is that it is not real time.

The best aspect of ROS is the availability of command line tools to inspect the running processes. The use of several command line commands like rostopic list/echo/hz/info, rosnode list/info etc. makes debugging very easy. One can also record data using rosbag and visualize data using rviz. All these properties makes ROS a very powerful platform and the presence of lot of online help makes troubleshooting easier. This is the reason why ROS has a global acceptance in the robotic community. The ultimate goal of any project is to make it as user friendly as possible in all the aspects. So we chose ROS over other middlewares in the Tele-MAGMaS project.

6.4 Software and Hardware Integration

In this section the integration of various software components among each other and with the hardware compoments is described. We can get an overview of the communication between main hardware components with the computers and the control PC from Figure 6.4. The hardware components are shown in ellipses and the computers are shown in rectangular boxes. Some harware components like the manipulator (iiwa), the motion capture cameras (mocap cameras) and the Vision Sensor have their own dedicated computers which are shown in smaller rectangles and the Control PC is shown in the big rectangle.

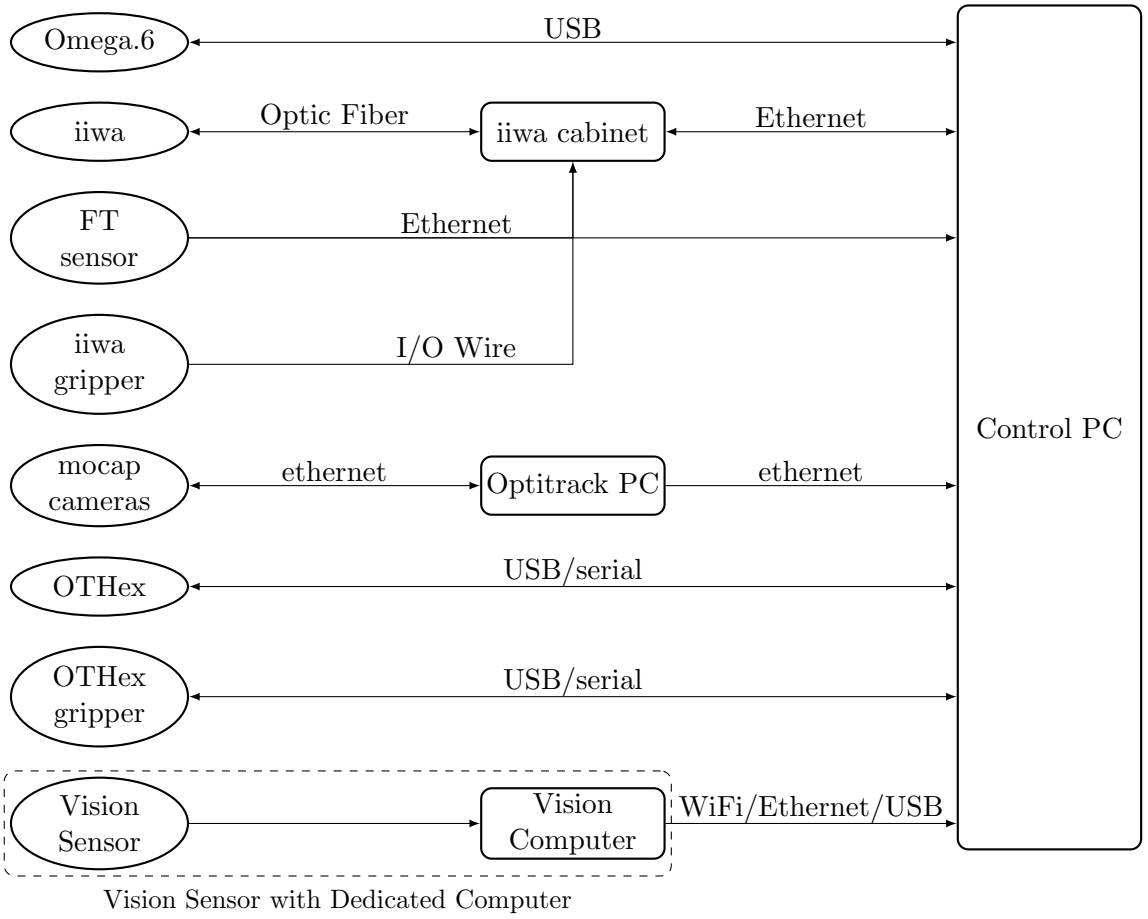

FIGURE 6.4: Main communication links between Hardware Components (Ellipses) and the Computers (Rectangles)in the overall system [2]

The mode of communication, Ethernet, USB, Wifi, etc., between the components is also clearly elucidated. In practice, the vision sensor and the dedicated computer are generally

embedded as one physical entity, providing an integrated product from the manufacturers' point of view.

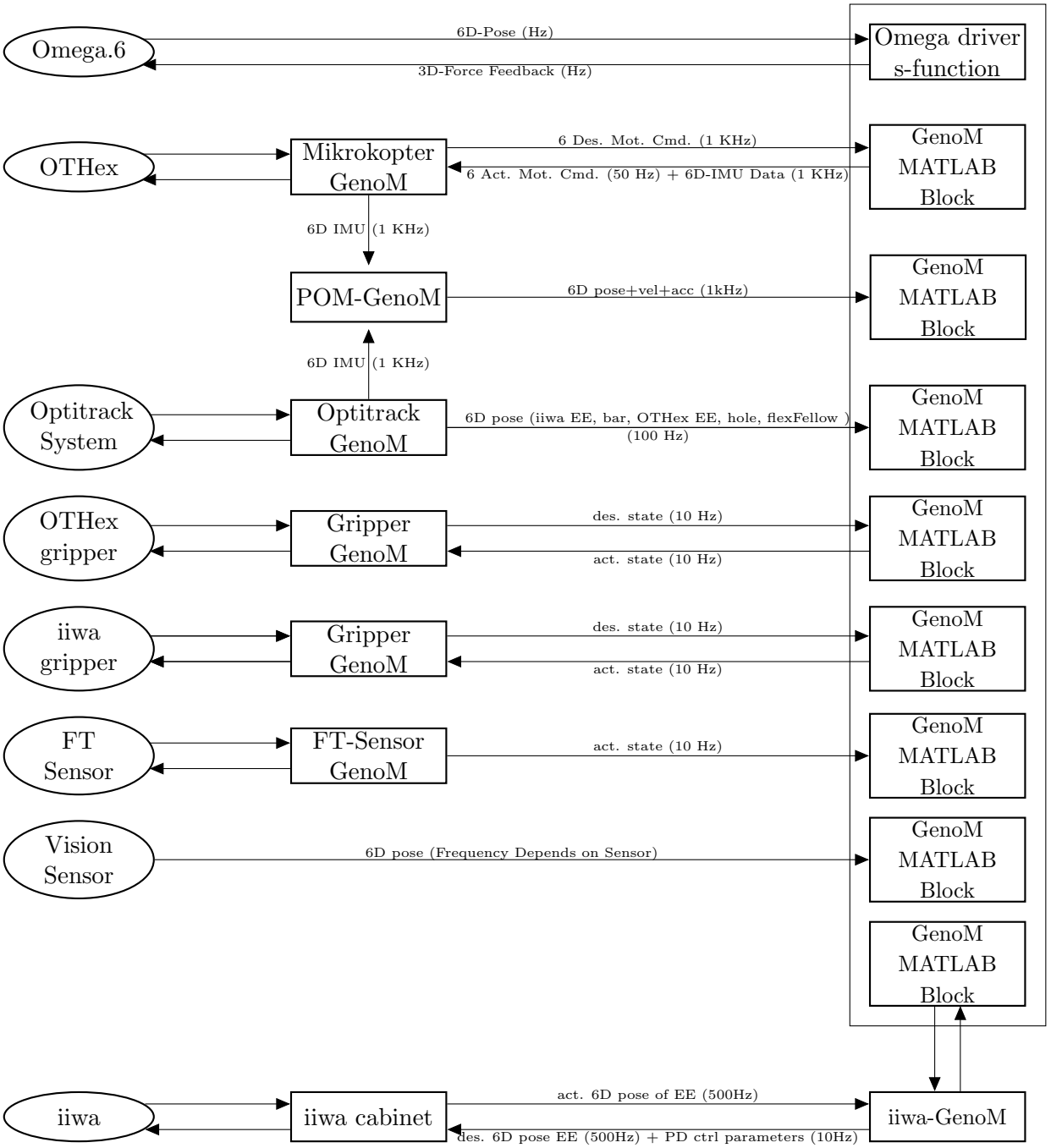

FIGURE 6.5: Flow of information from Hardware to MATLAB in the Tele-MAGMaS architecture

The flow of information from Hardware to MATLAB via the corresponding GenoM components is shown in Figure 6.5. One can refer to Figure 6.2 to get a detailed overview of the

communication process. For the sake of explanation, let's consider just the OTHex and try to understand how it sends/receives data to/from the controller in MATLAB/Simulink. The OTHex is the Aerial Robot that we employ in our project and it's flight control and motor control boards are supplied by Mikrokopter. In the present context, OTHex refers to the Mikrokopter Flight and Motor Control boards. The GenoM component Mikrokopter-GenoM is the first software element which talks to OTHex with the help of Mikrokopter device driver. Since ROS is our chosen middleware, the Mikrokopter-GenoM is a software component built with the genom3-ros template and therefore it runs as a ROS node and communicates with other components (nodes) through rostopics, rosservices, rosactions, etc.. The genomix server is used as a bridge between the ROS middleware and the matlab-genomix package (running on MATLAB), this particular package helps to and fro communication between Mikrokopter-GenoM (or any GenoM component) and MATLAB/Simulink environment where all of the high level control algorithms are written/running.

There are some hardware elements like the Omega.6 haptic device and the vision sensor which do not have corresponding GenoM component. For the Omega.6 haptic device, the communication with MATLAB/Simulink is done using a s-function and the physical connection is through USB cable. The vision sensor publishes the 6D pose as a rostopic and we can use the rosix server (instead of genomix server) which facilitates to and from communication between MATLAB/Simulink and the node publishing the necessary topic.

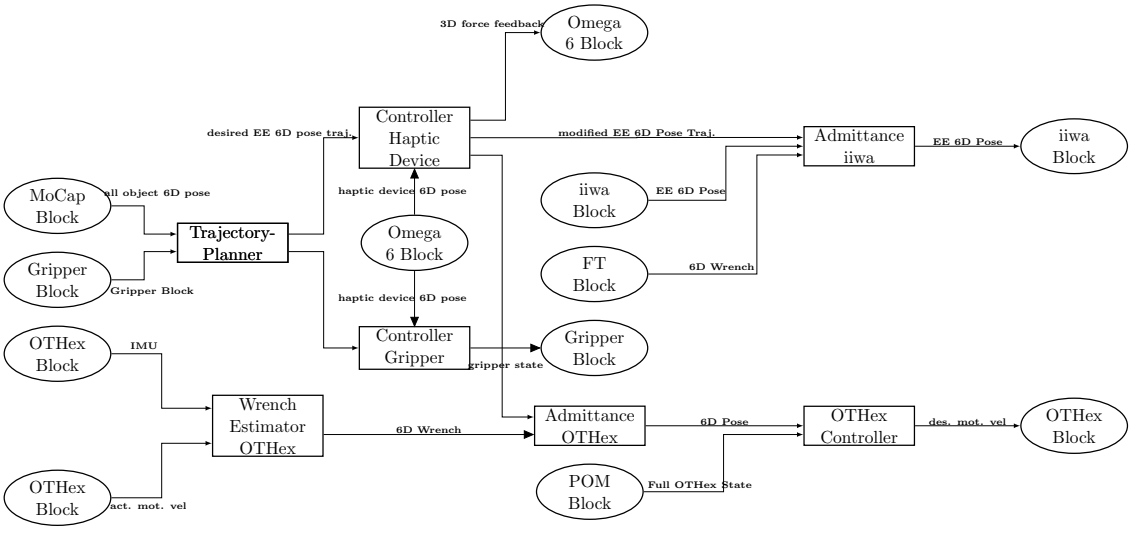

FIGURE 6.6: GenoM Components and Matlab/Simulink Blocks

The block diagram of GenoM Components(ellipses) and the algorithms written in MATLAB/Simulink (rectangles) is illustrated in Figure 6.6. The control algorithms run on

the MATLAB/Simulink blocks and the GenoM Components are sources or sink of data for the MATLAB/Simulink blocks. What follows is a brief description of each of the MATLAB/Simulink blocks.

1. **Trajectory Planner**: It takes as input the 6D pose of all the objects in the scenario from the MoCap's GenoM component and also the status of the Gripper (open/close) from the Gripper's GenoM component and gives as output the desired trajectory of the end effector (EE) which is fed to the controller of the Omega6 Haptic device. It also gives as output the action that the gripper must undertake. The desired EE pose also detemines the desired position of the OTHex.

2. **Controller Haptic Device**: The controller of the Haptic device takes as input the desired end effector (EE) trajectory and also takes input from the Haptic device (Omega6 Block). Based on the inputs from the Haptic device, it makes certain changes on the desired trajectory of both the EE and the OTHex. This modified trajectories are sent to the respective Admittance filters. It also gives a 3D force feedback as output which is fed to the Omega6 GenoM component. Although a framework for this component was developed, it was not implemented in the final system

3. **Controller Gripper:** The Gripper Controller gives the gripper state as output which is fed to the gripper GenoM component. The control decides the gripper state by taking inputs from the Trajectory Planner and the Haptic Device.

4. **Admittance iiwa:** The admittance filter gives the compliant EE trajectory as output, which is fed to the GenoM component for the KUKA iiwa LBR 14 R820. The compliant trajectory is determined by taking as input, the external wrench from the Force Torque Sensor GenoM component, the actual 6D pose of the EE from the KUKA iiwa LBR 14 R820 GenoM component and the modified trajectory generated from the controller of the haptic device.

5. **Wrench Estimator:** The Wrench Estimator takes IMU input and the actual motor velocities as input from the OTHex GenoM component and estimates the 6D Wrench on the flying platform which is the output of the Wrench Estimator block.

6. **Admittance OTHex:** This block implements the admittance filter for the OTHex. It takes as input the 6D Wrench generator by the Wrench Estimator and the modified Gripper trajectory generated by the Haptic device controller. It gives as output a compliant trajectory.

7. **OTHex Controller:** The OTHex controller takes the compliant 6D pose of the Gripper from the admittance filter designed for the OTHex and the full state of the OTHex as input from the POM GenoM component (which implements an Unscented Kalman Filter) and gives the desired motor speed as output which is fed to the OTHex GenoM component.

This concludes the discussion of the overall software architecture and it's integration with the hardware.

Chapter 7

Tele-MAGMaS Results

In this Chapter, the results of cooperatively manipulating a bar with the help of the KUKA LBR iiwa 14 R820 and the OTHex is discussed. The plots presented will show that the ground manipulator and the aerial manipulators precisely track the desired position despite the external wrench exerted by the long bar. The results from the ground manipulator are presented in Section 7.1 and the results from the OTHex is presented in Section7.2.

7.1 Results from KUKA LBR iiwa 14 R820

The desired joint positions, the actual joint position and the error between the desired and the actual are shown in Figures 7.1, 7.2 and 7.3 respectively. The legends A1 to A7 refer to the seven joints on the KUKA LBR iiwa 14 R820.

A visual inspection of Figure 7.1 and 7.2 confirms that the position tracking is accurate but this fact is verified by looking at the error plot in Figure 7.3. The error is negligibly small thus ensuring that the desired trajectory is closely tracked.

The torques measured at each joint are show in Figure 7.4. The fact that the joint torques are within the respective joint limits is well illustrate in the Figure 7.5. [1] The maximum magnitude of torque exerted on joints from A1 to A7 are 5.37 Nm, 136.12 Nm, 15.2 Nm, 62.46 Nm, 7.07 Nm, 27.45 Nm and 3.35 Nm respectively and they are well below the safe threshold. Had the OTHex not been there in the opposite side, the ground manipulator would have operated in the verge of its safe torque envelope to manipulate the long and heavy bar.

[1] Refer to Table 3.2 for joint torque limits

FIGURE 7.1: Desired Joint Positions Vs. Time

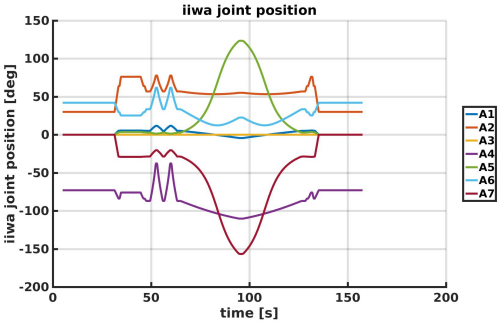

FIGURE 7.2: Actual Joint Positions Vs. Time

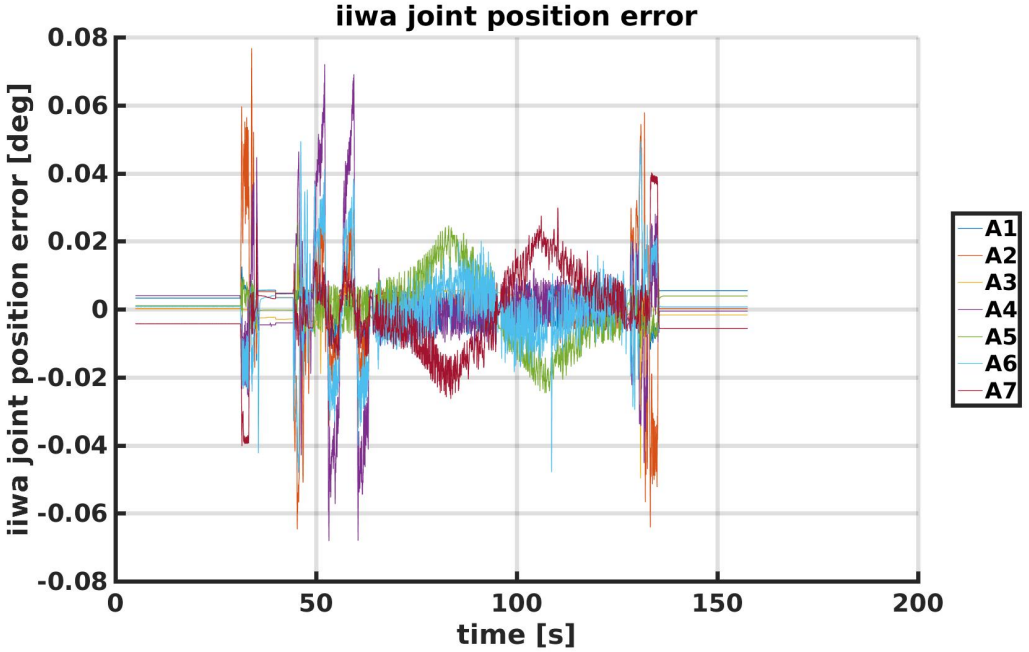

FIGURE 7.3: Joint Position Error Vs. Time

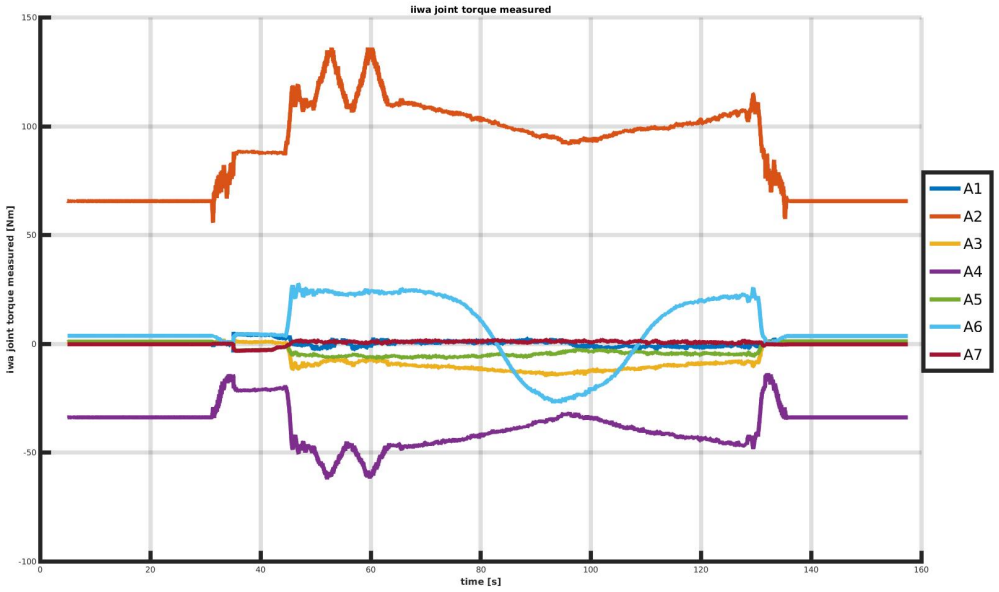

FIGURE 7.4: Measured Joint Torque Vs. Time

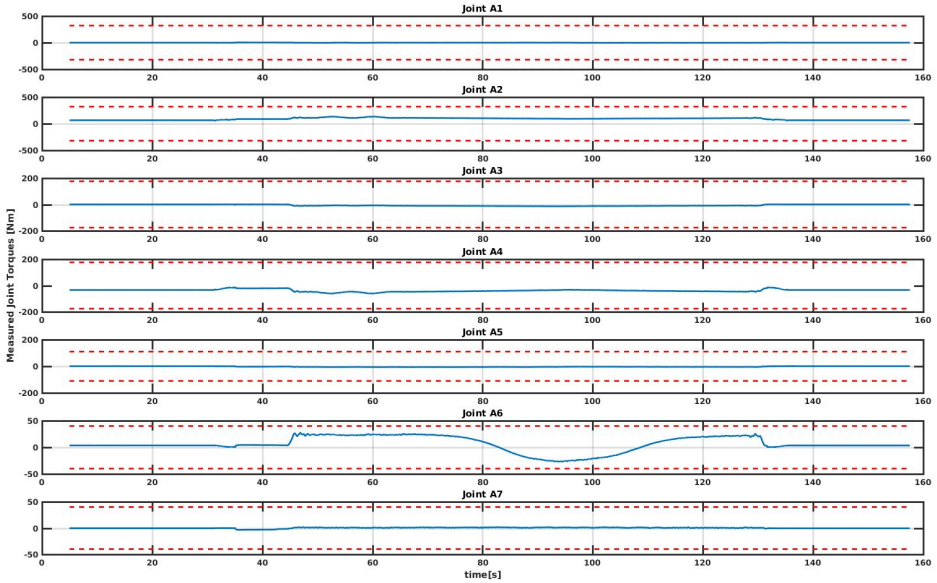

FIGURE 7.5: Measured Joint Torque(Blue) Vs. Time with Joint Torque Limits(Red)

7.2 Results from OTHex

In the current section the accuracy of pose tracking for the OTHex is shown. The accuracy of the position and orientation tracking can be seen by the negligible errors in the Figure 7.8 and 7.11 for the postion tacking and orientation tracking errors respectively. The controller designed for OTHex is able to track the desired trajectory in 6D with high precision, with an average position error of 0.0111 m, 0.0202 m and 0.0087 m along x, y and z respectively and an average orientation error of 0.5695°, 0.6088 ° and 1.0497 ° in roll, pitch and yaw respectively.

The approximate trajectory of the OTHex can be guessed from Figure 7.7. The OTHex first goes to a grasping point, grasps one end of the bar and then moves up and finally comes down and goes back to the spot where it started. The to and from motion from the resting point to the grasping point is manually controlled but the motion during cooperative manipulation is autonomous. In fact, during the cooperative manipulation phase, the OTHex tries to track a trajectory that is a function of the pose of the end effector of the ground manipulator grasping one end of the bar and the geometry of the bar (the length to be specific). In the ground robot's end effector frame, the OTHex's gripper must track a pose which has the same orientation as the ground manipulator's end effector, but displaced along one of the principal axes by an amount which equals the length of the bar.

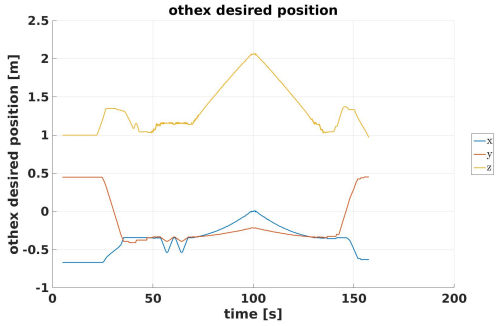

FIGURE 7.6: Desired OTHex Position Vs. Time

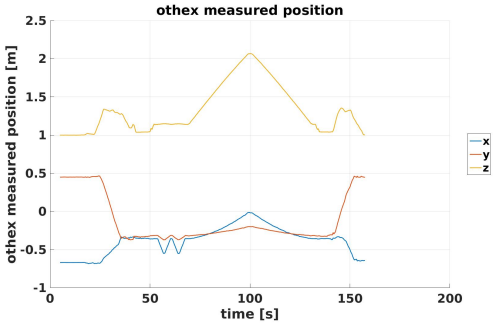

FIGURE 7.7: Actual OTHex Position Vs. Time

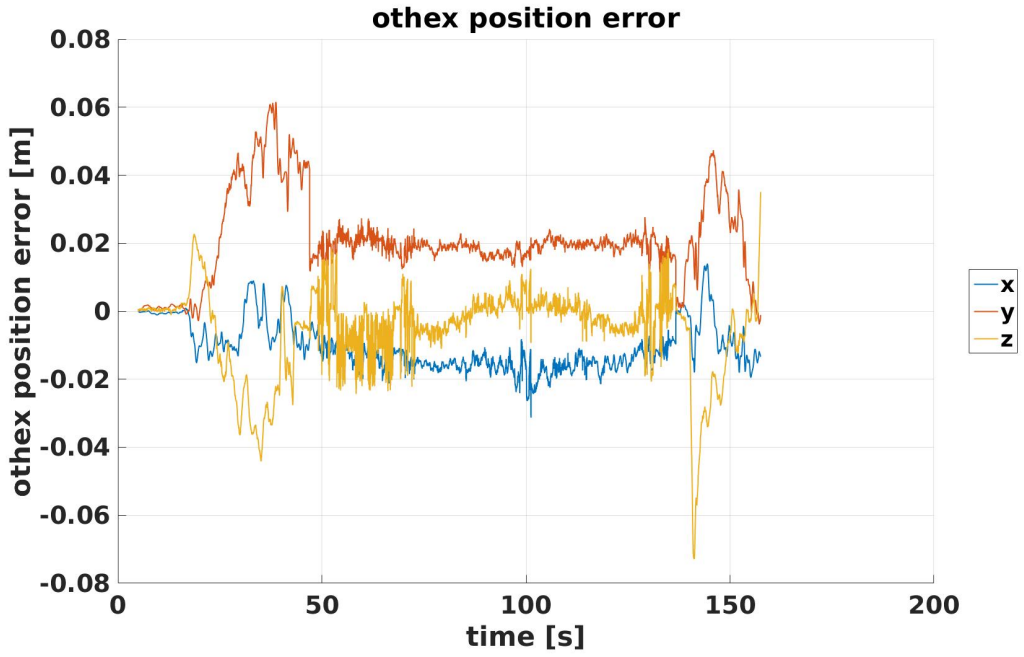

FIGURE 7.8: OTHex Position Error Vs. Time

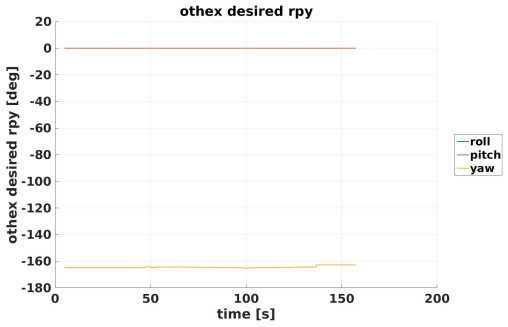

FIGURE 7.9: Desired OTHex Orientation Vs. Time

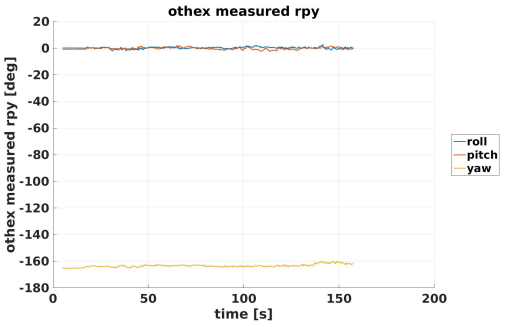

FIGURE 7.10: Actual OTHex Orientation Vs. Time

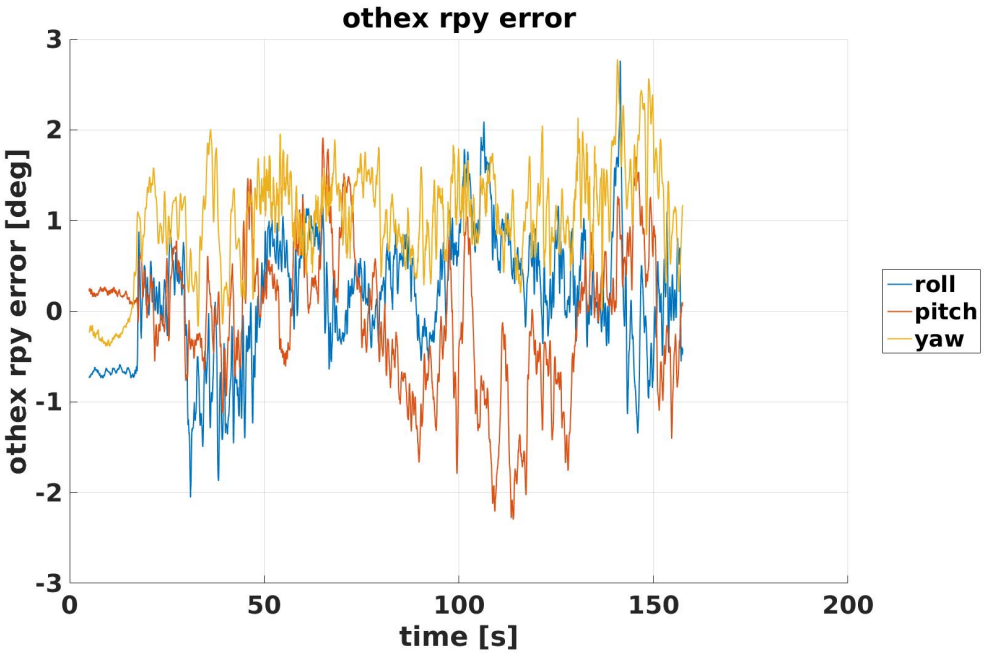

FIGURE 7.11: OTHex Orientation Error Vs. Time

7.3 Visualization

The Tele-MAGMaS concept is shown in the Figure 7.12 and all the subsystems are labelled.

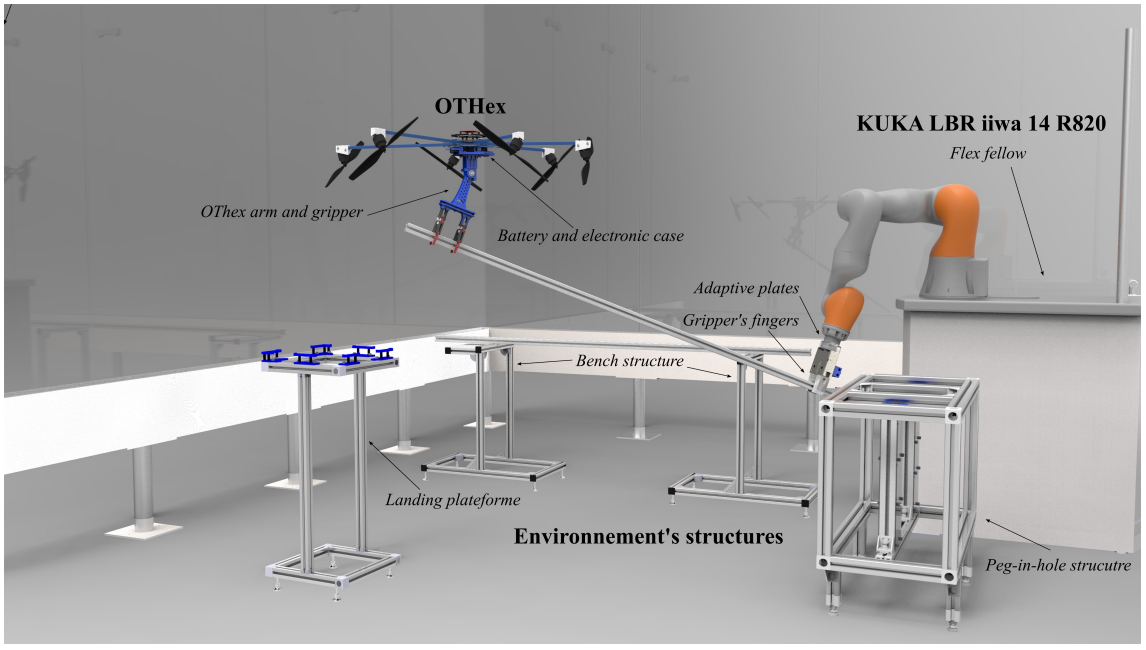

FIGURE 7.12: The Tele-MAGMaS Concept

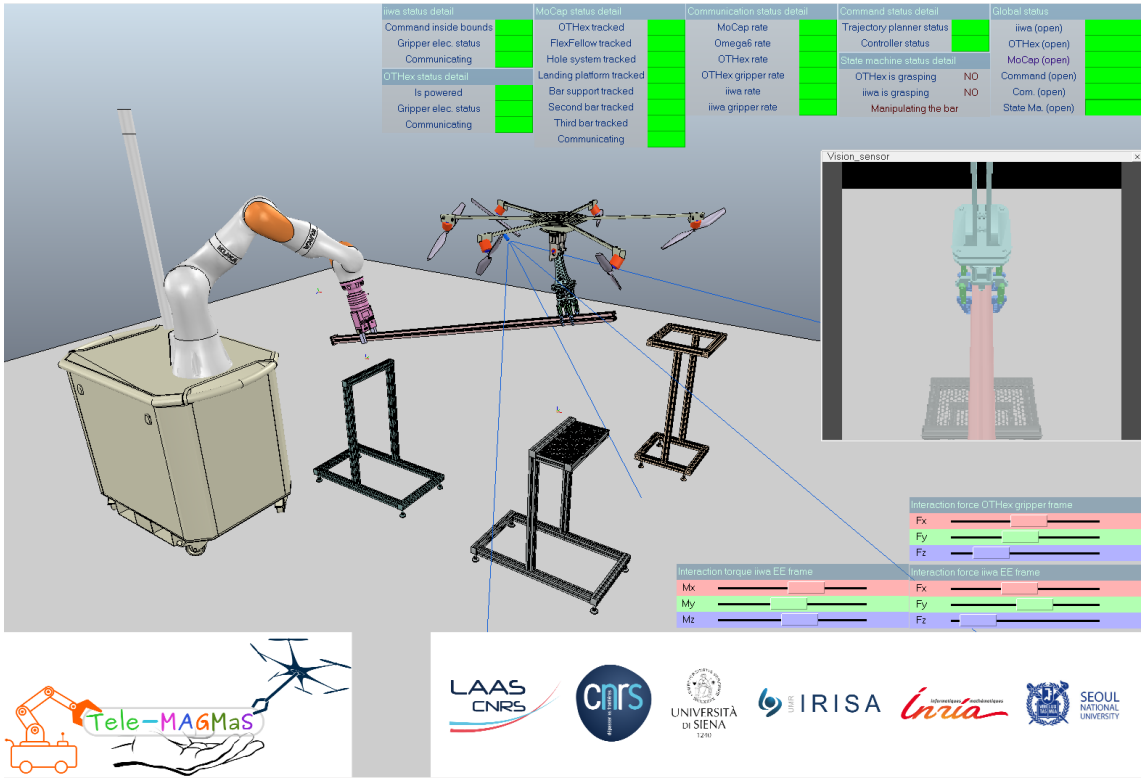

FIGURE 7.13: The Visualizer in V-Rep

For simulation and visualization an interactive simulator was designed in V-Rep (Figure 7.13). The simulator was used to emulate many trials before implementing it on the real system and it communicated with the high level control commands coming from MATLAB/Simulink by means of respective s-functions. As shown in Figure 7.13, the visualizer also gave the status of many parameters concerning each of the major subsystems in an interactive way. For example, looking at the visualizer one can check if all the subsystems are working fine or not.

This concludes the discussion about the results from the Tele-MAGMaS project. As already discussed, the KUKA iiwa LBR 14 R820 and the OTHex are able to track the desired joint position and pose respectively while manipulating the bar cooperatively with the joint limits of the ground robot within the safety threshold.

Chapter 8

MoCap denied State Estimation

State estimation is the problem obtaining reliable estimates of postion and velocity. In the Tele-MAGMaS project, the motion capture system was used in conjuction with the IMU onboard the OTHex to estimate the linear and angular position, velocities and accelerations.

The major disadvantages of using only IMUs for localization and navigation is that it suffers from accumulated error because the guidance system incrementally integrates the acceleration to calculate the velocity and position, hence, measurement errors accumulate over time. This leads to an ever increasing drift. A constant error in acceleration results in linear error in velocity and a quadratic error in position. Therefore, it is necessary to use other sources of data for external corrections. The motion capture (MoCap) solves this purpose. An Unscented Kalman Filter (UKF) (implemented as a GenoM compomenet called pom[1]) takes inputs from the IMU and the MoCap and generates a fused state which is more accurate than both the sources.

Although the MoCap works seamlessly indoors, it cannot be used outdoors. The main aim of Tele-MAGMaS is to use a heterogeneous team of robots for disaster response, decommissioning of nuclear plants, moving parts in factories, etc. and environments like these do not generally provide the luxury of installing the MoCap systems. Besides, it is also important to note that the OTHex is not a wireless platform and all the data between the OTHex and the ground PC is physically transmitted/received by means of USB/Serial cable. The future plan is to run all the control algorithms on board the OTHex instead of running them on the ground PC and also make the system wireless. As the MoCap

[1]This component collects position/velocity/acceleration measurements from other components, and generates a fused state estimation from these sources.

data consumes a lot of bandwidth, the WiFi infrastructure does not allow to seamlessly transmit high frequency motion capture data without loss of packets. Therefore, it is not advisable to use WiFi when transmitting MoCap data as it may result in loss of data packs which may deteriorate the performance of the controllers running onboard. In fact it highly depends of the available network because many labs around the world are already using the WiFi to transmit the MoCap data and the network infrastructure of LAAS-CNRS is being revamped to meet this challenge but still there are some problems when it comes to data transmission using WiFi.

Summarizing the demerits of MoCap:

1. It cannot be used outdoors.
2. The MoCap data transmission rate highly depends on the network configuration and data packet may be lost when using WiFi.

8.1 Alternatives to Motion Capture

There are quite a number of alternatives to the MoCap system which can be used onboard a robot to estimate its pose relative to some reference. Some of the popular alternatives are GPS (Global Positioning System), LiDAR, Time of Flight Camera, Monocular Camera, Stereo Camera, RGB-D Camera, etc.. One or a combination of these sensors can be used along with the onboard IMU to estimate the state of a robot. But aerial robots have a constraint on the payload and power supply and more so when the primary application of the aerial robot is to do aerial manipulation. So, the sensor used must have minimal weight and power consumption. In general cameras are the most suitable sensors for medium sized aerial vehicles. Cameras have extremely low Size, Weight, and Power (SWaP) footprint. What follows next is a brief description about all the kinds of sensors listed above.

1. **GPS:** GPS receiver is one of the most commonly used sensor on many mobile robots including aerial robots operating outdoors, but the caveat is that it is only available outdoors and reliable in open spaces. It is not reliable near tall buildings, walls, in places where there is too much occlusion by natural and man made structures, etc. GPS is affected by multipath reflections from tall buildings, by reflections and masking in trees, and by ionospheric delay. While it is suitable for aerial robots flying high in open skies, it is not suitable for aerial robots meant to work as aerial manipulators in constrained environments.

2. **LiDAR:** LiDAR is an effective sensor for estimating the state of a robot in GPS denied environments but it is heavy and not suitable for an aerial robot meant for aerial manipulation. Besides, it consumes more power when compared with other suitable sensors. Moreover it is very expensive. Planar LiDAR are cheaper but they provide data only in one plane and need to be rotated about an axis (by using an actuator) to give 3D information and this makes it complicated for our use.

3. **Time of Flight(ToF) Camera:** Time of flight cameras can be used outdoors but they have limited resolution. Unlike laser scanning systems (like LiDAR), ToF cameras illuminate an entire scene. Due to multiple reflections, the light may reach the objects along several paths, consequently care should be taken with multiple path reflections.

4. **Monocular Camera:** A monocular camera is one of the most inexpensive camera available and there are a number of algorithms for doing visual odometry and visual slam using monocular cameras but as the depth is not observable from just one camera, the scale of the map and estimated trajectory is unknown. There are many packages which use downward facing cameras to estimate the pose but the surface must be textured for those packages to work and it cannot be assured in all the circumstances. In addition the system bootstrapping requires multi-view or filtering techniques to produce an initial map as it cannot be triangulated from the very first frame. Most importantly, monocular SLAM suffers from scale drift and may fail if performing pure rotations in exploration [19].

5. **Stereo Camera:** A stereo camera is a simple setup and can be assembled by using two monocular cameras. Unlike monocular cameras, stereo setups also give the depth information and do not require multiview for system bootstraping. The baseline, i.e. the distance between the horizontal geometric centers of projection of the two cameras is an important parameter for depth measurement. For points which are relatively farther as compared to the baseline (40-50 times), the problem reduces to monocular case. So, for stereo vision based odometry/SLAM to work, there should be enough texture in the nearby[2] scene.

6. **RGB-D Camera:** In stereo camera, the left and the right images are used to measure the depth but with RGB-D Cameras, the depth is obtained by using an IR projector and a receiver and the features from the scene are extracted from the RGB image captured by the RGB camera. RGB-D cameras are very popular and

[2]the concept of near and far depends on the baseline, if the baseline is large then it can track farther and vice versa

work well indoors. They are light weight and consume less power in comparison to LiDARs and the likes, but the only demerit is that it cannot be used outdoors because the IR pattern projected by the IR projector is washed by sunlight.

7. **Marker based approaches:** Marker based approaches are very common and they use only a monocular camera and visual markers in the scene. The visual marker's location are known in the world frame and this helps to solve the depth ambiguity that results from monocular vision. But this is not suitable for our use because the Tele-MAGMaS project is meant to work in unstructured environments without any prior information.

From the above discussion, Stereo Vision and RGB-D cameras seem to be the only convenient and practical solution. Hence, they were chosen over others.

8.2 Visual SLAM (V-SLAM)

A brief idea about SLAM and V-SLAM is presented in this section.

8.2.1 SLAM

SLAM is the acronym for Simultaneous Localization and Mapping. It is a process in which a mobile robot, starting at an unknown initial location, in an unknown environment, incrementally builds a map of the environment and at the same time uses this map to localize itself. SLAM is the basis of navigation in most autonomous mobile robots.

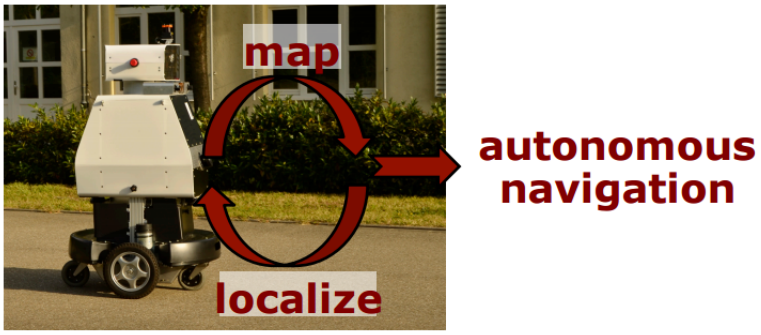

FIGURE 8.1: Navigation using SLAM

The Figure 8.1 taken from [31] illustrates the SLAM philosophy. The Robot simultaneously builds a map and localizes and then navigates autonomously. The successful application

of SLAM in a variety of robots ranging from robotic vacuum cleaner to planetary rovers, from aerial robots to autonomous undersea vehicles motivated us to explore its use in the Tele-MAGMaS project.

There are many approaches for implementing SLAM, important among them are:

1. **Kalman Filter(KF) and Extended Kalman Filter(EKF) based SLAM**: Presented here are some points regarding KF and EKF based SLAM.

   - KF based SLAM was the first ever implementation to solve the SLAM problem.
   - KF based SLAM is applicable only to linear systems and EKF based SLAM is applicable to non linear systems, linearized once around an operating point. So, EKF can diverge if non-linearities are large.
   - High computational complexity because the covariance matrix's dimension is $(6+N)\times(6+N)$, and inverting such a matrix (in real time) can be cumbersome. Here N is the number of landmarks.
   - Owing to the large computational complexity, it cannot be used for Large scale SLAM.
   - Simple formulation and implementation.

2. **Sparse Extended Information Filter(SEIF) based SLAM**: Presented here are some points regarding SEIF based SLAM.

   - SEIF SLAM is an efficient approximation of the Extended Information Filters for the SLAM problem.
   - Applicable to non linear systems like the EKF SLAM, linearized once around an operating point.
   - Constant time complexity, so it does not depend on the number of landmarks detected.
   - Owing to the constant time complexity, it can be used for Large scale SLAM.
   - The quality of SLAM is not as good as the EKF based approach.
   - Implementation is not easy.

3. **Particle Filter(PF) based SLAM**: Presented here are some points regarding Particle Filter based SLAM:

   - The Pose can belong to any probability distribution, unlike the previous cases which are strictly Gaussian.

Qualitative comparison of different SLAM approaches					
	KF	**EKF**	**SEIF**	**PF**	**Graph-SLAM**
Computational Complexity	N^2	N^2	Constant time complexity	$M \log N$	Linear in number of edges
Distribution Assumption	Gaussian	Gaussian	Gaussian	Pose: Any, Landmarks: Gaussian	Gaussian + Outliers
Linearization	Only Linear Systems	Non Linear Systems linearized only once	Non Linear Systems linearized only once	No linearization needed	Relinearization at every iteration
Large Scale SLAM	Not suitable	Not suitable	Suitable	Suitable	Suitable

TABLE 8.1: Qualitative comparison of different SLAM approaches, N: number of landmarks, M: number of particles

- No linearization is need for the robot's motion model.
- Time complexity is logarithmic in the number of landmarks, so it is not as complex as the KF/EKF based approaches.
- Suitable for Large Scale SLAM.

4. **Graph based SLAM** Presented here are some points regarding Graph based SLAM: Presented here are some points regarding Graph based SLAM:
   - This method implements the least square approach to solve the SLAM problem.
   - Complexity is linear in the number of edges.
   - It assumes a Gaussian distribution but can take care of outliers easily.
   - Relinearization is done at each iteration so it will not diverge as much as EKF or SEIF.
   - It is suitable for Large Scale SLAM.
   - It is one of the state of the art SLAM algorithms.

A summary of all the approaches is given in Table 8.1. From the comparison we can conclude that Graph SLAM is the most suitable approach for implementing in our system because its computational complexity is low, it relinearizes the model at every iteration, it can take into account the effect of outliers, etc.

8.2.2 V-SLAM

Since we are using vision sensors for state estimation, we will be dealing with the problem of visual SLAM, also called V-SLAM. V-SLAM is very popular in the robotics community, as a consequence of advantages of visual sensor over other alternatives. In Section 8.1 the benefits of using Stereo and RGB-D camera is discussed and in this Section a brief description about commonly available, state of the art stereo/RGB-D development kits is given. These developement kits have their own SLAM algorithms implemented on their onboard computers. The SLAM algorithms work out of the box but if the user wants, he/she can replace the onboard SLAM algorithm with any other SLAM algorithm of his/her choice.

Two methodologies have become popular in V-SLAM:

1. Filtering based methods to fuse data from all images with a probability distrubution. For eg. EKF based SLAM.
2. Non filtering based methods, also called the key frame based methods which rely on optimization technics like Bundle Adjustment(BA) for uncertainity reduction. For eg. Graph based SLAM.

As discussed in Sub Section 8.2.1, Graph based SLAM algorithms are superior in all respect to EKF based SLAM (Table 8.1) but the caveat with Graph SLAM is that we do not get a covariance matrix associated with the pose which can give us a qualitative measure of how noisy the data obtained is.

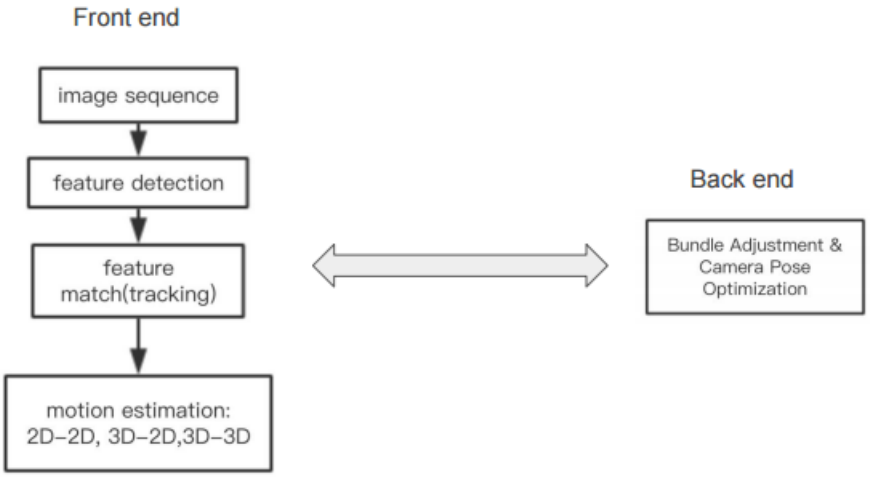

FIGURE 8.2: Visual SLAM, Front End and Back End Flow Chart

A simplified flow chart of the front and backend of a general V-SLAM algorithm is shown in Figure 8.2. In simple terms, the vision sensor first acquires an image from the environment, then features are detected in it. The same features are tracked/matched in the next incoming image. The relative transformation in the image coordinates helps to find the relative transformation in the world frame (or any other reference frame) in terms a transformation matrix between the current and the previous pose. Similarly the transformation matrices are recovered incrementaly from one camera pose to the next and so on and finally all the acquired transformations are concatenated to give the current estimate of the pose with respect to the starting point or any other reference frame. While all of this happens in the front end, in the back end a pose graph optimization is done and minimization of reprojection error (Bundle Adjustment) is done to reduce the uncertainity in the acquired poses. In V-SLAM algorithms two types of Bundle Adjustment(BA) are done, one is the local BA which optimizes a window of past poses and the other is the global BA which optimizes the entire pose graph on detecting a loop closure. In local BA the window size is chosen as a compromise between realtime performance and accuracy of the data. The more the window size, the greater the accuracy and slower the computation and vice-versa.

8.2.2.1 Parrot S.L.A.M.dunk

Parrot S.L.A.M.dunk is an integrated kit which has a stereo camera and a plethora of other sensors like ultrasound transmittor/receiver, an IMU, a barometer, etc. Most importantly it has an onboard computer which runs Ubuntu 14.04. It has the ROS middleware onboard to acquire data from all of these sensors and feed this data to a native SLAM algorithm. The SLAM algorithm publishes the pose of the sensor and the point cloud generated by it. This data is published as rostopics. In addition to the pose and the point cloud the user also has access to other data like the left and right images, the depth image, the barometer data. etc., all published as ROS topic. The SLAM algorithm implemented on the device is not open sourced. Besides SLAM, it has other functionalities like obstacle detection, etc. The user is at his/her luxury to use the device's SLAM algorithm or run his/her own algorithm onboard the device or on a PC. The device is very ergonomic and light weight (140 g). It can be connected to a PC by a USB cable or by means of WiFi connection. This device needs an external source of power.

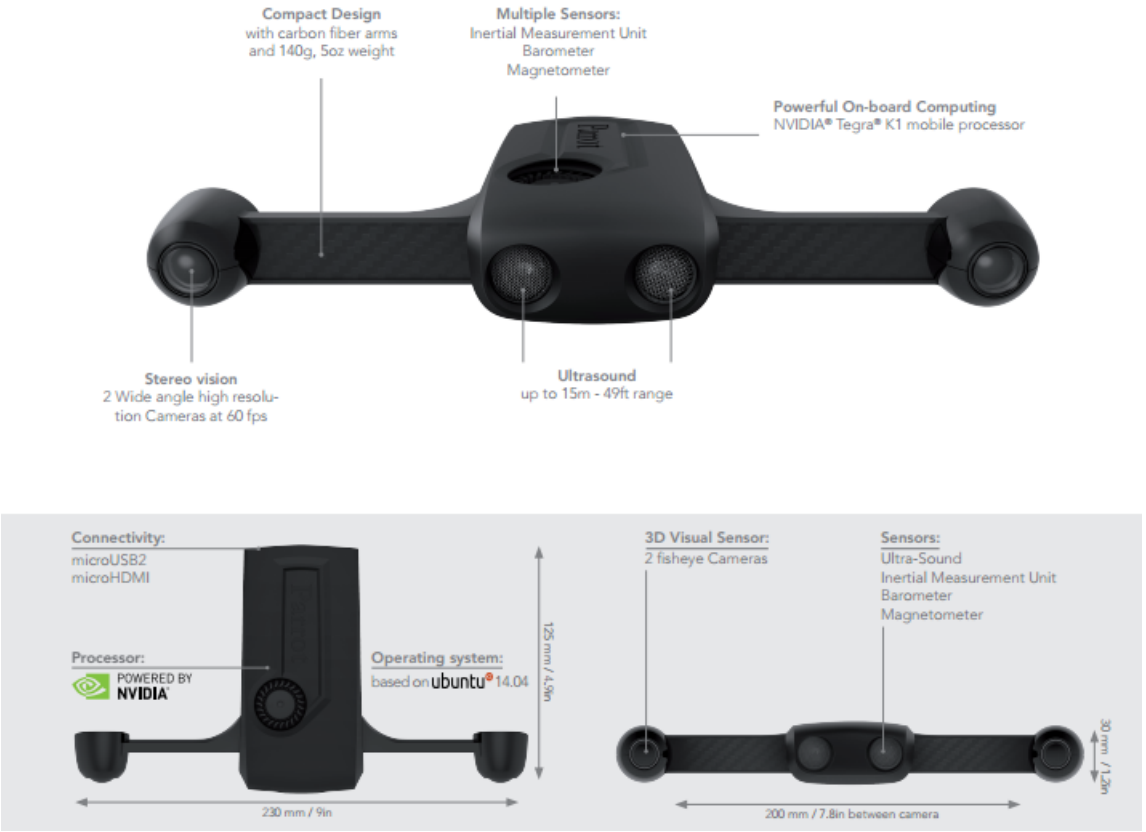

FIGURE 8.3: The Parrot S.L.A.M.dunk

8.2.2.2 Intel Euclid

The Intel Euclid is also a very recently launched product by Intel. This device is different from both stereo and RGB-D cameras. It takes the positive aspects of stereo vision and RGB-D cameras both. It has an Infrared Stereo Camera setup, an infrared projector, a fisheye camera and a color camera. It implements the so called Active Stereo Vision based technology to acquire depth images. Unlike the RGB-D camera it doesn't project a pattern of IR light on the scene, rather it illuminates a scene with a simple IR projector just like a torch illuminates a scene and uses the IR stereo camera setup to determine the depth image. Unlike the RGB-D camera it can be used outdoors with features in the range of 0.5 to 4 meters. In strong sunlight, the depth data may be erroneous because of infrared light emitted from the surrounding.

Just like the Parrot S.L.A.M.dunk, Intel Euclid also has a number of other sensors like the IMU and runs Ubuntu 16.04 on an onboard computer. It has ROS pre-installed and all the data is published as rostopics. It has its own SLAM algorithm called the RealSense SLAM

which is not opensourced. It is lightweight, compact and ergonomic. It has an onboard detachable lithium ion battery and weighs 146 g with the battery. It has a web based user interface that cab be used to run ROS nodes or a group of nodes. The data published by this device includes the 6D pose of the center of projection of fisheye camera, the depth images, the image from color and fisheye camera, the IMU data, etc. One can choose to either use the the native SLAM algorithm or any other SLAM package or both. It is lighter than Parrot S.L.A.M.dunk even with an onboard battery. A dedicated onboard battery for vision sensor is a major advantage in the field of aerial robotics because it allows the system to be completely wireless. As the Intel Euclid has it's own battery, it neither needs additional wires nor takes a share of the power meant for running the motors of the aerial robot.

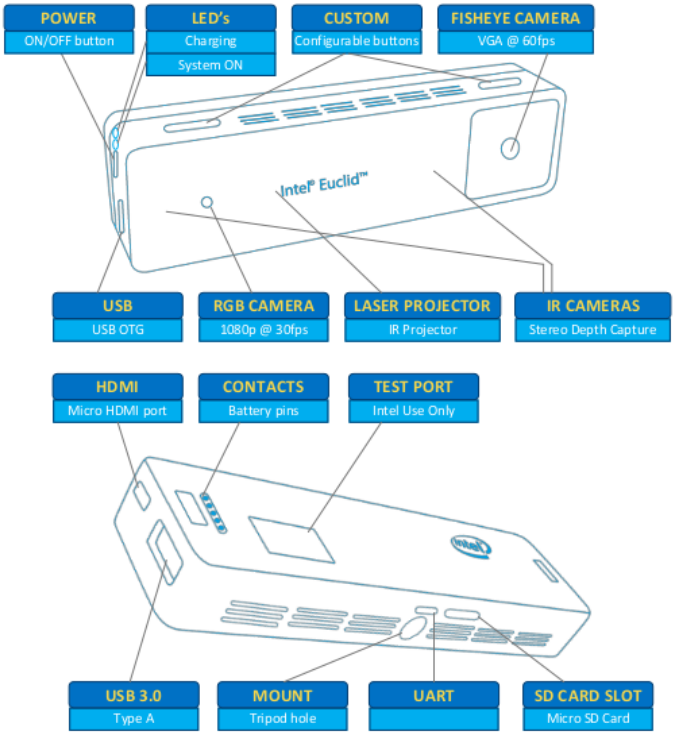

FIGURE 8.4: Intel Euclid

A short technical comparision between the Parrot S.L.A.M.dunk and the Intel Euclid is given in the Table 8.2. From the technical comparison and the description given above, it is evident that Intel Euclid is a better platform as compared to the Parrot S.L.A.M.dunk. The Intel Euclid has a more powerful computer and more RAM and storage. This leaves enough room for installing our own algorithms and testing it along with the native algorithm.

Summarizing, the Intel Euclid has the following clear advantages:

- It has its own onboard source of power, a light weight battery.
- It works on the state of the art, active stereo vision technology.
- It weighs almost the same as the Parrot S.L.A.M.dunk.
- It has better more computing power than the Parrot S.L.A.M.dunk.
- It has an easy to use web user interface.
- It is completely wireless and connects primarily by WiFi without using any dongle, etc.
- Last, but not the least, it is available at less than half the price of Parrot S.L.A.M.dunk.

TABLE 8.2: Parrot S.L.A.M.dunk and Intel Euclid Technical Details

Parrot S.L.A.M.dunk and Intel Euclid Technical Features		
	Parrot S.L.A.M.dunk	**Intel Euclid**
Vision Technology	Passive Stereo Vision	Active Stereo Vision
Cameras	Stereoscopic vision: • Stereo setup of fisheye cameras • Stereo baseline: 20 cm • Rolling shutter type • 960p, 30 fps (adjustable)	Multiple cameras used: 1. IR Stereo Setup: • Stereo baseline: 7 cm • 640 × 480, 30 fps (adjustable) 2. Fisheye: • Global Shutter • 640 × 480, 30 fps (adjustable) 3. Color Camera: • Rolling Shutter • 320 × 240, 30 fps (adjustable)
IR Projector	None	IR Projector with a baseline of 2 cm with respect to the left IR camera, between the left and the right IR cameras
Sensors	Ultrasonic, IMU, Barometer and Magnetometer	Accelerometer, Digital Compass, Barometer, Altimeter, Temperature Sensor and Humidity Sensor

Onboard Computer	Processor: NVDIA Tegra K1, RAM: 2 GB DDR3, Memory: 16 GB	Processor: Intel Atom x7-Z8700, Quadcore CPU, Burst Speed 2.4 GHz, Intel Gen8 LP GPU, RAM: 4 GB, Memory: 32 GB (+ 128 GB microSD)
Operating System	Ubuntu 14.04	Ubuntu 16.04
SDK	ROS Node	ROS Node
Connectivity	Micro USB 2.0 OTG, USB 3.0 Host, Display: Micro HDMI, WiFi: Through USB Dongle	WLAN, Bluetooth
Battery	None	Available

The advantages of the Intel Euclid platform encouraged us to procure the Intel Euclid Sensor for testing and experimentation.

8.3 Comparative Study of Intel Euclid RealSense SLAM and ORB_SLAM2

The Intel Euclid has its own SLAM algorithm installed onboard to help robots map and localize in an environment. The SLAM algorithm running onboard the Euclid is called the RealSense SLAM algorithm. This algorithm is not opensource but its API is opensourced. Since Euclid also publishes all the data collected by its sensors (as rostopics), one can also integrate his/her own SLAM algorithm to estimate the state of the robot.

In this Section a comparitive study between the native RealSense SLAM [3] and the open sourced ORB_SLAM2 is presented [4]. Before delving into comparision, a brief overview of the RealSense SLAM and ORB_SLAM2 is given in the following subsections.

In general, the following common points must be kept in mind while running SLAM in any Stereo or RGB-D camera:

1. The quality of the cameras' view is important. Ideal conditions are well-lit areas, without large changes in lighting.
2. The scene must have a lot of texture, the device does not use any visual markers so it needs a lots of features in the scene to localize itself. A more complex scene with lots of objects or geometry is best

[3] `https://software.intel.com/sites/products/realsense/slam/developer_guide.html`
[4] `https://github.com/raulmur/ORB_SLAM2`

3. A lot of reflective surfaces such as glass or mirrors can be problematic.
4. The scene must be static, i.e. there must not be any moving objects.
5. SLAM works best when the camera motion is mostly translation, as opposed to rotation.
6. Rotations must be done slowly, zero radius turns are difficult for the tracker to follow, so it's best to rotate while translating.
7. For RGB-D camera or Active Stereo Vision Systems like the Intel Euclid, too much sunlight can be problematic.

8.3.1 RealSense SLAM

RealSense SLAM uses a fisheye camera, an IMU and a depth camera to localize itself and build a map of the enviroment. Since it combines information from the IMU, RealSense SLAM is an advanced version of Visual SLAM and hence called Visual-Inertial SLAM. It does not need any prior information about the environment to localize itself, no visual markers are needed.

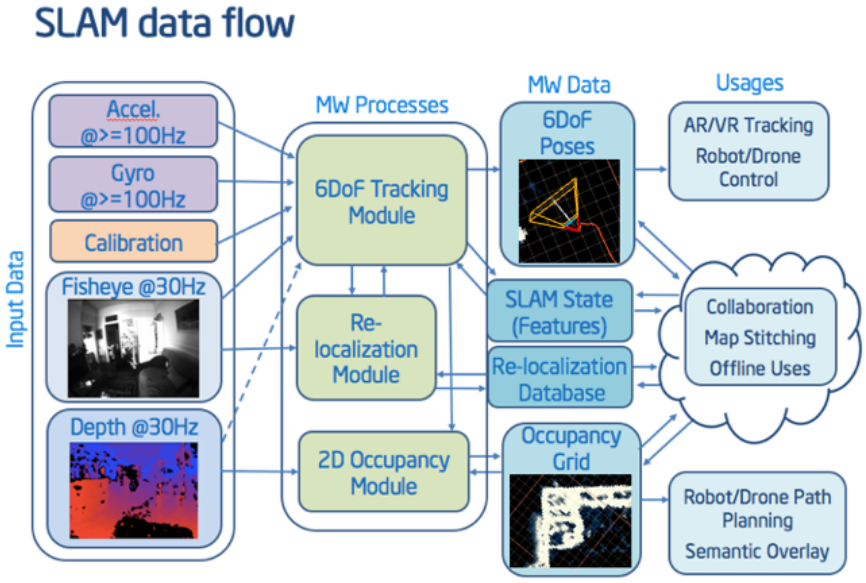

FIGURE 8.5: RealSense SLAM Pipeline

The RealSense SLAM pipeline is shown in Figure 8.5. In the context of our application in the Tele-MAGMaS project, the 6-DoF tracking module is most important. The 6-DoF tracking gives as output the pose of the origin of the projection center of the fisheye

camera, in real time at 30 Hz. Since RealSense technology is not opensourced, there is not much information about the SLAM algorithm running on board the Intel Euclid.

8.3.2 ORB_SLAM2

The ORB_SLAM2 is an opensource V-SLAM library that implements a Graph based V-SLAM algorithm for computing camera trajectory in real time [19]. This library can be used for Monocular, Stereo and RGB-D cameras. Unlike the RealSense SLAM algorithm, the ORB_SLAM2 algorithm does not use any IMU data for estimating the camera pose. Therefore, while RealSense SLAM is a Visual Inertial SLAM algorithm, ORB_SLAM2 is a Visual SLAM algorithm.

The ORB_SLAM2 also comes with a ROS wrapper which makes its easy to integrate with sensors publishing data in rostopics. Hence this library can be integrated with the Intel RealSense device.

ORB_SLAM2 is a feature based SLAM algorithm. It uses the ORB (Oriented FAST and rotated BRIEF) features. ORB features are used because it allows real time performance without the requirement of GPU acceleration.

When running for RGB-D cameras, the ORB_SLAM2 takes the following inputs:

- Color or Gray Scale Image.
- Depth Registered Images.

So, it can be integrated with any device which gives the above two data streams. Besides, one also has to give as an input the parameters like the camera parameters, the baseline between the IR projector and receiver and the Depth Map Factor, but these are generally fed in through a yaml file.

The ORB_SLAM2 pipeline is shown in Figure 8.6. It runs three threads in parallel, Tracking, Local Mapping and Loop Closing. These three parallel threads can create a fourth thread for full Bundle Adjustment after loop closure.

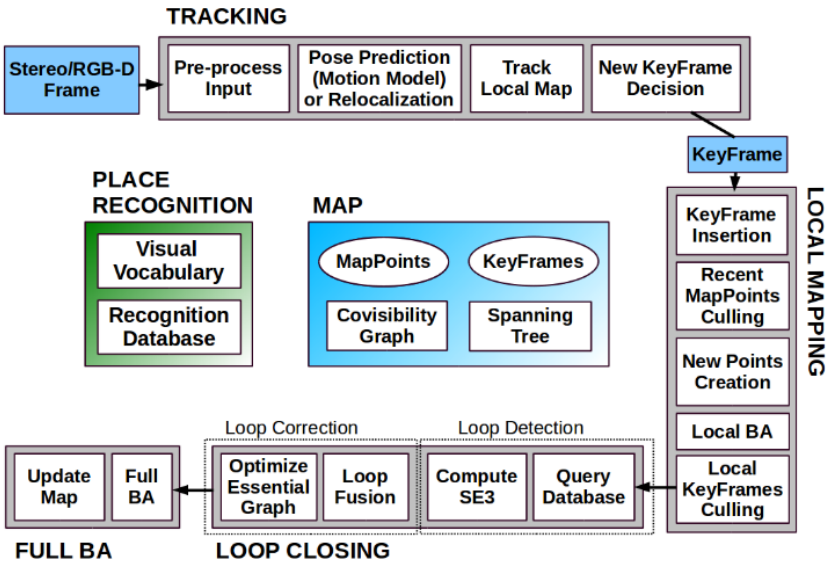

FIGURE 8.6: ORB_SLAM2 Pipeline [19]

Since ORB_SLAM2 is opensourced, all the information about it can be found in [19] and [32].

The Tracking thread takes care of every incoming frame from the RGB-D/Stereo/Monocular Cameras, preprocesses the incoming data and decides if it is necessary to insert a new pose (or keyframe). It detects changes in the scene. If a moving object comes in an otherwise static scene then new key frames are inserted even if the camera is static. Therefore, this algorithm cannot be used in an environment which has a lot of moving objects. If tracking is lost due to sudden motion or occlusions then the place recognition module is activated to relocalize the camera.

The Local Mapping thread processes the new keyframes sent by the tracking module and local BA is performed to reduce uncertainty in the pose.

The Loop Closing thread searches for loops with every new incoming keyframe. On detecting a loop, a similarity transform is done to give an idea about the drift accumulated in the loop. After the drift has been calculated, both the extremes of the loop are aligned and the points which were same in both the beginning and the end of the loop are fused. A pose graph optimization is then done to make the loop globally consistent.

8.3.3 Results

In this subsection, the comparative results between RealSense SLAM and ORB_SLAM2 is discussed. The RealSense SLAM algorithm always runs on the computer available onboard Intel Euclid but the ORB_SLAM2 can be run on the Intel Euclid or on a ground PC, but in the second case it has to communicate with the sensors of Intel Euclid via WiFi which may fail sometimes because of interference from other WiFi hotspots or because of other users using the same WiFi. Loss of data packets during WiFi transmission may deteriorate the performance of the controllers running on the Aerial Robot. The robot may exhibit unpredictable behavior which may not be safe for the people surrounding the testing area. Hence it is advisable to run the ORB_SLAM2 algorithm on the Intel Euclid's computer which is mounted on the aerial robot.

Running the ORB_SLAM2 on the Intel Euclid has its own demerits because ORB_SLAM2 may be computationally too intensive for the miniature computer on the Intel Euclid. It should be noted that the makers of ORB_SLAM2 implemented and ran this algorithm on an Intel Core i7-4790 desktop computer with 16 Gb RAM which is far superior than the computer on Intel Euclid.

8.3.3.1 ORB_SLAM2 running on a PC

In this case, the ORB_SLAM2 algorithm is running on a ground PC and the RealSense SLAM algorithm is running on the Intel Euclid. The ground PC receives the data from the color images and the depth images from the Intel Euclid using WiFi and feeds it into a ros node running in it. The data logging is done in Simulink with the help of a rosix server [5]. The result of this trial is shown in Figure 8.7. The RealSense SLAM appears erroneous. Except the pitch and yaw, all the other quantities are not consistent with the ground truth generated by the motion capture, but the ORB_SLAM2 algorithm seems to track the ground truth better than RealSense SLAM. At about 70 seconds, the ORB_SLAM2 loses track for sometime and it is visible from the flat curve at that time in all the plots corresponding to ORB_SLAM2 (blue). It is because the algorithm running on the PC did not receive the images published by Euclid during that time owing to some data packets lost in WiFi transition, but the positive aspect about ORB_SLAM2 is that when the tracking is resumed it, stays closer to the ground truth. Since it is not desirable to have data loss in transmission it is not advisable to run the algorithm in the ground PC. So, the next trials are done with ORB_SLAM2 running on the Euclid.

[5]The rosix server is like the genomix server, refer Chapter 6 for more information.

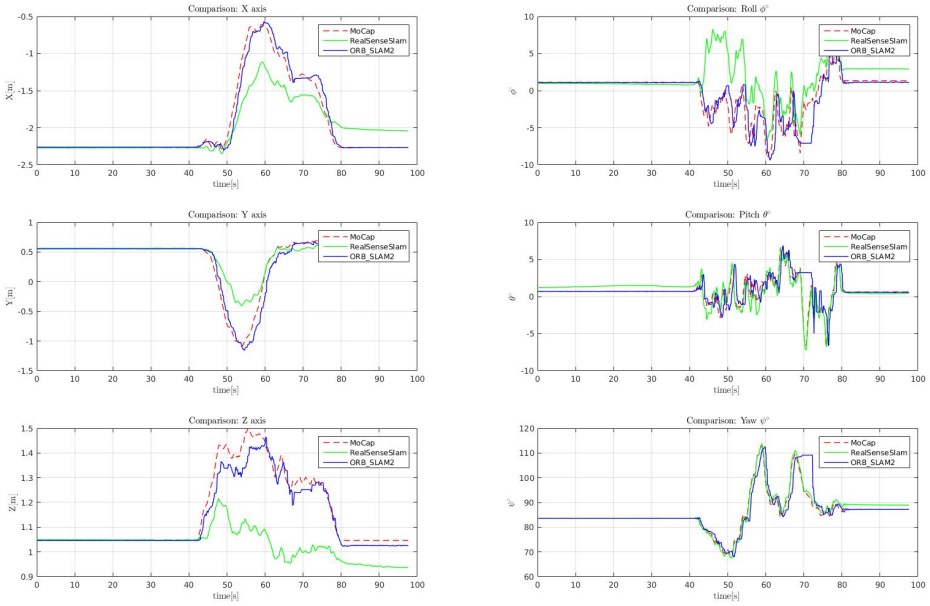

FIGURE 8.7: ORB_SLAM2 running on PC, RealSense on Intel Euclid

8.3.3.2 ORB_SLAM2 running on Intel Euclid

In the first trial (Figure 8.8) it is clearly visible that the position tracking is better with ORB_SLAM2 than with RealSense SLAM and the orientation tracking is satisfactory with both the algorithms. It should be noted that the trajectories for each trial are rather different from one trial to another, so it may not be the best for comparison.

In the second trial (Figure 8.9), one can draw the same conclusion as the first trial but it is visible in the plots that ORB_SLAM2 briefly loses track between 40-60 seconds and 80-100 seconds but correctly relocalizes after a few seconds. Between 40-60 seconds it is prominently visible as a spike in the plot for X axis and between 80-100 seconds it can be seen as flat curve in all the plots. It may be because ORB_SLAM2 is too computationally intensive for the Intel Euclid's computer. The ORB_SLAM2 was originally tested and bench marked on a superior computer than the Intel Euclid. Besides, for all these trials both RealSense and ORB_SLAM2 are running on the Intel Euclid, in all probability the performance of ORB_SLAM2 will improve if RealSense SLAM is shut down. In addition to this fact, unlike the RealSense SLAM, the ORB_SLAM2 alorithm is not optimized to run on this platform. So, optimizing the ORB_SLAM2 code to run it on the Intel Euclid can solve this problem. The fact that ORB_SLAM2 is computationally intensive can be

verified by looking at the CPU and memory usage of the Intel Euclid when ORB_SLAM2 is running on it It can be seen that all the four cores are completely saturated and the memory usage is also quite high. The RealSense SLAM does not lose track easily because unlike ORB_SLAM2 it fuses the data from the IMU too. Although this is a informed guess, one cannot say anything more about it because the RealSense software is not opensource and its working is not known.

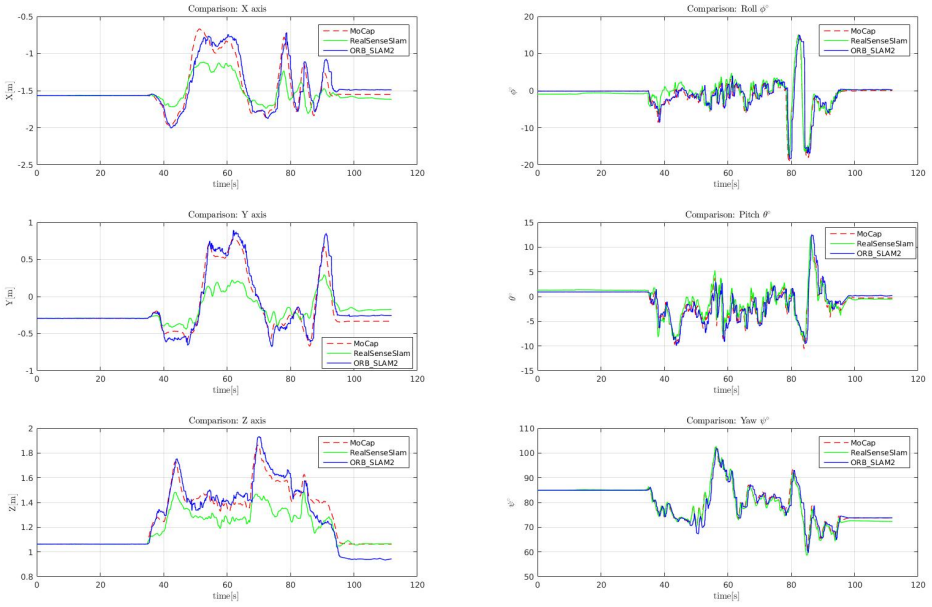

FIGURE 8.8: ORB_SLAM2 running on Intel Euclid, RealSense on Intel Euclid [Trial 1]

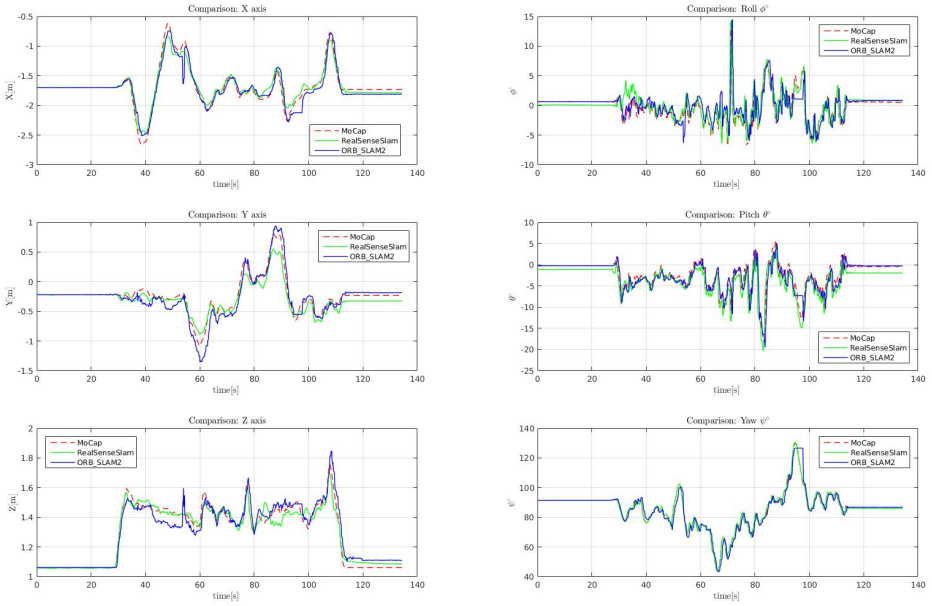

FIGURE 8.9: ORB_SLAM2 running on Intel Euclid, RealSense on Intel Euclid [Trial 2]

In the third trial (Figure 8.10) it is seen again that the position tracking is better with ORB_SLAM2 but there is some drift with both RealSense SLAM and ORB_SLAM2 at the end of trial. ORB_SLAM2 briefly loses track between 60-80 seconds and it is prominently visible in all the postion plots. The orientation tracking is satisfactory for both RealSense SLAM and ORB_SLAM2 but the pitch tracking appears to have some offset in the case of RealSense SLAM.

The fourth trial (Figure 8.11) was done by moving the OTHex faster with some abrupt motions. So it is evident that the position tracking results are not satisfactory. Nevertheless, the positions generated by ORB_SLAM2 is closer to the ground truth than the positions generated by RealSense SLAM. The roll tracking is better in ORB_SLAM2 but the pitch and yaw tracking are better in RealSense SLAM.

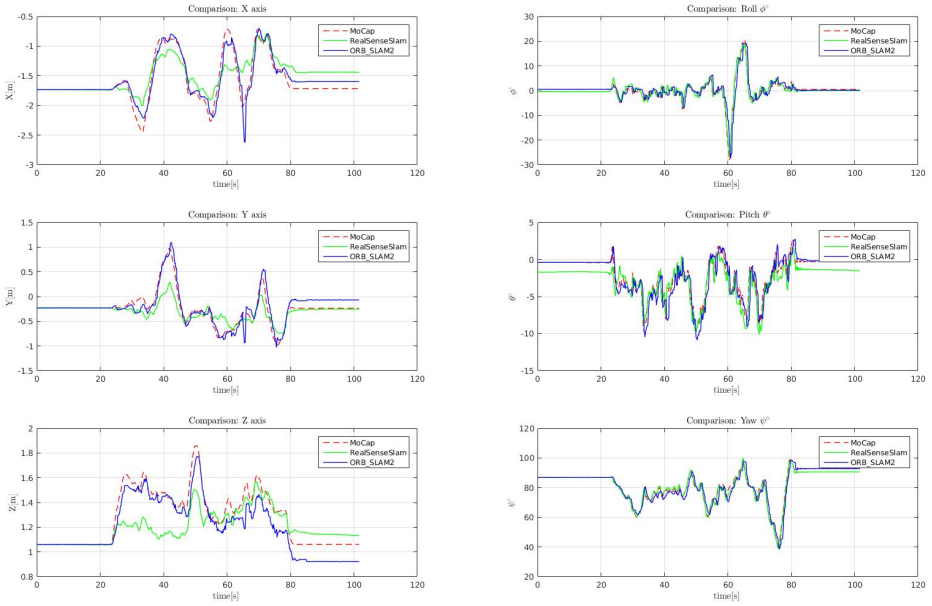

FIGURE 8.10: ORB_SLAM2 running on Intel Euclid, RealSense on Intel Euclid [Trial 3]

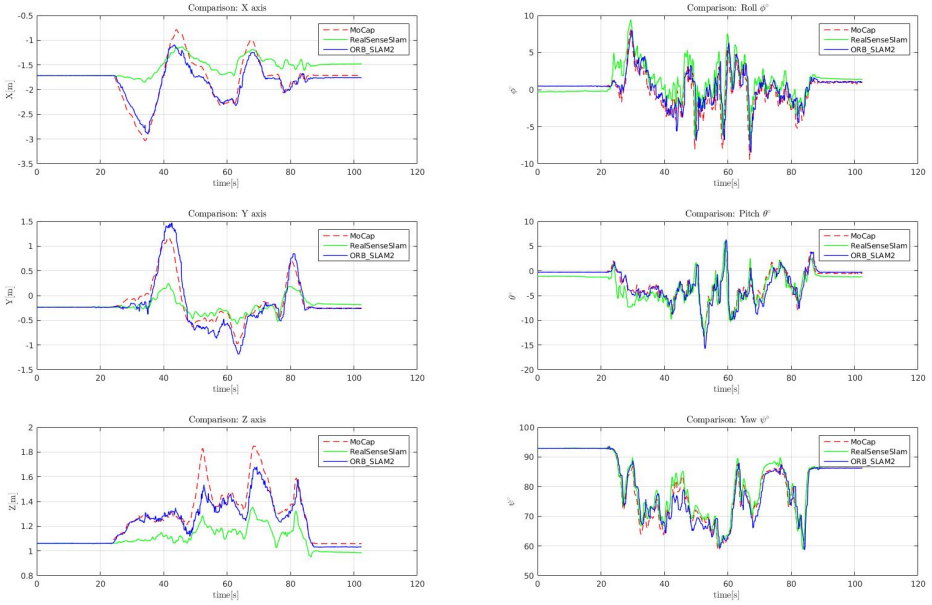

FIGURE 8.11: ORB_SLAM2 running on Intel Euclid, RealSense on Intel Euclid [Trial 4]

The average error for each of the trial for translation along and rotation about each axis is given in Table 8.3. Globally, we can say that the position tracking in ORB_SLAM2 is

		$e_{x_{avg}}$ [cm]	$e_{y_{avg}}$ [cm]	$e_{z_{avg}}$ [cm]	$e_{\phi_{avg}}$ [◦]	$e_{\theta_{avg}}$ [◦]	$e_{\psi_{avg}}$ [◦]
ORB_SLAM2 on PC	ORB_SLAM2	3.6840	3.7977	2.0063	0.6505	0.5292	1.0822
	RealSense SLAM	15.3474	8.9952	11.0132	1.8126	0.5864	0.7522
ORB_SLAM2 on Euclid [Trial 1]	ORB_SLAM2	7.4761	5.9195	4.8141	1.3483	1.0835	1.5816
	RealSense SLAM	12.9230	14.6061	11.2043	0.8395	0.6214	0.6099
ORB_SLAM2 on Euclid [Trial 2]	ORB_SLAM2	6.4076	7.5078	3.6750	1.0179	0.9139	1.7056
	RealSense SLAM	5.3349	6.9323	2.5688	0.6430	1.1685	0.4428
ORB_SLAM2 on Euclid [Trial 3]	ORB_SLAM2	8.2165	9.0154	7.7771	1.0822	0.7461	1.8139
	RealSense SLAM	18.1904	11.3812	11.0682	0.7211	0.9536	1.1265
ORB_SLAM2 on Euclid [Trial 4]	ORB_SLAM2	8.3063	10.3748	4.6970	0.8287	0.9522	1.7415
	RealSense SLAM	26.6494	15.5209	15.8149	1.2583	0.8264	1.1310

TABLE 8.3: Average errors: ORB_SLAM2 and RealSense SLAM

significantly better than RealSense SLAM and the orientation tracking in RealSenseSLAM is marginally better than ORB_SLAM2.

8.4 Conclusion

It is easy to conclude from the discussion in this Chapter that vision based state estimation is the preferred solution if the weight and power consumption are the deciding constraints. Moreover vision based systems are cheaper than other alternatives and the presence of many opensource algorithms to implement Visual Odometry or Visual SLAM makes it much more preferable. The advantages and disadvantages of three fundamental vision sensors, i.e. monocular cameras, stereo cameras and RGB-D cameras have also been elucidated in this Chapter. Among these alternatives stereo and RGB-D cameras are preferable owing to the fact that with these sensors there is no scale ambiguity because one can easily get the depth information from these cameras. A market survey was also done to choose the best sensor for our system and Intel Euclid was preferred over the Parrot S.L.A.M.dunk owing to its low cost, light weight, the availability of onboard power supply (i.e. a battery), a more powerful computer, the option to use WiFi without the need of an additional dongle and the use of Active Stereo Vision Technology. The Active Stereo Vision Technology is more usable outdoors that the RGB-D technology.

In addition to this, the various types of general SLAM algorithms have been discussed and it is concluded that the Graph based SLAM algorithms are more suitable for real time application.

In the last part a comparison is made by doing experiments with a widely used opensource SLAM algorithm called the ORB_SLAM2 and the closed source native SLAM algorithm of Intel Euclid, better known as the RealSense SLAM. It can be concluded that ORB_SLAM2 is a better choice as compared to RealSense SLAM because, on an average its performance is better than RealSense SLAM and since it is opensourced, a lot of help is available online. Moreover it is easy to understand the internal working of ORB_SLAM2, so it is not difficult to predict what is going wrong and make necessary changes. the RealSense SLAM shows a marginally better tracking of the orientation probably because it uses the data from the IMU, so the final solution could be to use the position data from the ORB_SLAM2 algorithm and the orientation data from RealSense SLAM.

Finally, to fully integrate this system in to the Tele-MAGMaS, a lot more experimentation needs to be done. The flying arena needs to be modified to make it more conducive to implement vision based state estimation. Some of the primary modifications could be to cover the glass windows with some textured drapes and cover the shiny floor with a textured mat. Since the ORB_SLAM2 and the RealSense SLAM do not generate any covariance associated to the pose published by them, it is difficult to integrate them with the UKF[6] implementation used at LAAS-CNRS. The covariance is not available from ORB_SLAM2 because it uses Graph based SLAM approach which uses Bundle Adjustment (BA) to minimize the uncertainty. It is not clear how the RealSense SLAM works but its API has no function/getter to give the covariance associated with the pose as output. This problem has been addressed in [33], but its implementation in real time can be an issue owing the large size of the matrices involved.

[6] the pom-genom component

Chapter 9

Conclusion and Future Work

In conclusion, I want to highlight my contribution to the Tele-MAGMaS project and then mention the future work related it.

Primarily, I was involved in the joint development of a genom3 component for the KUKA iiwa LBR 14 R820. This component was an interface between the real hardware and the high level control algorithms written in MATLAB/Simulink, it has inverse kinematic capabilities allowing both control of the arm in joint and Cartesian space, it makes possible the control of the gripper mounted on the flange of the arm, and also makes available to the high-level controller the arm state information such as joint position, joint torque and Cartesian position. This was a joint work supervised by a PhD student. I also tested and tuned several V-Rep environments, which were necessary for the visualization of the final and intermediate experiments and for the simulations used during the early phase of the control software development. The V-Rep environment communicated with MATLAB/Simulink using s-functions which were already provided to me. Additionally I parametrized the key system information display in V-Rep, using functions primarily developed at University of Rennes and the IRISA lab. I also worked on the integration and testing of the omega.6 haptic device. The framework was already developed by our partners in the University of Siena, Italy, I was involved in integrating it to our systems in LAAS-CNRS. I also worked on the trajectory generator, especially for the KUKA iiwa LBR 14 R820, this was the part of the overall task planner. This work involved the description of the task through state and associated waypoints to grasp and manipulate the bar alone by the ground manipulator. In this primary phase of the project, i.e. using the manipulator alone, we relied on the V-REP inverse kinematics, part of my job was to ensure that V-Rep

Inverse Kinematics solver does not generate unusual joint configurations for the given waypoints. I also worked, by assisting a PhD student, on the overall integration of various subcomponents developed by numerous people working on the Tele-MAGMaS project, until participating in the final demonstration at LAAS-CNRS for a panel of potential industrial partners.

My second main contribution was to investigate the use of vision sensor for state estimation. To this end, I went through the available literature on SLAM, Visual SLAM, Visual Odometry and the use of visual sensors for state estimation of aerial robots, this followed a complete market survey of the available associated sensor choices. Based on my report, we went for an prototypical sensor form Parrot, a French company. I worked in hand with them to procure and integrate one of the Parrot S.L.A.M.dunk and along the path we discovered that its performances did not match our expectations.

Since the device by Parrot was not satisfactory, we shifted our focus to another sensor by Intel, called the Intel Euclid. The major milestones while working with Euclid was to use rosix for data acquisition in MATLAB/Simulink and to tune the ORB_SLAM2 to make it usable with the Intel Euclid. In the process I also briefly tested many other Visual Servoing/SLAM/Odometry libraries available online, like ViSP (Visual Servoing Platform), SVO(Semi-direct monocular Visual Odometry), rovio(Robust Visual Inertial Odometry) and okvis(Open Keyframe-based Visual-Inertial SLAM) to see if they are easily integrable with Intel Euclid. I found that ORB_SLAM2 is integrable very easily. Although Intel Euclid has its own SLAM algorithm, the RealSense SLAM, the preliminary results obtained convey that ORB_SLAM2 package has a better performance and can be integrated in the Tele-MAGMaS project after some more tests and experimental validations. I also conducted the mechanical integration of the Intel Euclid sensor on our testbed, by getting designed and 3D printing some structural components by a mechanical engineer in our group, this was of paramount importance to test the performances of the sensor and algorithms in real flight configuration.

As far as the future of the Tele-MAGMaS is considered, the immediate plan follows three main research directions. First, to completely integrate this vision sensor with the OTHex flying platform so that the system can also work in outdoor or indoor environments without MoCap system. Second, to exploit the full capacity of the generic framework to use more than one aerial robots in the Tele-MAGMaS, in order to increase the aerial manipulators physical action on the load, and also allow oddly shaped loads to be manipulated by the full Tele-MAGMaS. And lastly to develop the control framework and conduct the full

integration, both soft- and hard-ware, for mobile ground manipulators instead of static ground manipulator, thus paving the way for the use of Tele-MAGMaS in the real world.

Bibliography

[1] R. R. Murphy. Human-robot interaction in rescue robotics. In *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*.

[2] M. Mohammadi, A. Franchi, D. Barcelli, and D. Prattichizzo. Cooperative aerial tele-manipulation with haptic feedback. In *2016*, pages 5092–5098, Daejeon, South Korea, Oct. 2016.

[3] S. J. Tricamo and F. L. Swern. Control of multiple robotic arms engaged in cooperative manipulation and assembly operations. In *[Proceedings] 1988 International Conference on Computer Integrated Manufacturing*.

[4] A. Petitti, A. Franchi, D. Di Paola, and A. Rizzo. Decentralized motion control for cooperative manipulation with a team of networked mobile manipulators. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*.

[5] Bruno Siciliano and Oussama Khatib. Cooperative manipulators. In *Springer Handbook of Robotics*.

[6] V. Kumar. Aerial robot swarms. In *2013 ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2013.

[7] Q. Jiang and V. Kumar. The inverse kinematics of cooperative transport with multiple aerial robots. *IEEE Transactions on Robotics*, 2013.

[8] G. Gioioso, A. Franchi, G. Salvietti, S. Scheggi, and D. Prattichizzo. The flying hand: A formation of uavs for cooperative aerial tele-manipulation. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*.

[9] T. Petrovic, T. Haus, B. Arbanas, M. Orsag, and S. Bogdan. Can uav and ugv be best buddies? towards heterogeneous aerial-ground cooperative robot system for complex aerial manipulation tasks. In *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*.

[10] A. Franchi, C. Secchi, M. Ryll, H. H. Bulthoff, and P. R. Giordano. Shared control : Balancing autonomy and human assistance with a group of quadrotor uavs. *IEEE Robotics Automation Magazine*, 2012.

[11] A. Franchi, C. Secchi, H. I. Son, H. H. Bulthoff, and P. R. Giordano. Bilateral teleoperation of groups of mobile robots with time-varying topology. *IEEE Transactions on Robotics*.

[12] H. Yang and D. Lee. Hierarchical cooperative control framework of multiple quadrotor-manipulator systems. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4656–4662, May 2015.

[13] H. N. Nguyen, S. Park, and D. Lee. Aerial tool operation system using quadrotors as rotating thrust generators. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1285–1291, Sept 2015.

[14] G. Gioioso, M. Ryll, D. Prattichizzo, H. H. Bülthoff, and A. Franchi. Turning a near-hovering controlled quadrotor into a 3d force effector. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*.

[15] G. Gioioso, M. Mohammadi, A. Franchi, and D. Prattichizzo. A force-based bilateral teleoperation framework for aerial robots in contact with the environment. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*.

[16] N. Staub, M. Mohammadi, and A. Franchi. Towards robotic magmas: Multiple aerial-ground manipulating systems. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*.

[17] D. Scaramuzza and F. Fraundorfer. Visual odometry: Part 1 - the first 30 years and fundamentals. *IEEE Robotics and Automation Magazine*, 2011.

[18] Zheng Fang and Yu Zhang. Experimental evaluation of rgb-d visual odometry methods. *International Journal of Advanced Robotic Systems*, 2014.

[19] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[20] King Sun Fu, R. C. Gonzalez, and C. S. G. Lee. *Robotics: Control, Sensing, Vision, and Intelligence*. McGraw-Hill, Inc. New York, NY, USA, 1987.

[21] T. Lee, M. Leoky, and N. H. McClamroch. Geometric tracking control of a quadrotor uav on se(3). In *49th IEEE Conference on Decision and Control (CDC)*, pages 5420–5425, Dec 2010.

[22] Markus Ryll, Giuseppe Muscio, Francesco Pierri, Elisabetta Cataldi, Gianluca Antonelli, Fabrizio Caccavale, and Antonio Franchi. 6d physical interaction with a fully actuated aerial robot. In *2017 IEEE International Conference on Robotics and Automation*.

[23] S. Rajappa, M. Ryll, H. H. Bülthoff, and A. Franchi. Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*.

[24] G. Jiang and R. Voyles. A nonparallel hexrotor uav with faster response to disturbances for precision position keeping. In *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014)*.

[25] D. Brescianini and R. D'Andrea. Design, modeling and control of an omni-directional aerial vehicle. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*.

[26] S. Park, J. Her, J. Kim, and D. Lee. Design, modeling and control of omni-directional aerial robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[27] M. Ryll, D. Bicego, and A. Franchi. Modeling and control of FAST-Hex: a fully-actuated by synchronized-tilting hexarotor. In *2016*, pages 1689–1694, Daejeon, South Korea, Oct. 2016.

[28] Ayssam Elkady and Tarek Sobh. Robotics middleware: A comprehensive literature survey and attribute-based bibliography. *Journal of Robotics*, 2012.

[29] D. Brugali and P. Scandurra. Component-based robotic engineering (part i) [tutorial]. *IEEE Robotics Automation Magazine*, 16(4):84–96, December 2009.

[30] D. Brugali and A. Shakhimardanov. Component-based robotic engineering (part ii). *IEEE Robotics Automation Magazine*, 17(1):100–112, March 2010.

[31] Cyrill Stachniss. Robot mapping course, university of freiburg.

[32] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *CoRR*, abs/1502.00956, 2015.

[33] V. Ila, L. Polok, M. Solony, P. Smrz, and P. Zemcik. Fast covariance recovery in incremental nonlinear least square solvers. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4636–4643, May 2015.