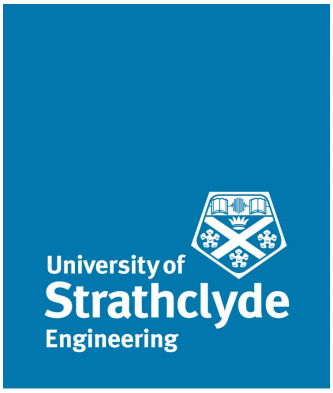


Delay Modelling and State Estimation Techniques for Robust Telepresence Robot Navigation

Submitted by

Barnali Das

In fulfilment of the requirement

for the degree of Doctor of Philosophy

Centre for Ultrasonic Engineering

Department of Electronic and Electrical Engineering

University of Strathclyde, Glasgow, UK

March 09, 2023

Copyright

Abstract

A *telepresence* system allows to perform remote actions using a *telepresence robot* over a distance. The human operator controls the movements of the robot by sending control command signals over a communication channel and receives feedback to acknowledge if the *telepresence robot* has followed the instructions. Telepresence systems recently gained popularity due to their emerging usage in many applications including hospital consultations, remote co-working in offices, security and surveillance, factory inspections or instructor-led education. However, latency constraint introduces major challenges for precise and reliable robotic control in remote environment. Latency (*i.e.,* time delay) can be caused by multiple factors including communication network issues, the physical distance between the human operator and the *telepresence robot*, processing data and system errors. Time delay also produces a visual mismatch between received navigation state feedback and the actual state of the robot in the remote environment, which negatively impacts the human operator's performance.

This thesis aims to address issues related to latency by proposing new state estimation techniques for robust navigation of *telepresence robots* and develops an associated framework and a simulation environment. The thesis can broadly be categorised into three main parts, *1)* a telepresence framework consists of an off-the-shelf commercial (differential-drive) telepresence robot Beam plus, a multi-camera motion tracking system (VICON) and Robot Operating System (ROS); *2)* a new state estimation algorithm called Augmented State Extended Kalman Filter (AS-EKF) that compensates time delay; and *3)* a simulation environment to reproduce the telepresence system with predictive technology using open-source software RViz and Gazebo. Time delay scenarios are considered for both certain and uncertain cases where the latter were modelled using probability density functions (PDF). The results show significant performance improvements compared to the

standard Extended Kalman Filter (EKF) that does not consider delays. The simulation framework offers wider adaptability when a physical system is not plausible, and a controlled experimental environment is desired.

COVID impact statement

In accordance with the University of Strathclyde's guidance, the following COVID impact statement is added.

Pre-Covid-19 research plan

This thesis intended to investigate the issues around time delay in telepresence systems and to propose techniques that compensate for such delays using new state estimation methods. The research plan included the following main topics: *1)* development of a framework representing a telepresence system using a commercial robot, motion tracking system (for robot navigation) and ROS, *2)* proposition of new state estimation technique for delay compensation using filtering methods, and *3)* verification of the new technique on a real like situation where camera-based visual SLAM was to be used as a tool to measure robot pose.

Details of Covid-19 related disruption

Covid-19 restriction impacted me on two accounts, *1)* unavailability of family support both during the birth of our first child and afterwards childcare for over a year and *2)* restriction to access the lab where I've developed an experimental set up for the visual SLAM (#3) (and conducted initial experiments).

Summary of the decision taken

Given the difficult situation, it was decided to work on a simulation environment to emulate the telepresence system, thus easing the issues with lab access restrictions and working from home. However, this introduced significant additional work as I needed to learn simulation software such as RViz, and Gazebo and their

interaction with ROS up to the level of developing a new working framework. The initial (now abandoned) work on visual SLAM is included in this thesis as part of future work.

Acknowledgements

I am sincerely grateful to my supervisor, Dr Gordon Dobie, for his guidance, patience and faith in me. I sincerely appreciate his excellent support throughout this journey. I wish to acknowledge the University of Strathclyde, Glasgow, UK and PPS UK Ltd. for providing me with a Faculty Innovation studentship, without which this research work could have never begun. Many thanks to William, Liam, Amir and others in the Center for Ultrasonic Engineering (CUE) lab for their technical help during the project.

This endeavour would not have been possible without the support of my husband, who generously provided invaluable knowledge and feedback. One of my most outstanding achievements during this PhD is the arrival of our daughter; without her continuous support, my journey would have ended a year earlier. I would also like to thank my parents and in-laws for their continuous support and encouragement throughout this PhD journey.

Contents

List of Figures

List of Tables

1 Introduction

Telerobotics is the area of robotics concerned with the control of robots from a distance. It is a combination of two major sub-fields, teleoperation, and telepresence. While teleoperation is the operation of a machine at a distance, telepresence refers to a set of technologies that allow a person to feel as if they were present, to give the appearance that they were present, or to have an effect, at a location other than their true location. The telepresence system allows a human operator to control and navigate a mobile robot around the remote environment and interact with their audiences through video conferencing [1]. For robot communication, depending on the distance, a wired (when everything is local) or a wireless (if the robot and the operator are miles away) connection is used.

In general, the telepresence system is composed of a local site (where a human operator drives a hand-controller device); a remote site (where a mobile robot interacts with the physical world); and a communication channel that links both sites. the telepresence system provides interactive two-way audio and video communication and physical manipulation with a remote sender and a receiver for building a communication system between two people in different places.

These systems, which are primarily used in the context of promoting social interaction between people, became popular in many emerging applications including hospitals and healthcare surgery or consultations, remote co-working in office spaces, tour guidance, security and surveillance (*e.g.,* remote night watch person [2], factory inspection, instructor-led educations and many more.

Telepresence robots suffer significant challenges during navigation in the remote site due to varying communication time delays [3] frequently caused by the present state of the network. Moreover, the distance between the human operator and remote sites of the telepresence system introduces time-varying delays adding distortion in the reference commands, response time, and feedback signals resulting in instability or poor performance of the system. The time elapsed between making an action decision and perceiving the consequences of that action in the environment introduces uncertain time delay [4, 5].

This uncertain time delay produces a visual mismatch between the received navigation state at the operator's side and the actual navigation state of the robot in the remote environment which negatively impacts the human operator's performance [6, 7]. Therefore, it is advantageous to compensate for such time delays for robust navigation and manipulation of a telepresence robot. This thesis focuses on proposing new techniques to compensate for time delays in the telepresence system for robust robot navigation.

There are many different approaches used in the literature to overcome the time delay in telepresence systems including increasing levels of automation, more sensors on the robot, and predictive technology. The predictive technology in this research includes a state estimation algorithm, display, and graphical models to predict the state of the robot based on the robot's delayed current state and commands sent by the operator [8].

The state of a robot is a set of position, orientation, and velocity, which is the robot's motion over time. This includes the estimation of the state of the robot's kinematic system by combining knowledge from a priori information and sensor measurements. State estimation in dynamical systems is crucial in real-world applications as the true state is unknown and sensors have limited precision, therefore, provide only a sequence of uncertain noisy measurements. Predictive display using control command and robot state estimation algorithm immediately display graphically the robot's estimated pose without time delay on the computer display.

The estimated pose is usually superposed on the display of delayed measurement from the actual robot measurement.

In this research, we aim to develop a state estimation framework in the real environment with a state-of-the-art commercially available telepresence robot. The framework is built with a differential drive robot using position-tracking sensors in the real environment. This framework is used to develop a new delay-compensated state estimation algorithm called Augmented State Extended Kalman Filter (AS-EKF).

We also recreate the real experimental framework and the environment using ROS RViz and Gazebo software. Accurate simulated models of the real experimental robot and the working environment with supporting elements are also designed in the simulation environment. The Gazebo can able to create a 3D scenario on the computer with robots, obstacles, and other objects, while ROS serves as the interface for the robot. RViz is also a powerful 3D robot visualization tool for ROS applications it provides a convenient GUI to visualize sensor data, robot models, and environment maps, which is useful for developing and debugging robot controllers [9].

Both, the simulated experimental results and the real experimental robot navigation agreed very well and showed robust robot navigation in a delayed environment using the proposed AS-EKF algorithm.

1.1 Problem statement

Seamless telepresence experience requires the implementation of human sensory elements such as vision, sound, and remote manipulation. Functional requirement blocks of an ideal telepresence system can be described as follows:

- The system usually includes **visual feedback**. Ideally, the entire field of view of the user is filled with a view of the remote location, and the viewpoint corresponds to the movement and orientation of the user.

- **Sound** is the easiest sensation which can be implemented with higher fidelity [10].
- The ability to **manipulate** at a remote area or environment is an important aspect for telepresence users and can be implemented in several ways depending on the needs of the user. Typically, movements of the user's hands or control commands are sent through the communication channel to the remote location. A telepresence robot in a remote location then follows those movements or commands as closely as possible.

However, the effectiveness of the telepresence system varies by the degree of fidelity, there are factors in telepresence which pose major challenges to precise and reliable robotic control for the human operator in an unknown environment. The following list identifies a set of research issues that need to be addressed for robust robot navigation:

1. **Time delay:** Telepresence is surprisingly difficult and slows to a human operator who, physically and directly performs a complex task far from the remote site. A particular difficulty is the delayed feedback response from the telepresence robot. Delays can be caused by distance, but other common causes include network switching delays, communication drop-out, processing delays, and slow dynamics of the slave telepresence robot.
2. **Limited field of view:** The human operator is unable to perceive the peripheral surroundings of the telepresence robot and must rely on his mental map and intuition for the surroundings to navigate the robot. Cameras don't provide the operator with the same degree of peripheral vision as the human eye. The operator's eye movements relating to attention or fatigue also deteriorate the performance.
3. **Problems with multiple cameras:** A problem was found when it comes to the use of multiple cameras and screens. To operate multiple cameras and

screens, the operator needs to switch his / her focus between them. Often these results add confusion instead of helping the operator [11].

4. **Depth perception:** Depth perception from a single camera is often limited and is not helpful for the human operator. This makes it harder for the operator to estimate distances. When operators navigate in a certain space they can rely upon stereo vision to help in judging distances. This is however not the only means by which we make estimations, we also use the size of reference objects and are helped by the parallax effect.

5. **Frame rate:** A low frame rate due to camera hardware or limited bandwidth can lead to a degraded sense of motion for the operator. This in turn leads to the operator using *a drive then stop* strategy for navigating in the remote environment. This is less accurate and sometimes error-prone.

6. **Other problems:** The telepresence system includes several sensors and actuators which pose measurement and hardware system errors.

Time delay produces a mismatch between the received navigation feedback at the operator's side (Local site) and the actual navigation state of the robot in the remote environment (remote site) which negatively impacts the human operator's performance. These two different scenarios create a conflict in human perception. Remote manipulation depends on the human operator's performance and is limited by the human's motor skills. Remote perception is very challenging to cope with the virtual display different from the physical environment. Teleoperation in a remote environment with time delay is very difficult and highly stressful for the human operator which leads to mental fatigue. Therefore there is a clear need to address such issues.

In this work, we are interested in the total perceived time delay of the telepresence robot manipulation which is the time from when the human operator sent a command until the visually perceives the reaction on the robot in the feedback information. An overview of the time delay within a telepresence system is shown in

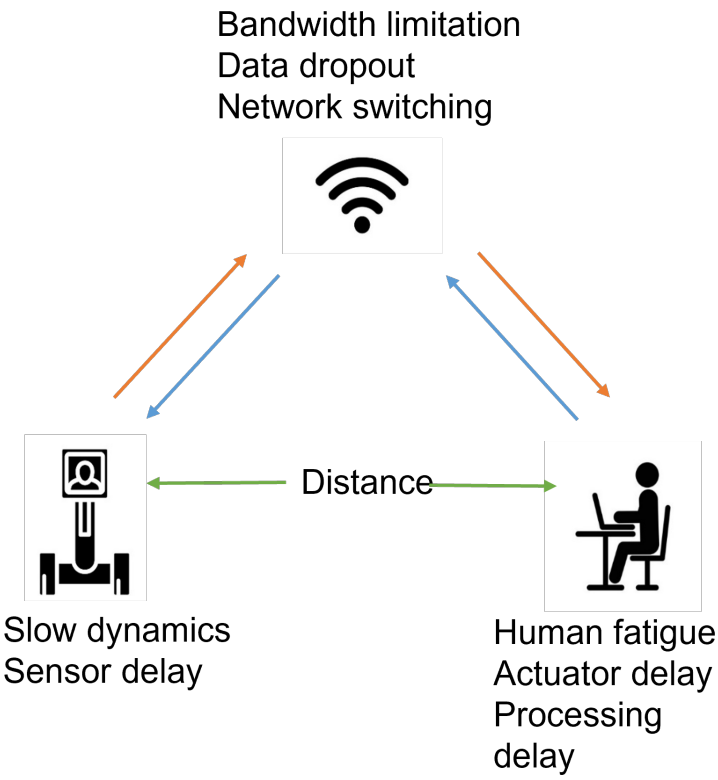


Figure 1.1: Potential causes of time delay in a telepresence system that needs to be addressed for robust robot navigation.

Figure 1.1. Our focus is on how to mitigate the challenges posed by time delays in robot navigation using new filtering techniques. It is worth noting that the causes of time delay, visual delay, and sound delay are not within the scope of this work.

1.2 Research contributions

Understanding of the time delay issues in the telepresence system is discussed in the form of a state-of-the-art report in Chapter 3. This summarises various time delays including in the communication channels and the impact of time delays in local and remote sites. Derived research questions are used as input for the following chapters. In this thesis following contributions are made:

- **C1.** A real-world experimental framework is proposed in Chapter 4. This consists of a state-of-the-art commercial telepresence robot, a VICON mo-

tion tracking multi-camera system, and methodologies for robot control and navigation using ROS and correction of systematic error using a standard UMBMark technique.

- **C2.** A new state estimation technique is proposed in Chapter 5 that considers time delay in telepresence robot navigation. The new algorithm incorporates augmented states to compensate for time delays which are modelled through two probabilistic density functions (PDF) of Gamma and Gaussian distributions. The new algorithm called Augmented State Extended Kalman Filter (AS-EKF) exhibits superior performance over standard EKF.

- **C3.** We have designed and developed a new simulation environment in Chapter 6 for telepresence systems that enable robot navigation and pose estimation using predictive technology. This also includes the predictive display which is necessary for human operators at the local site. Such a simulation framework allows for conducting controlled experiments instead of a real-world telepresence system which is expensive and not always practical to deploy for experimental purposes.

1.3 Publications

To date, the thesis has been produced following two conferences and one journal publication, along with another journal paper in preparation.

1.3.1 Published

- J1. Das, B. and Dobie, G., “Delay Compensated State Estimation for Telepresence Robot Navigation”, Robotics and Automation Systems, Journal, Elsevier, December 2021.

- C2. Das, B., Dobie, G. and Pierce, S. G., “A delay aware state estimation technique for telepresence robot navigation”, In proc. 3rd IEEE International Con-

ference on Robotic Computing (IRC 2019), IEEE, Mar 2019, Naples, Italy, pp. 624-629.

- C1. Das, B., Dobie, G. and Pierce, S., "State estimation of delays in telepresence robot navigation using Bayesian approaches", In proc. Towards Autonomous Robotic Systems: 19th Annual Conference (TAROS 2018), Bristol, UK, July 2018, Springer, pp. 476-478. (Nominated for best poster).

1.3.2 In preparation

- J2. Das, B. and Dobie, G., "Robot navigation and pose estimation using predictive technology", 2023.

1.4 Thesis outline

The thesis is organised into seven chapters. Chapter 1 provides the introduction and Chapter 2 presents a general overview of the telepresence systems, their application areas and the background that is necessary for this research. Chapter 3 reports the state-of-the-art analysis with a focus on time delay issues within telepresence systems.

Chapter 4 describes the real-world experimental framework, which consists of an off-the-shelf Beam plus telepresence robot and VICON motion tracking camera system followed by Chapter 5, which proposes a new approach for state estimation assuming uncertain delayed sensor measurements of a telepresence robot during navigation.

Chapter 6 presents the design and development of a simulation environment for robot navigation and pose estimation using predictive technology. Finally, Chapter 7 concluded the thesis and discussed future work where initial experiments are conducted by proposing a visual SLAM-based experimental set-up that allows robot pose estimation in unknown environments.

2 Overview and Background

In a wider understanding of the thesis, this chapter presents a general overview of the telepresence system and its application areas and the background information that is necessary for the proposed research. The chapter captures the wider context of teleoperation and its applications in a variety of domains. The selected examples are carefully chosen where time plays an important role, and therefore, issues around time delay impact the performance of such systems. This is a key motivation of this research, and further insight of the challenges are discussed with necessary details.

2.1 Telerobotics, Teleoperation and Telepresence

Telerobotics makes possible the execution of tasks that cannot be carried out directly by humans with their physical presence, *e.g.*, inability to reach the working area due to dangerous environmental conditions. Although robotics research incorporates new architectures for a variety of application needs, executing complex tasks are still challenging and robotics intelligence largely relies on the advances in computers [12]. Due to poor perception systems, robot intelligence is yet to be advanced enough to produce intelligent behaviour that deals with tasks in unstructured environments; presents uncertainties; and requires too much skill. Telerobotics is a combination of two major sub-fields, (a) teleoperation [13] and (b) telepresence [1, 14, 15].

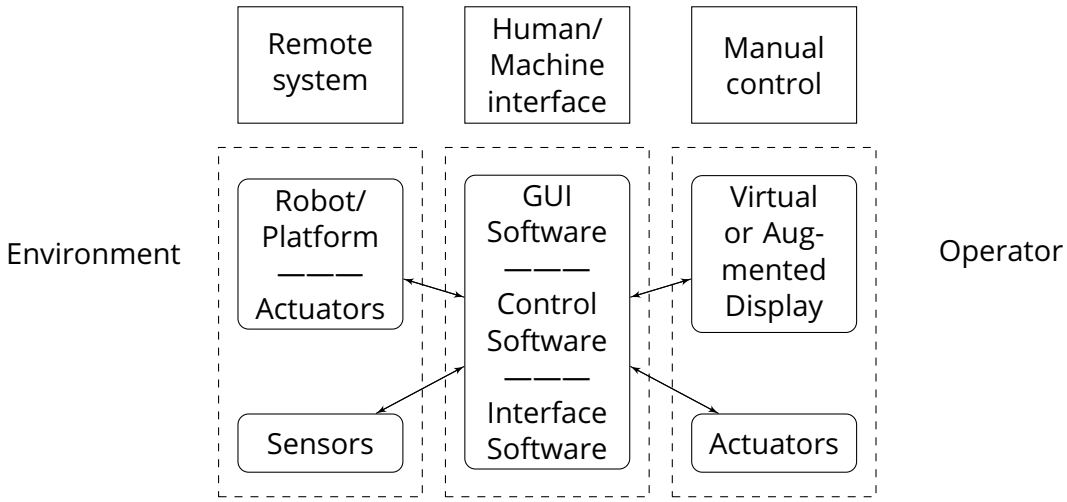


Figure 2.1: An overview of teleoperation technology which offers human operators to control the robot over the remote site using the sensors and actuators.

Teleoperated system offers human operators to control a mobile robot to complete tasks in remote environments. On the other hand, telepresence systems enabled human operators to feel present in a remote environment through the remote robot. The teleoperated system is composed of a local site (where a human operator drives a hand-controller device); a remote site (where a mobile robot interacts with the physical world); and a communication channel that links both sites. An overview of an example system is shown in Figure 2.1.

2.1.1 Teleoperated system

In a teleoperated system, a human operator controls the movements of the mobile robot from some distance away. The human operator sends signals to the mobile robot to control it then the feedback signals come back from the mobile robot, telling the human operator that the mobile robot has followed the instructions. A simplified architecture of the teleoperation system is shown in Figure 2.2.

It is also important to define two other terms of the teleoperated system, namely, *teleoperator* and *telerobot*.

- **Tele-Operator:** A teleoperator is a machine allowing a human operator to move about, sense, and mechanically manipulate objects at a distance. It

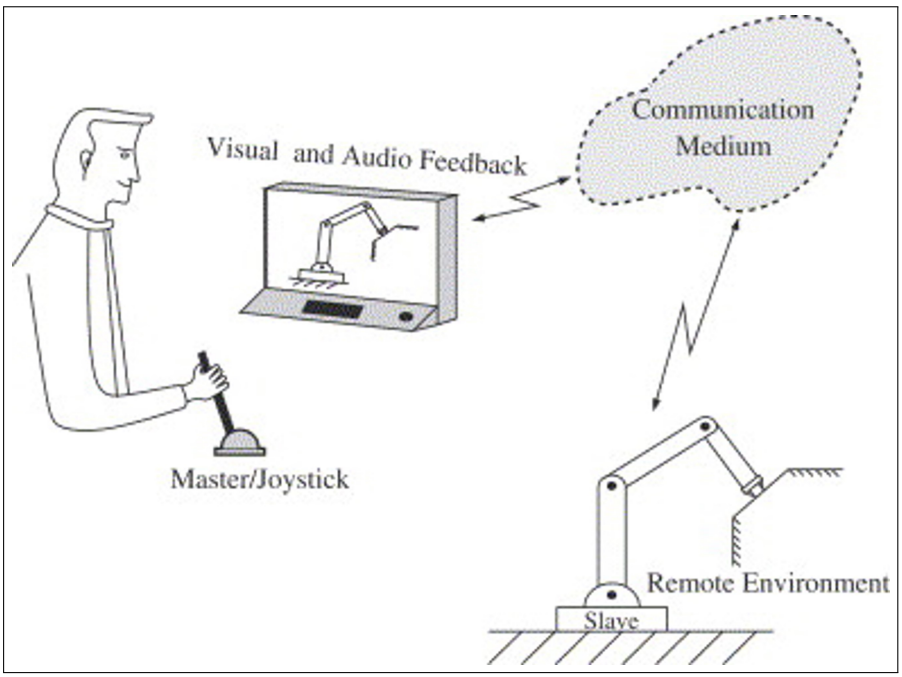


Figure 2.2: General architecture of teleoperated system where the human operator controls a mobile robot from some distance [13].

usually has artificial sensors and effectors for manipulation and mobility and also a means for humans to communicate with both.

- **Tele-Robot:** A telerobot is a subclass of teleoperators in which the machine acts as a robot but is monitored by a human supervisor and reprogrammed from time to time.

The history of modern teleoperation began at the end of the 1940s when the first master-slave manipulator was developed. After that, adapting video technology and force feedback to teleoperation made the development of telepresence systems rapid. Computer technology brought advanced control loops and virtual reality into the remote site in the telepresence systems [16].

2.1.2 Application areas of teleoperation

Teleoperated robots were used in a wide range of real-world applications including nuclear, space, underwater, military, medical, and other industries. This section discussed the most popular and important application areas of teleoperation.

Some of these applications are further grouped into sub-fields as they are diverse.

Figure 2.3: Example of a teleoperated robot for nuclear power plant application: iRobot Warrior [17].

2.1.2.1 Nuclear industry

Inspection and maintenance are essential in the nuclear industry. It is not easy to carry out such maintenance tasks since the environments are usually highly radioactive and unsafe for human workers. iRobot Warrior [18] (Figure 2.3) is suitable for indoor and outdoor use maintaining mobility on rough terrain in urban environments and all weather conditions. Warrior has been performing operations for 16 months in areas of the disabled power plant (Fukushima nuclear plant) where radiation levels and temperatures are too high and unsafe for people.

2.1.2.2 Space application

The space is a very challenging environment for teleoperation applications. The physical presence of a human to operate a vehicle in space requires many resources or is impossible. Therefore it is more efficient to use teleoperated vehicles.

(a) Lunokhod 2 [19]

(b) ETS-VII / Kiku-7 [20]

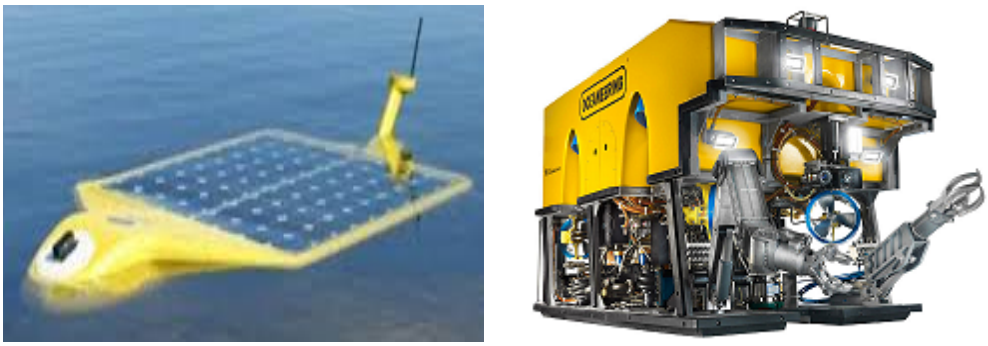
(c) Canadarm2 [21]

Figure 2.4: Examples of teleoperated robots for space applications.

Space exploration Lunokhod 2 (moonwalker) [19] (Figure 2.4a) was an unmanned lunar rover that landed on the Moon as a part of the Lunokhod program. The mission was to collect images of the lunar surface, examine ambient light levels, perform laser ranging experiments from Earth, observe solar X-rays, measure local magnetic fields, and study the soil mechanics of the lunar surface material. There was a five-man team of controllers on Earth who sent driving commands to the rover in real-time. Curiosity [22] is a car-sized robot rover exploring Gale Crater on Mars as a part of NASA. The rover's goals include investigation of the Martian climate and geology; assessment of whether the selected field site inside Gale Crater has ever offered environmental conditions favourable for microbial life, including investigation of the role of water; and planetary habitability studies in preparation for future human exploration.

Satellite ETS-VII (Engineering Test Satellite VII) / Kiku-7 [20] (Figure 2.4b) is launched by NASDA Japan, is the world's first satellite equipped with a robotic arm which was used to carry out several experiments related to rendezvous docking and space robotics.

Space station robotic arm The next generation robot arm, Canadarm2 [21] (Figure 2.4c), is an advanced robotic arm. This arm is capable of handling large payloads and helped build the entire orbiting complex. It has latches on either end, allowing it to be moved by both ground controllers and the expedition crews to various portions of the station. It has even been used to move astronauts around during spacewalks.



(a) Tavros02 [23] (b) Millennium Plus [24]

Figure 2.5: Example of teleoperated robots for underwater applications.

2.1.2.3 Underwater

Underwater operations were one of the first mobile applications where teleoperation technology was adopted. Today, remotely operated underwater vehicles (ROV) probably represent the largest commercial market for mobile vehicle teleoperation. ROVs are used in surveying, inspections, oceanography, and different simple manipulation and work tasks.

Tavros02 [25] (Figure 2.5a) is a Solar-powered Autonomous Underwater Vehicle (SAUV) to tweet water-quality data from its on-board sensors via long-range radio communications. Its Twitter feed is interspersed with periodic interjections

by a human operator. Every 20 minutes, Tavros02's followers get an update on its exact location and the temperature, saltiness, and pressure of the water around it. Another underwater teleoperated robot the Millennium Plus [24] (Figure 2.5b) is easily serviceable, accepts many tooling packages, and has simple survey integration capabilities. These characteristics accommodate heavier construction and completion work scopes underwater space.

(a) Atomics MQ-1 Predator [26]

(b) Gladiator [27]

(c) AN/SLQ-48 [28]

Figure 2.6: Examples of teleoperated robots for military combat applications.

2.1.2.4 Military applications

Autonomous robotics would save and preserve human life by removing serving soldiers who might otherwise be killed, while in service, from the battlefield. The use of autonomous or teleoperated fighters and bombers to destroy enemy targets is favourable because teleoperated planes are capable of performing manoeuvres which is dangerous for human pilots (due to high G-Force), plane designs do not require a life support system, and a loss of a plane does not mean a loss of

a pilot. Selected examples of this category are briefly described below.

Combat application The General Atomics's MQ-1 Predator [26] (Figure 2.6a) is an unmanned aerial vehicle (UAV) used by the United States Air Force (USAF) and Central Intelligence Agency (CIA). It is remotely operated by radio signals and satellite links. The Gladiator Tactical Unmanned Ground Vehicle(TUGV) [27] (Figure 2.6b) program was developed to support the United States Marine Corps conduct of Ship To Object Maneuver (STOM) missions through the use of a medium-sized, robotic system to minimize risks and eliminate threats to Marines during the conflict. The US Navy uses a remotely operated vehicle (ROV) called AN/SLQ-48 mine neutralisation vehicle (mnv) [28] (Figure 2.6c) system for mine combat. It is a remote-controlled submersible vehicle to identify underwater objects and, if they are mines, renders them safe.

(a) Dragon Runner [29]

(b) Packbot [30]

Figure 2.7: Examples of teleoperated robots for security, and rescue applications.

2.1.2.5 Security application

Dragon Runner [29] (Figure 2.7a) is designed for areas that are too dangerous for or inaccessible by human soldiers, particularly urban environments. Dragon Runner's front-mounted, tilting camera provides a video feed that is relayed back to its master controller by a wireless modem. It is used by many police department bomb squads to defuse or detonate explosives. It has also been used for improvised explosive device (IED) procedures on roadways via a remote teleoperated

control system.

2.1.2.6 Rescue application

PackBot [30] (Figure 2.7b) is a series of military robots by iRobot. More than 2000 were used in Iraq and Afghanistan. PackBots were the first robots to enter the damaged Fukushima nuclear plant after the 2011 Tohoku earthquake and tsunami.

2.1.2.7 Telesurgery

Robotic surgery requires the use of a surgical robot, which may or may not involve the direct role of a surgeon during the surgical procedure. The robot contains several sensors, which provide feedback data on the robot's current situation and a system to process this information so that the next action can be determined. One of the best-known surgical systems is the Da Vinci robot [31] (Figure 2.8) which can perform a variety of laparoscopic surgeries which involves scaling the surgeon's actions over a very small communication delay.

Figure 2.8: The example of a telepresence robot for surgical systems: Da Vinci Robot [31]

2.1.2.8 Industrial work

Industrial applications of teleoperated robots are numerous. They typically involve work conditions inappropriate for humans. The environment may be hazardous or unpleasant, or the required forces may be greater or smaller than the human can directly provide. A teleoperated robot tends to overcome weaknesses of human skills by sharing control as appropriate to the application. Examples of this group include applications in *highway*, *railway*, *power line maintenance*, *aircraft servicing*, and *mining*.

Highway ROEBL [32] (Figure 2.9a) a remote-operated, electric bucket loader skid steer solves two of the industry's major problems. For years, manufacturers have focused on driver safety. It reduces operator risk to zero by taking the driver off the vehicle. Additionally, the electric drive train solves another industry-wide problem — how to operate in close quarters where pollution can be dangerous.

Railway Mounted on the railway track, the Felix robot [33] (Figure 2.9b) is equipped with two 3-D profilometers and a GigE camera. Specific diagnostic algorithms have also been developed to process and analyse acquired data. Reflected laser light is captured by the camera. Once this information is computed, image and measurement data are then transmitted wireless to an operator for further analysis.

Power line maintenance Expliner [34] (Figure 2.9c) is a self-propelled robot that moves along overhead high-voltage transmission lines to perform an inspection of the lines by checking their external conditions, measuring the diameter and even detecting internal corrosion.

Aircraft servicing The ROBAIR aircraft inspection robot [35] (Figure 2.10a) is designed to climb on the wings and fuselage of aircraft to inspect rows of rivets and detect loose rivets and cracks. The robot uses a flexible array of pneumatic suction cups to adapt to surface curvatures to inspect rows of rivets with ultrasound,

(a) ROEBL [32]

(b) Felix robot [33]

(c) Expliner

Figure 2.9: Teleoperated robots for highway, railway and power line applications.

eddy current, and thermographic Non-Destructive Testing (NDT) techniques.

Mining applications Hitachi EX5600 Hydraulic Excavator [36] (Figure 2.10b) and a CAT 793D 240-ton haul truck [37] (Figure 2.10c) are used to remove the debris and repair the mine area. The equipment can rapidly move between manual and teleoperated control with the flip of a switch.

2.1.3 Telepresence system: two-way communication system

The research carried out in this work considers the application area of two-way telepresence systems and uses a state-of-the-art off-the-shelve telepresence robot for both the formulation of the research questions, development of the algorithms and testing. While further details of the specific system/environment are discussed in Chapter 4, this section provides necessary background information relating to this research.

(a) ROBAIR

(b) Hitachi

(c) Haul truck

Figure 2.10: Teleoperated robots for aircraft servicing and mining applications.

Telepresence systems are integrated with monitor, camera, microphone, and speaker systems [38] that allows real-time two-way collaboration between people who are not in the exact location. They can speak as if they are in the same room and easily share data. A telepresence robot is an integral part of a telepresence system and the key component that helps to place a human operator at a remote location instantly, providing them with a virtual presence, or *telepresence*. A telepresence robot is a computer, tablet, or smartphone-controlled robot which includes a video camera, screen, speakers, and microphones so that people interacting with the robot can view and hear its operator and the operator can simultaneously view what the robot is *looking* at and *hearing* [38]. The units can be moved around by a user who is not present at the robot site. The system design and functionality depend directly on their intended use and application. A brief description of hardware and software specifications of the most common telep-

resence systems can be found in [14]. A few telepresence robots are available on the commercial market (Figure 2.11) like Giraff [39], Pebbles [40] and Double [2] etc.

Figure 2.11: Example of different types of telepresence robots are available on the commercial market depending on the application areas [41]

A simplified architecture of a telepresence system with a telepresence robot is shown in Figure 2.12. As perceived, the system can be considered as a combination of three subsystems, namely, local site, remote site, and communication medium/system. As shown in Figure 2.12, human operators use computers, tablets, or smartphones to operate mobile telepresence robots at the remote site. The communication medium connects the robot and the host computer, often through a standard WiFi network using TCP/IP communication protocol.

2.1.4 Application areas of telepresence

Telepresence system in the context of two-way communication means audio and video communication with a remote sender and a receiver for building a communication system between two people in different places. These systems, which are primarily used in the context of promoting social interaction between people, are becoming increasingly popular within certain application domains such as health care environments, educational centres, independent living for the elderly, and

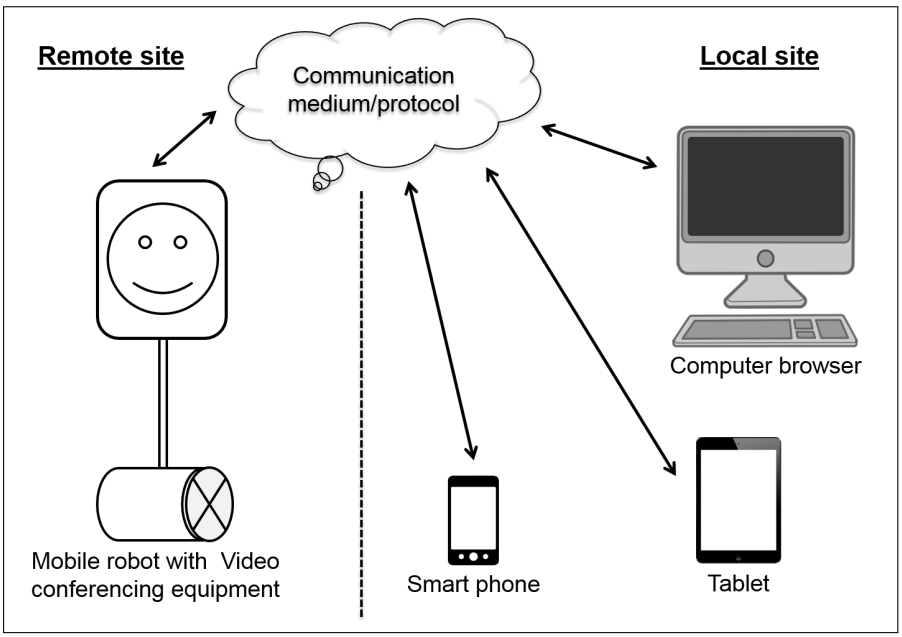


Figure 2.12: General architecture of telepresence system where, human operators use a computer, tablet, or smartphone to operate mobile telepresence robots at the remote site.

office environments [15]. The systems are characterized by an LCD screen, a web camera, a microphone, and speakers allowing communication between two parties.

2.1.4.1 Office environment

The number of tasks where many different collaborative teams are cooperating from a distance in the same work is increasing. One can simply log in to their control device and instantly virtually present in their remote location. A set of commercial telepresence robots has been offered in this environment to reduce the distance between different teams from different places, which decreases the travelling time and travel cost for the employees and also allows immediate access to the other site if needed. The telepresence system allows remote coworkers to visit their local coworkers and attend formal as well as informal meetings. Telepresence systems became very useful to the work-life balance and increase productivity.

Figure 2.13: The telepresence system allows remote coworkers to visit their local coworkers [42] [1].

2.1.4.2 Healthcare system

Telepresence systems seemingly revolutionised (or at the least significantly improved) the healthcare system where doctors, nurses, and other healthcare professionals work. Patients can communicate easily with their doctors via telerounds every day in the absence of their physical presence, which greatly reduces the location-centric (usually large cities) dependency and provide greater care to remote care facilities.

Post surgery: Telepresence systems have tremendously helped in patient monitoring, integration services, and private healthcare after patients are discharged from hospitals. This also keeps sick people out of waiting rooms.

Advice: Medical specialists can communicate and administer drug prescriptions to patients through telepresence systems. Doctors also administer treatment to remote patients through telepresence systems with the presence of a nurse at the patient's location. Medical professionals can expand their network of services through telepresence systems to far unreached locations with patients in need of quality medical care.

Care in home: Telepresence systems can be used to monitor patients at home.

Figure 2.14: Doctors administer treatment to remote patients through telepresence systems [43].

Instead of taking children to the hospitals for regular checkups, they can be monitored via video consultations by physicians. Telepresence systems can facilitate long-term health monitoring of the elderly in private homes.

2.1.4.3 Elderly care

Telepresence systems for elderly care serve various functions simultaneously, such as health surveillance, social interaction, and safeguarding. Some users rely on their telepresence robots as their remote eyes and ears for elderly family members. People can use a telepresence system for regular chats with seniors instead of using the phone or video conferencing. It could work as an emergency alert system if a person falls in an emergency situation.

2.1.4.4 Education

There are lots of applications for telepresence robots in education systems. It helps in primary education, higher education, home-bound instruction for ill and hospitalized students, and more. In many circumstances, the telepresence system in the education system increases students' engagement and connection to

Figure 2.15: Telepresence system for regular chats with seniors [44].

the course content. It can be used in regular classrooms and experience no disruption to normal classroom activities. For home-bound students, extended school absences can leave students feeling lonely and isolated. The telepresence systems help ill or hospitalized children stay connected to their school community.

2.1.4.5 General use

Telepresence systems are becoming more sophisticated in recent days. It is no wonder that telepresence systems are becoming a popular option for researchers to make it more robust by reducing its complexities for the human operator and useful for social interactions. With the COVID-19 pandemic, teaching and learning online (*e.g.,* Skype or Zoom) became the new norm. By integrating telepresence systems in our societies and improving the understanding of how human operators use robots, it is possible to improve the design of the systems more advanced that will attract people to use their day-by-day activities.

Figure 2.16: Telepresence system in education system increases students' engagement [45].

2.2 Research background

The research background consists of three main sections, namely *1)* the robot navigation challenges, *2)* a proposed approach with an experimental framework that will address the navigation challenges and *3)* a robot simulator which will validate the proposed approach.

2.2.1 Navigation challenges

Remote operation, in particular, navigation of the telepresence robots, depends on the human operator's performance and is limited by the human's motor skills. It largely relies on the perception of the remote environment. However, remote perception is widely considered challenging to cope with, especially with the virtual display, which does not offer the same interaction opportunity as the physical environment.

A particular difficulty is the delayed response from the robot. While delays can be caused by distance, other common factors include network switching delays, communication drop-out, processing delays (both in the robot and local host computer), and slow dynamics of the telepresence robot.

This research is interested in investigating and developing techniques that can

address the navigation challenges caused by the aforementioned time delays in a telepresence system. As time delays can be complex and difficult to measure separately and accurately, we rely on estimations of such collective delays through statistical distributions and propose generic solutions such as the augmented state Kalman filter. However, reproducing the real-life scenarios, we make use of a real telepresence robot for the experiments. In providing additional flexibility, this research also considers the development of simulation environments in the form of predictive displays.

2.2.2 Proposed approach

There are many different approaches used in the literature to overcome the time delay in telepresence systems. We aimed to use predictive technology to estimate the robot's state from the delayed measurement and compensate for the time delay using a state estimation algorithm. The predictive technology in this research includes a state estimation algorithm, display, and graphical models to predict the state of the robot based on the robot's delayed current state and commands sent by the operator.

In order to achieve that, we have divided the work into three distinct parts, namely, the preparation of a framework, the development of state estimation algorithms, and the development of a simulated environment for predictive display. The overall structure is shown in Figure 2.17, which also indicates various key components and their interactions. Brief background of these parts and their components are described below.

2.2.2.1 Proposed experimental framework

As a starting point of this work, we developed a new real-environment experimental framework that comprises of Robot Operating System (ROS), an industrial telepresence robot (Beam plus [17]), preparation of software that provides a control for operation and interaction between ROS and Beam plus, a motion tracking sys-

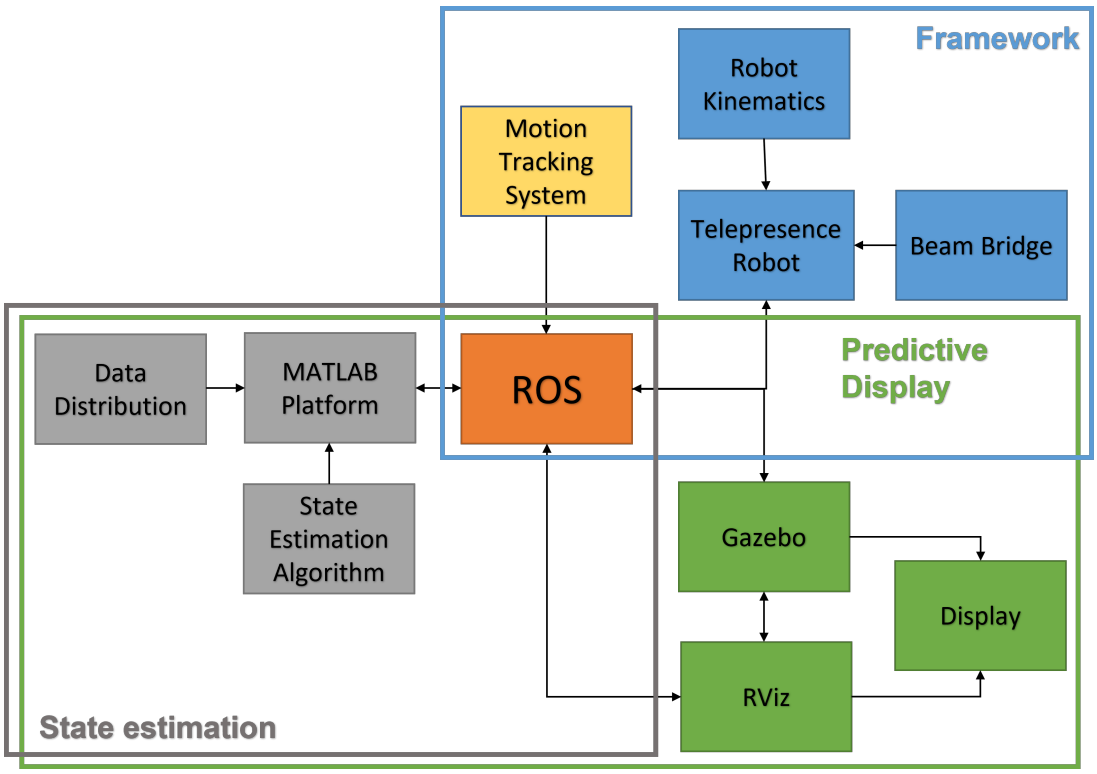


Figure 2.17: Background structure of the proposed research. They are grouped into three main areas: a) framework, b) state estimation and c) predictive display.

tems for data measurement/collection, and related preparation to address issues related to robot kinematics. We briefly described the backgrounds of the key components in this section with a detailed description of the framework in Chapter 4.

Telepresence robot The industrial telepresence robot Beam plus is used for designing and validating our hypothesis through controlled navigation, collection of real-time sensor measurement data in the remote environment, and estimating the robot's true state. The Beam plus is the latest telepresence robot offering from Suitable Technologies. Designed as a smaller home use which allows operators to visit with one another no matter where in the world they may be located. It can be operated using a computer, tablet, or smartphone after being connected to a WiFi source and downloading the Beam app. More details can be found in the Section 4.3.1.

ROS framework The framework was built using the Robot Operating System (ROS) [46], an open-source framework that helps researchers and developers build and reuse code between robotics applications. In this research, ROS plays a key role in the navigation and control of both the telepresence robot and the predictive display in the simulation environment. We have used ROS Kinetic Kame distribution [47] with Ubuntu 16.04 LTS operating system.

ROS driver (*beambridge*) While ROS is widely used for robot navigation and control, one of the key difficulties in this work is to interact with the commercial Beam plus robot used in this work. Although Beam plus was built using ROS at its core, it was not as accessible as some other robots built for research purposes. Our choice of the robot was motivated by the fact that it would resemble the real scenarios (*e.g.,* remote industrial robots in office/educational institute/hospital or care home environments where delays are common due to network connectivity, physical distances etc.), which however proved to be a significant challenge to use for controlled experiments.

In order to overcome this, a ROS driver was in need which is essentially a ROS node that makes a piece of hardware accessible from ROS. In this work, we adopted an existing ROS driver (*rosbeam* [48]). Changes are made to accommodate our model of Beam plus, which is different from the models used in the *rosbeam*. Due to the unavailability of relevant technical details, as with any commercial products, we performed an empirical study of its navigation and related hardware parameters and went through a series of trials in developing the desired customised ROS driver (named as *beambridge* in this work). *beambridge* is an essential component of this framework that helps to communicate with the telepresence robot from the host computer through standard WiFi communication.

Motion tracking system Motion tracking assists in tracking the movement of robots and transferring the sensed data to an application for further processing. In this research, we used VICON motion capture systems which use passive optical

motion capture technology. This technique uses retro-reflective markers on the robot body that are tracked by infrared cameras [49].

In our experiments, the retro-reflective markers attached to the robot reflected light generated from near the VICON camera lenses. Once reflected, the light was used to calculate the position of the markers within a three-dimensional space and transferred to the host computer for further processing. The position information (of the robot) is used as the measurement data in the remote environment.

State estimation algorithm In addressing the time delay issues in telepresence systems, this research relies on prototypical Extended Kalman Filter (EKF) [50], and a new algorithm called Augmented State EKF (AS-EKF). The core idea is to consider the time delays as a collection of small delays at which the robot states are estimated in the form of augmented states which, in effect helps to estimate the true position of the telepresence robot.

One of the key activities within the state estimation is delay modelling as delay may be caused by a number of factors as indicated earlier (Section 2.2.1). While some delays are consistent, our assumption is not all the delays are precise and hence uncertain. Therefore, it is important to model both certain and uncertain delays to obtain a consistent state estimator. The uncertain delays are modelled here by probabilistic density functions (PDF), such as *Gamma* distribution [51–53] and *Gaussian* distribution [54, 55].

This research proposes a new approach for state estimation assuming both certain and uncertain delayed sensor measurements of the telepresence robot during navigation. Further details of the key theories and the proposed algorithm can be found in Chapter 5.

It is worth noting that the state estimation algorithm was developed in MATLAB, while the Beam plus robot navigation was done using C++ programming language within the ROS environment. In the context of the overall system, it can be assumed that MATLAB is installed in the host computer for state estimation purposes and connected with both real telepresence robots through ROS control.

2.2.3 Robot simulator

Robotics simulation plays an important role in the design, development, verification, and validation of robotic systems [56]. Recent studies have shown that simulation may be used as a cheaper, safer, and more reliable alternative to the manual and widely used process of field testing. Robotic simulators are invaluable tools that allow developers to rapidly and inexpensively design, prototype, and test robots in a controlled environment without the need for physical hardware.

In the case of the telepresence system, it is even more prevalent as controlled experimental arrangements are complex, impractical in many cases, expensive, and not always necessary. For example, setting up a framework similar to ours (described in Section 2.2.2.1 and Chapter 4) requires larger experimental facilities along with an expensive telepresence robot and motion tracking system. Even then, experimentation with various delays is not plausible. Therefore, a simulator is necessary and beneficial for the research community as well as practitioners. In this work, we developed a simulator for such purposes, and the core components are briefly described below with further details in Chapter 6.

In this research, we used the Gazebo simulator and RViz to develop the simulation version of the real framework. As indicated earlier, we aim to develop a predictive display of robot pose, which helps to model delays and evaluate algorithmic performance in the simulation environment.

2.2.3.1 Gazebo

Gazebo [57] is one of the most frequently used simulators used along with the ROS framework. Although they are separate projects, the ROS official repository maintains a library *ros-indigo-gazebo-ros* that helps to communicate with Gazebo. It contains plugins that interface ROS with Gazebo. These plugins can be attached to objects in the simulator scene and provide easy ROS communication methods, such as topics - both published and subscribed by Gazebo and services. Packaging Gazebo as a ROS node also allows for it to be easily integrated into ROS default

method for running large and complex systems, called a launchfile [58].

2.2.3.2 RViz

RViz [59] is the *de facto* 3D visualization tool for ROS applications. It provides a view of robot models, captures information/measurements from robot sensors, and replays the captured data. It can display data from cameras, lasers, from 3D and 2D devices, including pictures and point clouds.

The difference between the two can be summed up in the following excerpt from Morgan Quigley (one of the original developers of ROS) in his book Programming Robots with ROS: "RViz shows you what the robot thinks is happening, while Gazebo shows you what is really happening".

The final necessary component of our simulation system is the display screens. It is anticipated there will be two display screens in front of the human operator. The first screen is for audio and video communication, while the second one is for robot navigation.

2.3 Chapter summary

This chapter provides an overview of teleoperated and telepresence systems, their usage scenarios, and brief background of our proposed research. Teleoperations are not a new concept and are widely used in many application areas including in nuclear industry, space applications, underwater, military applications, telesurgery, and many other industrial works. Our focus in this work is on telepresence systems which are often used for remote presence with two-way communication and a subject of emerging interest in social communication, office, hospital, care homes, schools, and other environments.

The issues relating the communication and other delays can severely affect telerobot navigation, and in this work, we proposed techniques that can address such issues. Therefore, it is important to have the necessary background, which was described in Section 2.2 followed by a state-of-the-art analysis in Chapter 3.

3 State-of-the-art analysis

3.1 Introduction

According to the levels of its autonomy, robots have been classified into autonomous robots and teleoperated robots. Equally, robots can be distinguished depending on the levels of human interference required to control robots. An autonomous robot has a high autonomy level and low human involvement level whereas a teleoperated robot has a low autonomy level and high human involvement level.

While a teleoperated robot is developed for manipulating operations at a distance, a telepresence robot is developed for social communication at a distance. Generally, a teleoperated robot helps extend a person's mechanical action beyond his reach. On the other hand, a telepresence robot provides interactive two-way audio and video communication with a remote sender and a receiver for promoting social communication between the two people.

In telepresence systems, the person being in the remote site is able to engage with local participants as if they are physically located at the point of the device. Since the point-of-view of both parties join on the same device, better engagement results. And since the device can move around, the remote people are able to move themselves to better locations or situations, creating a much better engagement with other locals outside the teleconference space. In the simplest terms, the fact that you can look or even move around the room increases the feeling of "being there".

Depending on the requirement, a telepresence robot should have the following capabilities,

1. Can be controlled remotely by the human operator.
2. Can be navigating smoothly in the environment.
3. Can transmit video and image files within the minimum time span.
4. Video conferencing.

To manoeuvre a telepresence robot efficiently fulfilling all the capabilities, we need to address all the challenges produces during remote telemanipulation. We have divided all the basic parts/requirements of a telepresence system into three categories (as shown in Figure 3.1), namely, *1) remote site, 2) communication channel* and *3) local site*; and described the associated challenges in each category. This is followed by the methods or algorithms researchers proposed to address such challenges.

3.2 Robot navigation at local site

Based on the available literature, we grouped the robot navigation at the local site into four categories, *1) virtual world, 2) computational infrastructure, 3) control system design* and *4) actuators*. A computation infrastructure enables the virtual world display and to operate the virtual world we need a control system including actuators to handle by the human operator.

3.2.0.1 Virtual world

In a telepresence system, the human operator uses real images and videos which are transmitted from the mobile robot and sends commands to the robot through the network for smooth robot navigation. However, the operator cannot execute a smooth operation in this cycle, because there are large time delays and the delays which depend on the state of the network are not constant.

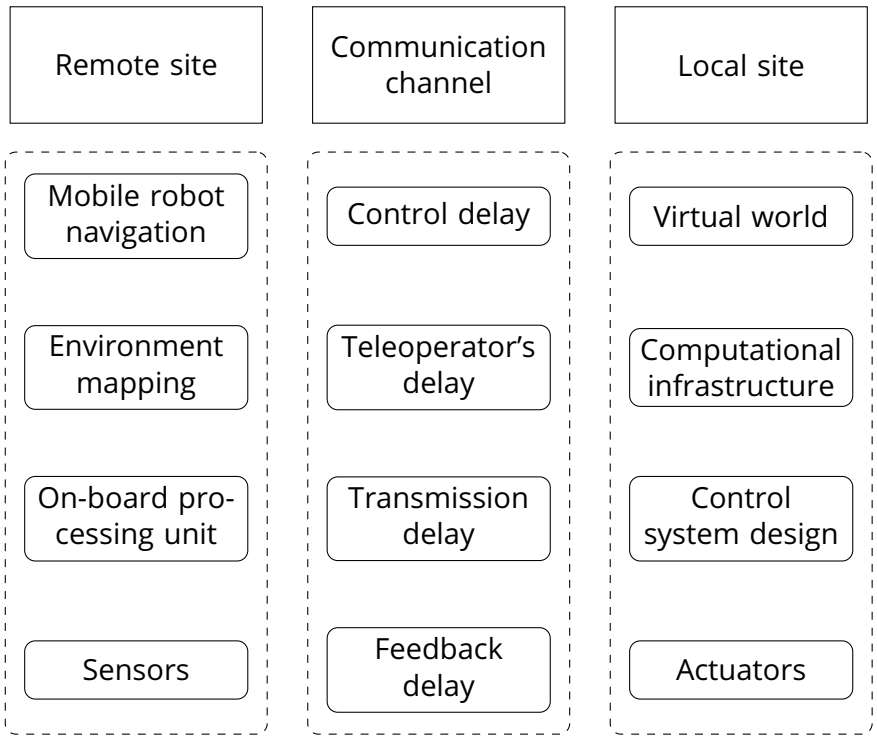


Figure 3.1: State-of-the-art dissection of telepresence system, largely categorised in a) remote site, b) communication channel and c) local site.

In their research Kawabata *et al.* [60, 61] have discussed the time delay problems in telepresence systems. They proposed a framework of human interface systems for the teleoperation of a mobile robot. The prototype of the telepresence system was constructed utilizing the virtual world as an operator interface. A telepresence system represents the use of virtual reality to virtually move the user to another location. It has the ability to interact with a real and remote environment from the user's perspective. The commands from the human operator could directly communicate with the virtual robot in the virtual world. Then the virtual images could be also communicated to the operator with little time delay. Therefore, the operator does not have to feel the delay in communicating the information and operates the robot easily.

Moreover, by using the image taken from the virtual robot moving in the virtual world, the intervals between the image data from the real world can be filled in by those virtual images. These images give the operator a realistic and clear visual

display for comfortable navigation of the mobile robots. Also, by using the virtual world the human operator can watch the object from various points of view. A human operator could simultaneously operate a robot in the real world as well as in the virtual world which could compensate for the incomplete data transfer to the real robot [62, 63].

Similarly, Sanguino *et al.* [64, 65] presented a mixed-reality interface approach to assist people in navigation tasks with teleoperated robots. The system combines visual stimuli generated from real and virtual sources to provide a fused interface. They developed Mixed-perspective Exocentric Display (ME3D) system (using augmented display) from a user standpoint for better orientation, more accurate manoeuvres and less workload.

A virtual impedance method (a virtual force) that enables real-time planning to follow the generated trajectory, while avoiding obstacles in the teleoperation, is proposed by Jin *et al.* [66, 67] in order to overcome the issues of limited communication bandwidth and narrow view angle cameras. In the virtual impedance method, the virtual force is generated according to both the distance between the obstacle and the robot and the approaching velocity of the obstacle. This virtual force is reflected in the operator who is holding the master which converts the force data transferred over the internet to a physical action.

3.2.0.2 Computational infrastructure

Multiple researchers [68–70] induce the use of *Cloud Computing* for computation offloading the navigation assistance of a service mobile robot. The information is processed in a private cloud platform.

Colledanchise *et al.* [71] and Santos *et al.* [72] have shown how to use event-based sampling to reduce the number of measurements done, thereby saving time, computational resources and power, without jeopardizing critical system properties such as safety and goal convergence. The results are particularly useful for real-time systems such as high-speed vehicles or teleoperated robots, where

the cost of taking measurements are higher, when considering the number of measurements for stops or transmission times.

3.2.0.3 Control system design

In various application environments, the use of telepresence robots can help to fulfil important tasks in distant areas that human operators are unable to present physically. In the remote site, the mobile robot, operated by a human operator from the local site can have a better view of its direct environment and a higher chance of detecting possible actions. If the human operator wants to control the robot, let it try to protect itself against collisions and rollover. Several control algorithms [73, 74] have been developed to address these control problems.

According to Siciliano and Khatib [75, 76], the robot usually has a layered architecture. Layers in the top level of the hierarchy can contain processes that, for example, perform cognitive tasks similar to humans. In the middle layers, tasks also involve complex processes like path planning, object handling, speech recognition, etc. Finally, in the lowest levels, reactive and real-time control operations are performed (*e.g.,* obstacle avoidance, guidance, beacon detection, signal communications processing, *etc.*). The amount of computation is not necessarily proportional to the level. Operators can use different teleoperation modes to control a telepresence robot, which can be pure or assisted teleoperation or with full autonomy.

Armbrust *et al.* [77, 78] described a concept of integration of different control modes with different levels of operator influence within one robot control system. It described how an operator can easily choose the control mode (*e.g.,* pure teleoperation, assisted teleoperation, and fully autonomous navigation) fitting best to the current situation. Using behaviour fusion, the control system allows an operator to continuously increase or decrease his influence on the robot's motion.

3.2.0.4 Actuators

Telepresence is a technology that allows a person to observe and interact with a remote environment. A computer is used to display a live video stream from a remote location, allowing a user to examine the surroundings. The telepresence system requires new ways to sense the various motions of the mobile robot and the operator's commands. This means new motors, new sensors and new actuators. Commercial telepresence solutions lack a physical actuator, which decreases the usefulness of the robot to general consumers.

Both Park *et al.* [79] and Wildenbeest *et al.* [80] proposed navigation methods for a teleoperated mobile robot moving in an unstructured environment utilizing a force feedback joystick to manipulate the unpredictability. Similarly, J. Du *et al.* [81] and Hunag [82] have developed teleoperated robotic 3-D mapping (TeRoM) systems consisting of a pan-tilt unit for the RGB-D camera and a joystick for controlling. David *et al.* [83] and Sanders *et al.* [84] investigated the improvements of the interactions using new algorithms for mixing data from ultrasonic sensor systems with joysticks controlling telepresence mobile robots.

3.3 Role of the communication channel

The major challenges often faced by teleoperated robots include latency of the system, the lag in the response to movements, *i.e.,* the mechanical and computer processing of the movement and response; lag in the visual representation and inadequate resolution of video data for seamless real-time communication.

3.3.0.1 Time delay in telepresence systems

A mismatch between user motion and robot movement or inadequate system control for small movements attribute to the previously mentioned challenges in Section 1.1. Distance between the human operator and remote sites of a telepresence system generally introduces time-varying delays adding distortion in the reference

commands and feedback signals resulting in instability or poor performance of the system. On contrary, the time elapsed between making an action decision and perceiving the consequences of that action in the environment introduces control delay.

An important feature of internet-based teleoperation is the possibility of data packet drop-outs. When congestion occurs in the network and some packets are lost, it may be advantageous to forget the old packet and transmit a new one which contains recent information. However, such incidents cause a time delay and impact robot navigation. If a significant amount of data is dropped, it may result in discontinuity of the reference trajectories and the forces transmitted between the master and the slave.

In a telepresence system, an operator's view is limited to images provided by one or more cameras mounted on the remote vehicle. Due to such arrangements, teleoperation generally requires a real-time stream of images transmitted from the remote operator to the control station. For this, the transmission link between the vehicle and operator must be very high bandwidth and very low latency.

3.3.0.2 Robot navigation in time-delayed environments

Mora *et al.* [85] presented a novel method to deal with the time delay and narrow bandwidth limitations inherent to rescue and search teleoperated mobile robotic systems. This method combines the visualization of two models of the teleoperated robot inside a 3D virtual environment. One model represents the position and orientation provided by real-time GPS located on the teleoperated robot and another model is based on inputs given by the human operator through the information gathered by the laser range finder sensor. The assisted teleoperated navigation system helps the human operator understand the localization of the teleoperated robot smoothly between every new sensory information data that reaches the human operator side.

In another approach, Hu *et al.* [86] introduced a 3D model-based predictive dis-

play system where the operator sees the predicted image instead of delayed video to control a robot remotely. A scene model was constructed online, representing the remote robot's working environment, which is not only used to generate a predicted image but can also supply a global or arbitrary rendering view. Tracking and constructing 3D geometry model are both running on-board to save data transmission bandwidth and allows for on-board autonomous tasks such as local path planning and obstacle avoidance.

As discussed in Section 3.2.0.1 Jin *et al.* [66] proposed a virtual impedance method that enables real-time planning to follow the generated trajectory in a limited bandwidth communication network, which is also relevant to this section.

3.4 Robot navigation at remote site

In remote environments the mobile robot is equipped with different types of sensors and an onboard processing unit, roaming around to accomplish the predefined task. The remote environment can be mapped using different types of Simultaneous Localization And Mapping (SLAM) algorithms, before navigating the robot in an unknown environment.

3.4.0.1 Mobile robot navigation

Telepresence robots require moving in an unstructured environment that includes unknown obstacles and uneven terrain. Thus, the ability to navigate without collision with obstacles and rollover in uneven terrain is a crucial issue for telepresence robots. Although the robot is manoeuvred by an operator at a remote site, it should control itself autonomously for avoiding collision with obstacles and for preventing rollover when it detects possible collision or rollover.

Multiple researchers [79, 87, 88] introduced navigation methods for teleoperated mobile robots moving in unstructured environments based on guided navigation algorithms and roll-over prevention algorithms. It reacts autonomously for avoiding collision with obstacles and for rollover when it detects possible collision

or rollover. Time taken to complete a teleoperated task with a mobile robot partly depends on how a human operator interacts with the mobile robot.

Kim *et al.* [89] suggested a hybrid autonomous/teleoperated strategy for reliable navigation with a framework of extended Kalman filter to localize the mobile robot. When the robot faces unexpected obstacles or a situation where a collision is expected, it sends a warning message and changes the teleoperation mode to autonomous mode to avoid obstacles. After avoiding the obstacles, it returns to teleoperation mode and the teleoperator has control again.

While currently available commercial telepresence robots lack autonomy, Alers *et al.* [90] believe that integrating more autonomy with recent AI research in a single framework can greatly increase the usability of these robots. They introduced three use cases: a) a robot that is remotely controlled by the client with GUI assisted by both visual augmentation and the robot itself; b) a scheduled meeting and c) a whiteboard meeting with multiple actors and points of interest. The authors implemented low-level autonomy on the robot in the form of assisted teleoperation.

Kuderer *et al.* [91] have presented an approach that allows a mobile robot to learn how to navigate in the presence of humans while it is being teleoperated in its designated environment [92, 93]. The method applies feature-based maximum entropy learning to derive a navigation policy from interactions with humans. The resulting policy maintains a probability distribution over the trajectories of all the agents that allow the robot to cooperatively avoid collisions with humans. In particular, this method reasons about multiple homotopic classes of the agent's trajectories, *i.e.,* on which sides the agents pass each other. They implemented the approach on a real mobile robot and demonstrate that it is able to successfully navigate in an office environment in the presence of humans relying only on onboard sensors.

3.4.0.2 Environment mapping

Teleoperation is a difficult task, especially when controlling robots from a remote operator station. This is particularly the case when the target area is unknown to a human operator, demanding control of the robots from a remote place. In general, the operator has to solve the problems of mission planning, target identification, robot navigation, and robot control, at the same time in an unknown environment. For untrained operators, control and target identification are already challenging on their own.

It can be difficult for a robot to recognize obstacles and other hazards in an unknown environment. Ollis *et al.* [94] described an approach to calculating terrain costs from Bayesian estimates using features vectors measured during a short teleoperated training run in similar terrain and conditions. The robot can learn to estimate the probability that any feature vector corresponds to traversable terrain.

Kleiner and Dornhege [95] presented a novel scan matching technique that re-evaluates data memories during the search, allowing robust pose estimation under varying roll and pitch angles of the robot enabling mapping on rough terrains. Experiments within different environments showed that the system produces comparably accurate maps in real time.

In their research Kubota *et al.* [96] discussed the monitoring system of a teleoperated robot and human interface based on visual information and distance information from the teleoperated mobile robot built by the SLAM. They developed navigation systems using map information. They used three different methods of (1) simple gesture navigation, (2) trajectory gesture navigation and (3) pointing navigation.

Borenstein *et al.* [97] introduced the TelOpTrak algorithm with heuristics-enhanced dead-reckoning for precision indoor tracking of teleoperated robots. TelOpTrak does not rely on GPS or external references; it uses odometry, a low-cost MEMS-based gyroscope, and heuristic assumptions about the structured nature of most indoor environments. TelOpTrak significantly reduces position errors, as accumu-

lated heading errors are almost always the the primary source of position errors in a dead-reckoning system.

J. Du *et al.* [81] have developed a teleoperated robotic 3-D mapping (TeRoM) system which enables efficient human-guided mapping of remote environments for realistic rendering and visualization. The system consists of a pan-tilt unit for the RGB-D camera and a joystick for control. It can generate 3-D maps of indoor environments through point cloud-based mapping, triangulation and mesh optimization.

Hu *et al.* [86] focused on a 3D model-based predictive display system where the operator sees the predicted image instead of delayed video to control a robot remotely. A scene model was constructed online, representing the remote robot's working environment, which is not only used to generate a predicted image but can also supply a global or arbitrary rendering view.

3.4.0.3 On-board processing unit

Sun *et al.* [98] introduced a detecting robot which was been designed based on the advanced 'System Of Programmable Chip' embedded technology [99–101]. One of the Field Programmable Gate Array (FPGA) based processing boards (by Altera, now Intel) was used as the robot's controller.

The information about the robot site was transmitted to the local site by a wireless transmission module. The operator can remotely control the robot and the manipulator on the PC interface. This system can realize the coordinated design of hardware and software. The multiprocessor cooperation can greatly enhance the integration of the system and increase the system's controlling and processing capacity.

3.4.0.4 Sensors

The mobile robot platform should contain multiple sensors to enhance users' situational awareness. These include digital temperature sensors, three-axis accelerom-

eters, speed encoders, gyroscopes, digital compasses, elevation detectors, GPS localizers, wind sensors, and so on. The data from these sensors provide the user with additional information about the remote environment and help create a more complete telepresence experience.

David *et al.* [83] investigated the improvements of the interactions using new algorithms for mixing data from ultrasonic sensor systems with joysticks controlling telepresence mobile robots. Chung *et al.* [102] have proposed a method to realize a more natural assisted navigation for telepresence robots during teleoperation. The proposed method utilized an omnidirectional chassis which is realized by a three-wheel drive mechanism and a new sensing method using ultrasonic sensors.

Sun *et al.* [98] have introduced a mobile robot that can realize navigating and positioning by use of GPS and EC; and can detect the site information by the ultrasonic and wireless camera.

T. Kot *et al.* [62] have tried to take control of the robot and navigation in unknown terrain easier for the operator by providing stereoscopic images and using virtual reality. They have used a new head-mounted display device which displays different images for each eye and designed a new graphical user interface.

Salmeron-Garcia *et al.* [68] used vision-based navigation assistance of a service mobile robot. The information extracted from onboard stereo cameras is processed for shared control of the robot teleoperation, that is, the smooth filtering of the teleoperated commands with the detected obstacles to prevent collisions.

Sato *et al.* [103] introduced a new teleconference system (Telegnosis) in which omnidirectional camera and PTZ (Pan/Tilt/Zoom) are combined. While the Omnidirectional camera captures 360 degrees of surrounding images, the PTZ camera captures more detailed images specified by the users in omnidirectional panorama images by pan, tilt and zoom operations.

J. Du *et al.* [81] have developed a teleoperated robotic 3-D mapping (TeRoM) system which consists of a pan-tilt unit for the RGB-D camera for robot navigation.

3.5 Time delay modelling in the telepresence system

Within a telepresence system, we are interested in addressing issues related to time delay and their impact on robot navigation; and proposing new techniques to compensate for such delays.

Although several methods and algorithms were proposed to address such time delay problems in the telepresence (also *Tele-operated*) systems it is still an open issue that needs to be addressed. The presence of time delay causes instability in the system and poor performance of the robot navigation. In this subsection, we described a few efforts of the time delay compensation methods in the telepresence systems.

Kawabata *et al.* [60] have proposed a framework of human interface systems for teleoperation to achieve smooth operation of a mobile robot through a communication link, considering time delays in data transfer. The prototype of the telepresence system was constructed utilizing the virtual world as an operator interface.

Colledanchise *et al.* [71] has also shown how to use event-based sampling to reduce the number of measurements done, thereby saving time, computational resources and power, without jeopardizing critical system properties such as safety and goal convergence.

Anderson and Vittorias *et al.* [104, 105] have introduced a new control law for controlling a teleoperator with time delay, which achieved stability for the teleoperator independent of time delay in the system. The model is based on the scattering theory, which allows the transmission and encoding of haptic data in time-delayed telepresence systems.

Funda *et al.* [106, 107] in his research proposed a new control methodology, called teleprogramming, which allows for efficient control of a robotic system in the presence of significant feedback delays without substantial degradation in the overall system performance. A teleprogramming system allows the operator to

kinesthetically as well as visually interact with a graphical simulation of the remote environment and to interactively, online teleprogram the remote manipulator through a sequence of elementary symbolic instructions.

An important feature telepresence system is a possible transmission delay and data packet drop-outs over the Internet. If a significant amount of time elapsed or data is dropped due to network congestion, it may result in discontinuity of the reference trajectories and the forces transmitted between the master and the slave. Brady *et al.* [108] developed a new robot-controlling model where communication propagation delays exist over the internet. This model is flexible enough to embrace the wide variety of possible communication mediums for remote teleoperation.

In this context, it is worth noting that the time delay may occur due to the medium of communications, *i.e.,* wired or wireless. While there is a limited chance for the time delay in wired connection, in most application scenarios, wired connections for telepresence systems are not possible as the robots are remotely connected with the operator which is physically distanced. While there are different wireless mediums that can be used such as Bluetooth, WiFi etc., the wireless options are often prone to issues relating to time delays due to the nature of communication medium [109, 110]. Even though Internet protocols such as TCP/IP has a sufficient mechanism for error correction or re-transmission of the missing packets, it all leads to time delay which is the key issue which is being addressed in this work.

Mora *et al.* [85] presented a novel method combining the visualization of two models of the mobile robot inside a 3D virtual environment. One model represents the position and orientation provided by real-time GPS located on the robot and another model is based on inputs given by the human operator through the information gathered by the laser range finder sensor to deal with the time delay and narrow bandwidth limitations.

While Hu *et al.* [86] introduced a 3D model-based predictive display system

where the operator sees the predicted image instead of delayed video to control a robot remotely. Natori *et al.* [111] have presented an effective time delay compensation method based on the concept of network disturbance and communication disturbance observer for bilateral telepresence systems under time-varying delay. They validated the time delay compensation method for both the cases of constant delay and time-varying delay with the Smith predictor.

Bejczy *et al.* [112] and Kikuchi *et al.* [113] proposed the development of predictive display systems based on high-fidelity real-time graphics overlay for use in time-delayed telemanipulation. Human-assisted camera calibration techniques were also developed for an exact alignment of the graphics image with the actual camera view.

Slawinski *et al.* [114,] have proposed a predicted control scheme applied to the teleoperation of a mobile robot with force feedback and time-varying delay. While the user receives delayed force and generates delayed commands permanently, the scheme predicts the user's intention and fuses such commands with a stable controller to achieve a collision-free trajectory for the mobile robot.

3.6 Research goals

In this work, we are solely interested in overall time delay or latency which refers to the time gap between the operator's input action and the received measurement response of the robot. Along with the development of new algorithms that intend to compensate for time delays, we also intend to develop a predictive display-based system at the local site and the proposed system aimed to support real-time robot position tracking and immediate control of the robot at the remote site.

As the mobile robot is controlled by a human operator through a communication network, the human operator should know the robot's current pose to control the robot smoothly. If the time delays were not compensated to estimate the robot pose correctly in the remote site, the human operator may cause an accident crashing any obstacles because of the robot pose which the operator inaccurately

recognised. While time delay is a challenging issue, apparently the above has little coverage in the literature (as described in Section 3.5).

It is worth noting that the causes of time delay or visual and sound delay are not within the scope of this work. We are only interested in the total perceived time delay, which is the time lapse from when the human operator sent a command to the robot in the remote environment until visually perceives the reaction on the robot and receives the robot's pose feedback information.

3.7 Chapter summary

This chapter provides a state-of-the-art analysis of relevant parts of telepresence systems that are relevant to this research, *i.e.,* robust robot navigation in time-delayed environments. Overall telepresence systems are dissected into three categories, namely, 1) remote site, 2) communication channel and 3) local site and issues related to time delays were discussed. It was concluded that while time delay is an important issue, the available literature in this domain is rather limited and therefore demands further research. This sets up the context for our research. Following chapters discuss further details including the development of a framework, the proposition of new delay compensation techniques, and predictive displays.

4 Experimental framework

4.1 Introduction

The main proposition of this thesis relies on the development of new algorithms for time delay compensation in robust navigation of telepresence systems. This chapter proposes a framework configuration to support such development.

The research is about the true state estimation of differential-drive telepresence robots that allow remote presence or activities in challenging environments that are not easy to handle or/and too expensive. Modern technology offered a set of technologies which allow human operators to feel as if they are present, to give the appearance of being present, or to have an effect, using a remote mobile robot, at a place other than their true location. Designing a telepresence system suffers a number of challenges which limit robust robot control in a remote environment. To deal with the challenges we need to develop algorithms to estimate the robot's true state compensating for the issues (*e.g.,* noisy sensor measurement, communication delay etc.) generated during the navigation.

We performed all the experiments within the robotics laboratory of the Centre for Ultrasonic Engineering, Department of Electronic and Electrical Engineering, University of Strathclyde, UK. However, recreating scenarios for telepresence robots is challenging and often impractical in a resource-limited lab environment. In this work we built a new experimental framework which enables modelling of the remote scenarios and hence supports the development of new algorithms, *e.g.,*

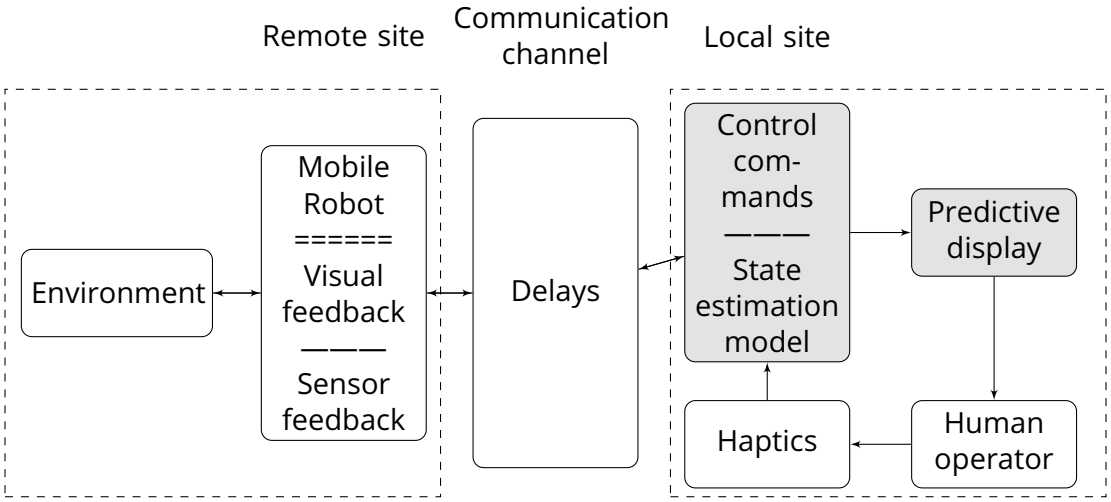


Figure 4.1: A brief architecture of the telepresence system with introducing uncertain time-varying communication delay.

compensating uncertain noisy sensor measurements during remote robot navigation.

4.2 Preliminaries

We aim to address issues related to the time delays affecting robot state estimation in the remote environment. The state estimation included the estimation of the state of the robot's kinematic system by combining knowledge from a priori information and sensor measurements. There is an amount of time gap between sending a control command to navigate the telepresence robot and receiving feedback from the robot about accomplishing the work. While this is a challenging issue, apparently the current literature has little coverage on this. Therefore, to address such issues, we have designed an experimental framework (as shown in Figure 4.1) by which we can manually customise the robot control commands, collect the real-time noisy sensor data, process the measurement data using nonlinear state estimation computational model and estimate the robot's true state by incorporating a new virtual state-based approach.

This work proposes a new experimental framework to address time delay by providing controlled means to emulate scenarios in a lab environment within an

enclosed space. Our ambition is to develop algorithms that help in estimating the robot's true state by modelling and compensating for communication time delays using uncertain delayed sensor measurements in telepresence robot navigation. State estimation in dynamical systems is crucial in real-world applications as the true state is unknown and sensors have limited precision, therefore, provide only a sequence of uncertain noisy measurements. We use the proposed framework for error modelling and algorithmic development purposes.

Referring Figure 4.1, in the local site the human operator uses haptic devices to transmit a control command to navigate the mobile robot around the remote environment. The mobile robot includes sensors to provide visual and positional feedback to the host computer at the local site. The host computer with a state estimation algorithm receives raw sensor data from the remote site and estimates the robot's true position which displays on a predictive display. In this system, the human operator uses real-time images and videos transmitted from the mobile robot captured by the sensors to estimate the robot's true state.

However, the operator cannot execute a smooth operation in this cycle, because there is 1) measurement noise due to sensor precision and environmental impact, and 2) more importantly time delays which largely depend on the state of the communication network which is often inconsistent and not necessarily continuous. Therefore, it is important to develop algorithms that can estimate the robot's true state by modelling positional and time delay-related errors in such scenarios. However, the recreation of real scenarios for telepresence robots for controlled experiments is not only complicated due to limited University lab space and physical location constraints but also impractical, expensive and not always necessary.

The proposed framework explains the procedure to collect the raw sensor data, model the time delay and estimate the true pose of a commercially available differential drive telepresence robot. The uncertainty of the delayed sensor measurement was modelled using a state estimation algorithm controlled by the

human operator. This process is particularly challenging especially for a differential drive robot where additional system errors occur due to the kinematics of individual wheels. Our framework enabled us to model and compensate for such errors in real laboratory environments where the robot navigation is captured in real-time using a multi-camera VICON motion capture system [49].

4.2.1 Chapter contributions

Major contributions of this chapter are:

- Development of Robot Operating System (ROS [46]) based experimental framework, designed to manoeuvre the robot and collect real-time sensor measurement data from the remote environment to the local site to estimate the robot's true state.
- Development of framework features that enabled the use of commercially available state-of-the-art differential-drive telepresence robot for research purposes.
- Experimentation with a non-linear filter-based state estimation technique that has been used to analyse and model the uncertain noisy sensor measurement.
- Methods to verify experimental claims using a set of VICON motion cameras emulating a real-environment scenario.

An overall experimental framework is shown in Figure 4.2 and details of the framework along with selected experimental results are described in the following sections.

4.3 Experimental framework

The proposed framework was built based on our experimental requirements, *e.g.,* state estimation of a telepresence robot in an environment with erroneous sensor

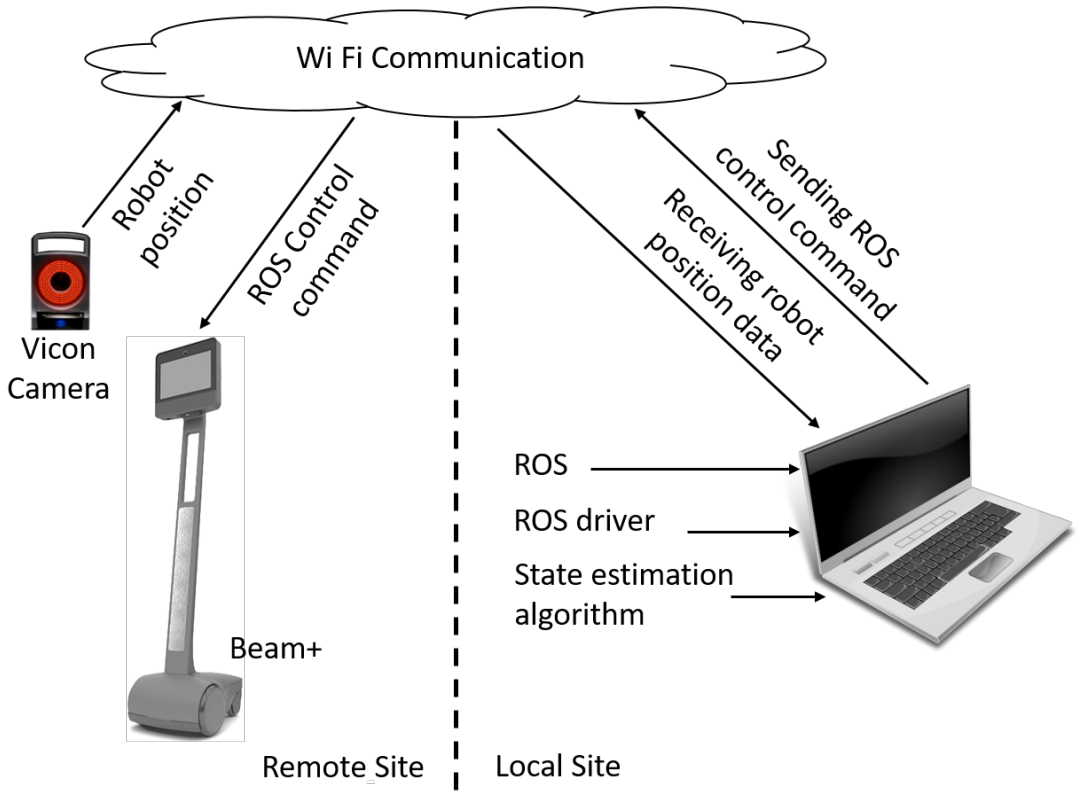


Figure 4.2: Experimental framework with a state-of-the-art off-the-shelve telepresence robot, controlled by customised ROS driver and VICON motion capture camera setup.

measurement due to system noise and uncertain delays (communication, processing etc.). We captured these requirements in the following section and described our framework which helps us to design, simulate and experiment in a controlled lab environment. The overall system architecture of our framework is depicted in Figure 4.2.

4.3.1 Requirements and framework components

In order to simulate various usage scenarios in our research there were some criteria that needed to be fulfilled in selecting a telepresence robot and setting up the software-hardware environment. These criteria are, the robot should be

1. a mobile robot,

2. remotely operable in a controlled manner

3. has teleconferencing capabilities, and

4. a means to track the robot's real position.

In fulfilling these requirements we make use of various components which are part of the framework: 1) telepresence robot, 2) control and simulation software, *i.e.,* ROS, 3) robot navigation and path planning and 4) tracking hardware. These components are described in the following subsections.

4.3.1.1 Telepresence robot

In order to have greater control over the robot, it is sensible to choose a differential-drive robot that has control of individual wheels. We have used Beam plus [116] which is a state-of-the-art market-leading differential drive telepresence robot in our experiments which has telepresence capability and control through WiFi communication. Beam plus has two built-in high dynamic range cameras, an LCD display and four microphone arrays with powerful audio amplifiers to provide a real, physical sense of presence in the remote environment. Beam plus is a smaller in size, less costly, Linux-powered mobile telepresence robot. The remotely-piloted Beam plus bot, which can be controlled via WiFi, run low-latency Skype-like video conferencing software on top of a Ubuntu-based embedded Linux OS as shown in the Figure 4.3. The Beam plus contains an embedded computer platform consisting of an Intel processor running a customized Ubuntu 12.04-based Linux OS, along with extensive Suitable Technologies application software.

Summary of Beam plus specifications Suitable Technologies currently lists these specs for the Beam Plus device:

1. Battery life: 2 hours of call time and 4 hours to charge

2. Display: 10-inch LCD flat-panel

3. Built-in cameras: two 640×480 HDR cameras; 30 fps video

Figure 4.3: The mobile telepresence robot Beam plus with the charging station.

4. Audio: 4-microphone array; powerful audio amplifier

5. Connectivity: dual-band 2.4GHz/5GHz WiFi

4.3.1.2 Control and simulation software

There are two parts of the software component in our framework: a) robot control using ROS and b) state estimation algorithmic development in MATLAB.

ROS and ROS driver The proposed framework used ROS to navigate and control the robot. While there are provisions to execute ROS commands through MATLAB, within the scope of this work we have used ROS and MATLAB separately. Although ROS is widely used in robotics, it is challenging to control any commercial robots with a closed ecosystem using ROS. A ROS driver (a ROS node to access the hardware) was used for experimental purposes. An existing open-access robot driver (*rosbeam* [48]) was modified, customized and installed in our Beam+ robot to communicate through an Internet (TCP/IP) enabled WiFi access point (2.4 GHz in our experiment) with a Linux-based (Ubuntu) host computer.

The robot and the host computer were connected through a WiFi network and communicated via TCP/IP. Several ROS packages that solve basic robotics problems including pose estimation, localisation in a map and mobile navigation were used in this work which includes several commands for launching nodes, introspecting topics and publishing control actions as a host to the telepresence robot. Using ROS commands we developed algorithms to instruct the robot to navigate following the pre-defined trajectory, monitor its progress, stop or redirect it along the way, and be told when it has succeeded (or failed). Also, we captured the robot's positional information during the navigation.

Robot navigation and path planning Controlled navigation of the mobile robot was carried out by implementing algorithms in ROS in a Linux-based host computer. The host computer connects the telepresence robot using the ROS driver and sends ROS control command to the robot to form a *raster-scan* navigation path and receives 3D positional data of the navigation captured by the VICON motion cameras through standard WiFi. The motion capture data was used for two purposes: a) to verify the system path and b) to simulate other scenarios by introducing noise and delay which helped to develop the experimental setup for robust navigation.

4.3.1.3 Tracking hardware

We captured the robot's navigation data using VICON motion capture system [49]. Twelve motion-captured cameras were installed and calibrated in the University lab which are capable of tracking the robot's true state during the experiment. We have attached some retro-reflective markers on the robot to represent it as a rigid body. VICON cameras along with their software were used to record the movement of the robot. They operate in three dimensions and tend to have high resolution, high accuracy and low variance [117, 118].

4.3.2 Kinematic model of the telepresence robot

The mobile robot was used in this research a four-wheeled differential drive telepresence robot. Two front wheels are drive wheels and the rear two wheels are castor wheels for stability. The drive wheels were controlled independently using Robot Operating Systems (ROS [46]) commands sent by the human operator from a host computer.

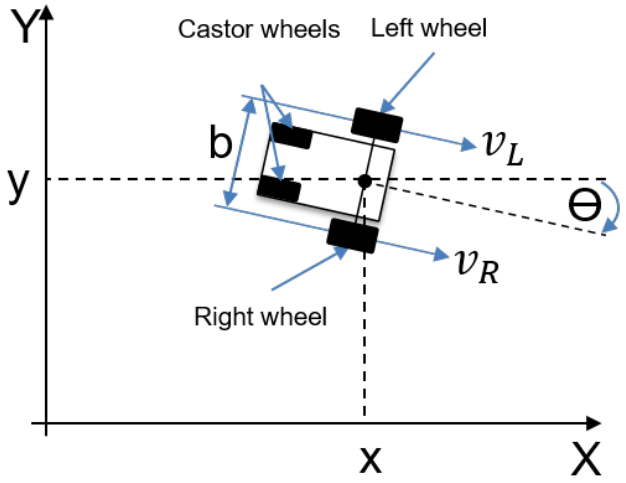


Figure 4.4: Kinematic model of the four-wheeled differential-drive telepresence robot where two front wheels are drive wheels and rear two wheels are castor wheels for stability.

The kinematic model [119] of the differential drive robot is defined as follows:

$$x_k = x_{k-1} + \Delta V_{k-1} * \cos\theta_k, \tag{4.1}$$

$$y_k = y_{k-1} + \Delta V_{k-1} * \sin\theta_k, \tag{4.2}$$

$$z_k = z_{k-1}, \tag{4.3}$$

$$\theta_k = \theta_{k-1} + \Delta\theta_{k-1}, \tag{4.4}$$

$$\Delta V_{k-1} = \frac{1}{2} * (v_{l,k} + v_{r,k}) * dt, \tag{4.5}$$

$$\Delta\theta_{k-1} = \frac{rw}{b} * (\omega_{l,k} - \omega_{r,k}) * dt, \tag{4.6}$$

where, x_k, y_k and z_k are the Cartesian coordinates of the robot, ΔV_{k-1} is the travelled distance at time step $k-1$ to k, θ_k is the angle between robot and x axis, $\Delta\theta_{k-1}$ is the rotation angle at time step $k-1$ to k, $v_{l,k}$ and $v_{r,k}$ are linear velocity of left wheel and right wheel, $\omega_{l,k}$ and $\omega_{r,k}$ are angular velocity of left wheel and right wheel, rw is the radius of the two drive wheels and dt is the sampling time.

4.3.3 Addressing systematic errors

In a differential drive mobile robot, incremental odometry errors are usually caused by the kinematic imperfections of the robot. The two most significant errors are generated from unequal wheel diameters and the uncertainty about the effective wheelbase [120]. As the Beam plus telepresence robot is a differential drive robot as described in Figure 4.4, it produces a significant amount of dead-reckoning errors during navigation. Using the UMBMark [120] method we have measured the dead-reckoning accuracy of the robot to find out the variance during the robot navigation and modelled it in the state estimation algorithm. The error E_d for the unequal wheel diameters is defined as,

$$E_d = D_R/D_L,$$

where, D_R and D_L are the actual wheel diameters. The error E_b for the uncertainty about the wheelbase is defined as,

$$E_b = b_{actual}/b_{nominal},$$

where b is the wheelbase of the mobile robot. If the average of two actual wheel diameters differs from the nominal wheel diameter, then the robot will experience an additional dead-reckoning error, called scaling error (E_s). E_s affects both the straight line and the turning motion and is easy to measure with just an ordinary tape. We have used UMBmark method [120] for measuring, comparing, and correcting dead-reckoning errors in our telepresence robot. A square path

experiment is performed ten times in both clockwise (cw) and counter-clockwise (ccw) direction as shown in Figure 4.5 and Figure 4.6.

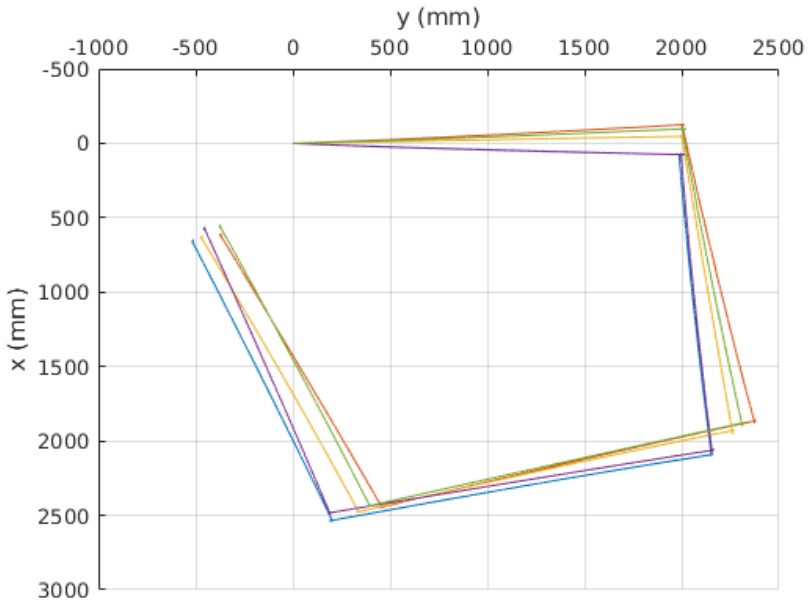


Figure 4.5: Square path formation by the telepresence robot in the clockwise direction.

The method classifies wheelbase errors as Type A and unequal wheel diameter errors as Type B and therefore it produces different errors when rotates in clockwise and counter-clockwise directions. If the robot had only Type A errors then it would travel in perfectly straight lines but at the corners, it would turn some amount more or less than 90° depending on the size of the error. We defined this difference in angle as α.

On the other hand, if the robot had only Type B errors it would make perfect 90° turns but move in an arc rather than a straight line. At the end of each leg of the square, the robot will have drifted off course and changed its heading by some amount, depending on the Type B error. This change in the heading is defined as angle β. The effect of Type A and Type B errors have shown in Figure 4.7.

However, unlike Type A errors, Type B errors are not symmetrical. It is possible that Type A and B errors cancel each other out and give the impression that there

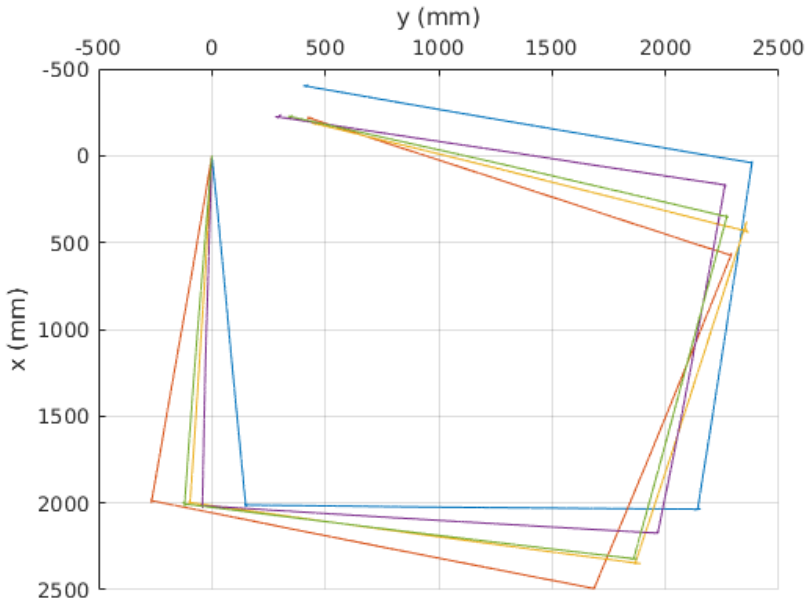


Figure 4.6: Square path formation by the telepresence robot in the counterclockwise direction.

is no error. This is partially why the UMBMark procedure uses both clockwise and counterclockwise measurements to calculate α and β accurately. The reason for performing the tests 10 times in both directions is to minimize the effect of non-systematic errors. Averaging the measurements give us *centre of gravity* (COG) for the systematic errors as shown in the Figure 4.8.

The COG values for x and y in both *cw* and *ccw* directions were used in the following formulas to calculate E_b and E_d. Using x or y values we can calculate α and β (in degrees) in the following two equations, respectively:

$$\alpha_x = \frac{x_{cw} + x_{ccw}}{-4L}\frac{180^\circ}{\pi},$$
$$\beta_x = \frac{x_{cw} - x_{ccw}}{-4L}\frac{180^\circ}{\pi},$$
$$\alpha_y = \frac{y_{cw} - y_{ccw}}{-4L}\frac{180^\circ}{\pi},$$
$$\beta_y = \frac{y_{cw} + y_{ccw}}{-4L}\frac{180^\circ}{\pi}.$$

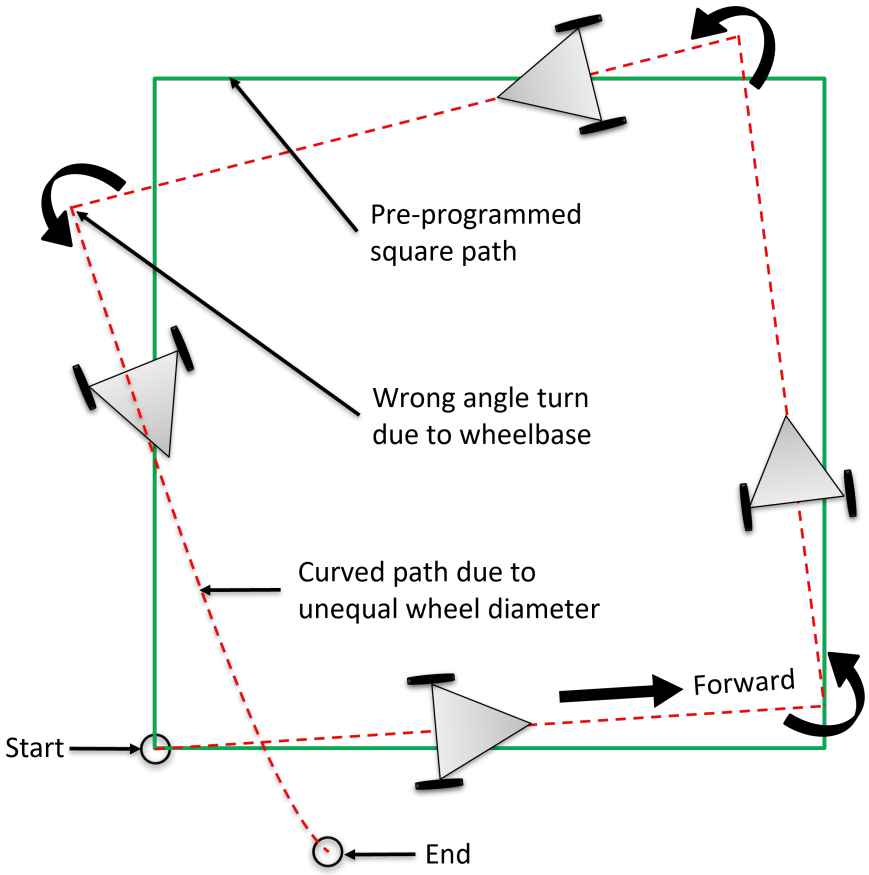


Figure 4.7: The effect of two systematic dead-reckoning errors, E_b for the uncertainty about the wheelbase and E_d for the unequal wheel diameters.

With β and α we can calculate E_d and E_b, respectively:

$$E_d = \frac{D_R}{D_L} = \frac{R + b/2}{R - b/2}$$
$$\text{where,} \quad R = \frac{L/2}{sin(\beta/2)},$$
$$E_b = \frac{90°}{90° - \alpha}.$$

Using these values we calculated two correction factors to implement in the state estimation algorithm to correct the robot navigation.

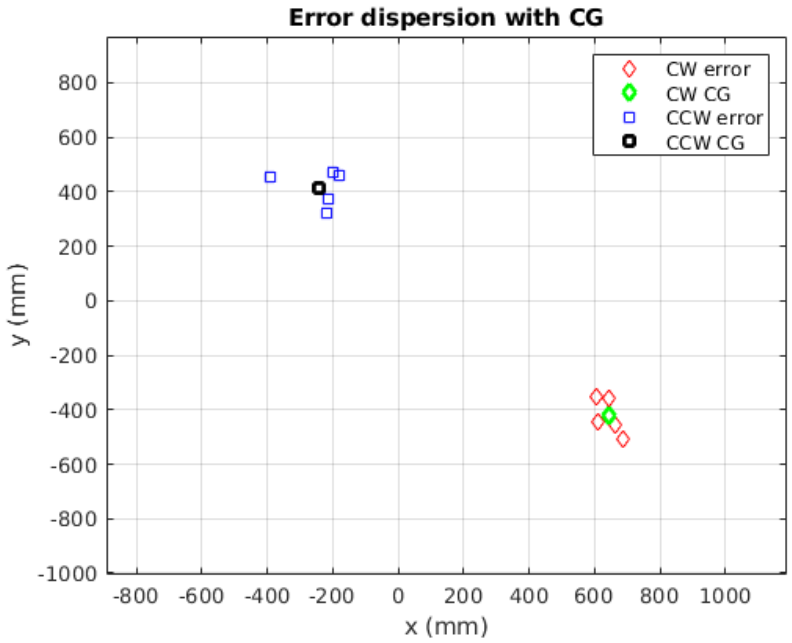


Figure 4.8: Results from running the UMBMark method in the clockwise and counterclockwise direction with an uncalibrated robot.

$$c_L = \frac{2}{E_d + 1},$$
$$c_R = \frac{2}{\frac{1}{E_d} + 1}.$$

The corrections were adopted during the algorithmic development and the output shows that the estimated robot path coincides with the VICON captured data path as shown in Figure 4.9. The result demonstrates that the proposed data collection framework effectively estimates the robot's true pose and can be used for robust robot navigation.

4.3.4 Robot operating system (ROS)

ROS is one of the software frameworks used in most universities for robotics research. Other than academic research, ROS is also used in robotics companies to prototype their software. Hobbyists used ROS to create different robot applications.

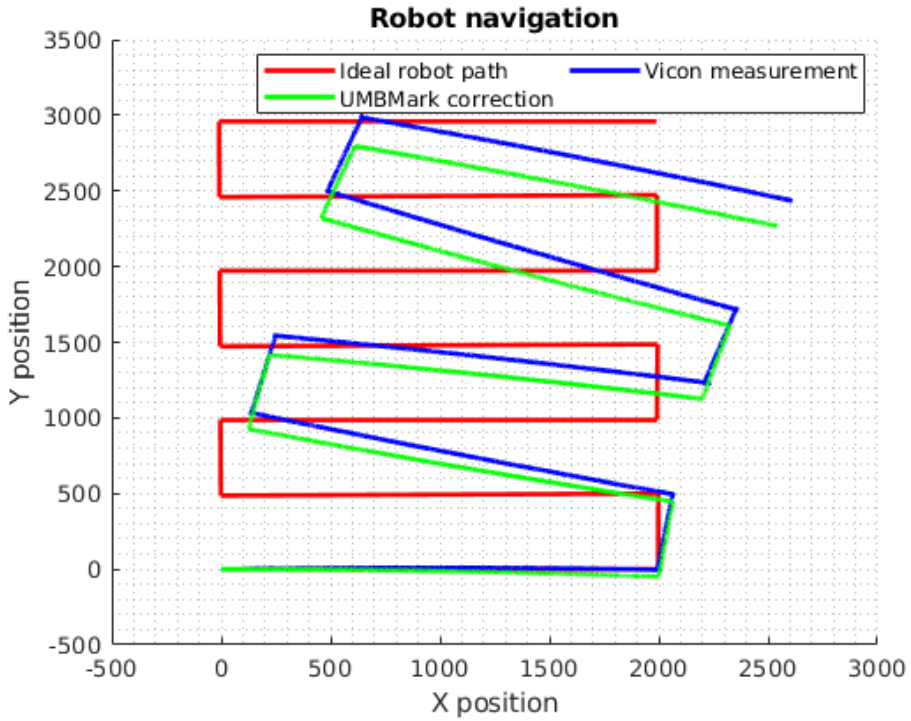


Figure 4.9: Systematic error correction using UMBmark method and validation using the proposed framework.

ROS is an open-source meta operating system or a middleware used in programming Robots. It is developed as part of the STAIR project [121] as well as the Personal Robotics Program [122] at Stanford University in cooperation with the robotic manufacturer Willow Garage. It consists of packages, software, building tools for distributed computing, and architecture for distributed communication between machines and applications. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. It can be programmed using python, c++, and lisp.

ROS is not a full-fledged operating system, it is a "*meta operating system*". It is built on top of a full operating system like Ubuntu. It is called an OS because it also provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management.

ROS distributions are named alphabetically. ROS can be officially built on Linux distributions but it also supports other operating systems. In this research, we have used the ROS version Kinetic Kame on the officially supported Ubuntu 16.04 LTS Linux distribution.

ROS package contains libraries, executables, scripts, and other artefacts for a specific ROS program. Packages are used for structuring specific programs. *gazebo_ros_pkgs* is a meta-package which provides packages for integrating ROS with the Gazebo simulator.

A ROS system is comprised of a number of independent nodes, each of which communicates with the other nodes using a publish/subscribe messaging model. ROS starts with the ROS Master. Publishers and Subscribers register to the master, then ROS Master tracks ROS topics being published by the publisher and ROS Topics being subscribed to by the subscribers as shown in the Figure 4.10.

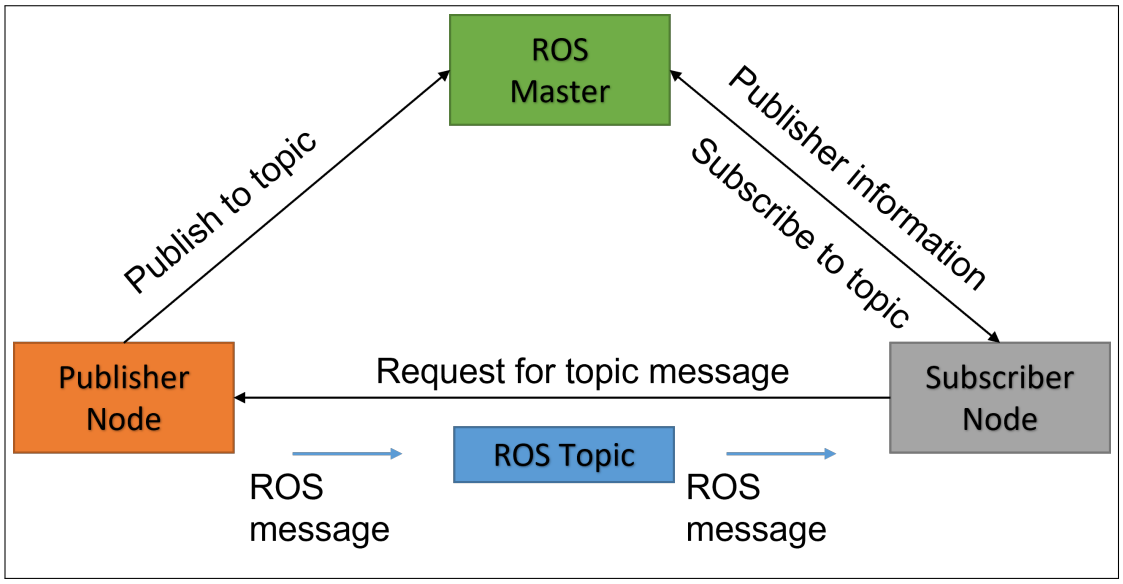


Figure 4.10: Overview of the Robot Operating System comprised with a number of nodes, topics, subscribers and publishers [123].

ROS Topics are the buses used by ROS nodes to exchange messages. Topic transport message between a publisher node and a subscriber node. Nodes communicate by sending ROS messages to each other using ROS Topic. A message can be of primitive type integer, floating-point, Boolean, etc. ROS service is one-to-one

two-way transport, it is suitable for request/reply interactions. A ROS node(server) offers a service, while another ROS node(client) requests for the service. The server sends a response back to the client.

Rviz is a 3D visualization environment that lets one combines sensor data, robot model, and other 3D data into a combined view. rqt_plot lets you visualize scalar data published to ROS topics. rqt_graph displays a visual graph of the processes running in ROS and their connections. ROS launch is used for starting and stopping multiple ROS nodes. Published topics are saved as .bag files, rosbag command line tool is used to work with bag files.

We aimed to build a ROS control command program for the telepresence robot to form different navigation paths. It allows us to remotely control and monitor the robot, connected to the host computer via WiFi connection. The robot was controlled by using a Keyboard on our computer. The main languages for writing ROS code are C++ and Python. The telepresence robot was connected to the host computer through beambridge driver.

4.3.5 Motion capture system (VICON)

In this research, motion capture is a vital component of developing the experimental framework. The telepresence robot must be able to follow the command of the human operator in order to navigate and interact in a remote environment. The motion capture system helped us to validate the experimental scenarios designed for the telepresence robot. The motion capture system was used to record the precise position and orientation of the robot at high frequency.

The motion capture system used in this research was the VICON motion capture system which involves multiple high-definition cameras. The VICON system is composed of multiple cameras set up around the perimeter of the measurement workspace at varying heights to obtain a full 360-degree view of the field, as seen in the experimental setup in Figure 4.11. Some retro-reflective markers were attached to the telepresence robot. Light from the VICON cameras is emitted and

reflected back from markers in the field of view. This yields the 3D position of each marker. Raw data sets were collected from varying different experimental parameters used in the experiments. The raw data sets were entered into the state estimation algorithm for further processing. The VICON system was connected with the host computer through *vicon_bridge* driver.

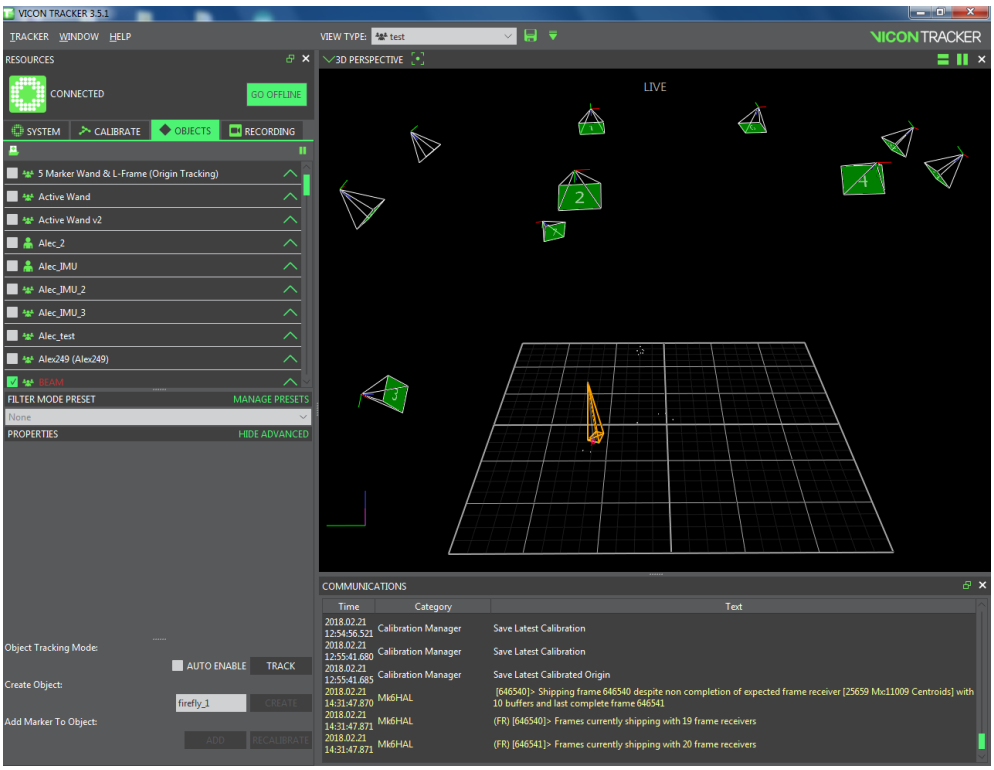


Figure 4.11: VICON motion capture System showing the robot as a target tracking object.

4.3.6 Framework set-up and data collection

The human operator connected the host computer and the telepresence robot using the ROS driver and sent the ROS control command to the robot to form a raster-scan navigation path and received 3D positional data of the robot navigation captured by the VICON motion cameras through the WiFi network.

The experiment was to create a raster-scan path where the mobile robot travel distance was horizontal 2000 mm and vertically 500 mm and the orientation was $90°$. We performed several runs to capture measurement data with a combination

of various linear velocities ($100 - 500$ mm/sec) and angular velocities ($100 - 500$ mm/sec). For all the experiments the mobile robot's starting pose was the same. The data is collected at a rate of 100 Hz.

The ROS control commands were sent from the host computer over the WiFi to manoeuvre the robot creating the predefined path. Position and orientation data were recorded and used as measurement data in the proposed algorithm. The robot wheel diameters and wheelbase were modified using the correction factor calculated by dead-reckoning.

On the other side, the captured robot navigation data using VICON cameras have a low variance (3.58 mm^2) as reported in [117, 124] which was also used to estimate the robot's true position. It is to be noted that the VICON captured positional data was used to simulate the noisy measurement by introducing random white noise.

Experimental parameters such as initial robot position, linear and angular velocity, correction factors for wheelbase and wheel diameter, robot variance and measurement time steps *etc.* are provided in Table 4.1.

Experimental parameters	**Value**
Initial position (x, y, z, θ)	(0,0,0,0)
Wheel radius (rw)	75 mm
Wheelbase (b)	263 mm
Velocity ($v_l, v_r, \omega_l, \omega_r$)	100 mm/sec
Wheelbase correction factor (E_b)	0.9691
Wheel diameter correction factor (c_l, c_r)	0.9969, 1.0031
Robot variance ($\sigma^2_{\Delta\theta} = \sigma^2_V$)	2.13
VICON measurement time steps	0.01 sec

Table 4.1: Parameters used in the real environment experiments using the proposed framework [125].

4.4 Measurement data

The VICON cameras captured various datasets of the robot's positional information during the navigation. The output of the VICON recorded data reports the translational x, y and z coordinates of the robot and rotational information as Euler angles. We assumed that the datasets were not ideal and could be delayed.

4.4.1 Time variant and noisy measurement data

The total time delay of the telepresence system is a combination of a number of reasons, such as network switching delays, bandwidth limitation, communication drop-out, hardware processing delays and slow dynamics of the mobile robot. Time delays can be caused by physical limitations such as distance or obstacles between the operator and the robot too. Total time delay can be both certain and uncertain.

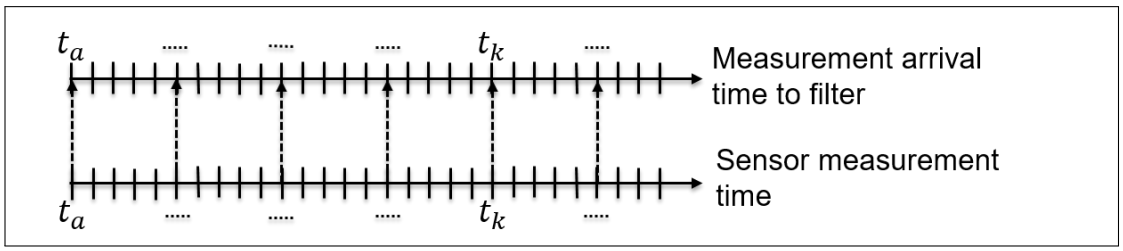


Figure 4.12: The figure shows an ideal case when sensor measurement data enters into the filtering algorithm without any time delay (Figure adapted from [126]).

In an ideal case considering **no time delay** in the system, in a single time occurrence, when sensor measurement data arrives at the computer, it coincides with the measurement data at the same time step available in a Filter as shown in Figure 4.12. In such cases filtering methods like EKF algorithm [127] [50] can be applied.

However, in reality, the system is assumed to be **delayed**. Considering time delay as **certain** in nature both time steps do not coincide with each other, which produces an amount of time delay during navigation as shown in Figure 4.13. In

such cases, the measurement equation should be redefined as [128]

$$z_k = h(x_{k-\tau_k}, v_{k-\tau_k}),$$

where τ_k is the number of delayed time steps.

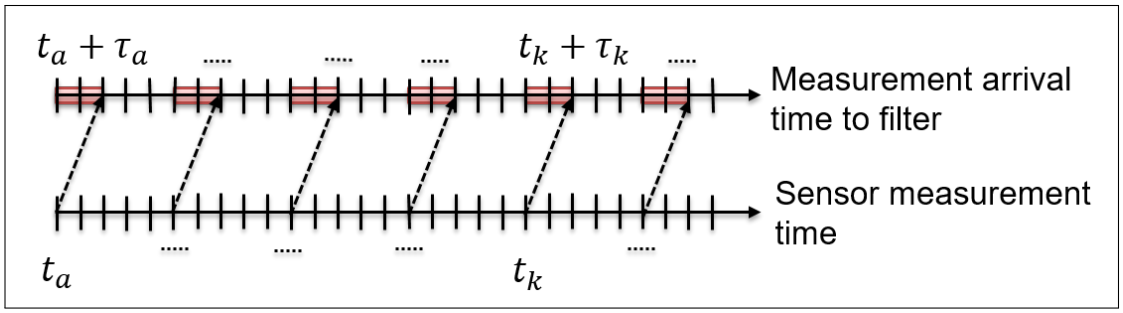


Figure 4.13: If the measurement data is corrupted by a certain time delay, the measurement data arrival time in the computer does not coincide with the moment when the data enters into the filtering algorithm (figure adapted from [126]).

In our case, we assumed that the **time delays τ_k are not precise or uncertain** due to various delays, namely, feedback delay and transmission delay and data packet drop-outs over the Internet due to network congestion and hence the arrival in different filter measurement time duration Δt are random as shown in Figure 4.14.

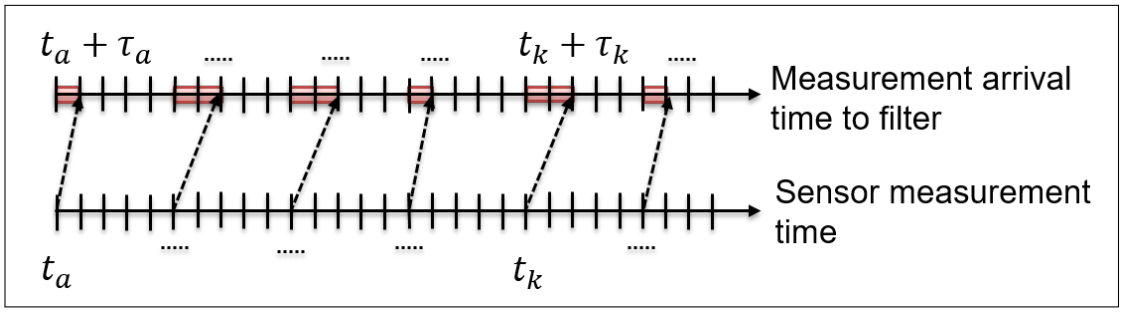


Figure 4.14: If the time delay is uncertain in nature and the arrival at different time steps in the filtering algorithm is random (Figure adapted from [126]).

Therefore, it is important to model uncertain delays to obtain a consistent state estimator. Such uncertain delays are modelled here by probabilistic density functions (PDF). When the measurement data arrived at the filter, the probability that

the measurement at a given time step was calculated by integrating the PDF over the time interval as shown in Figure 4.15.

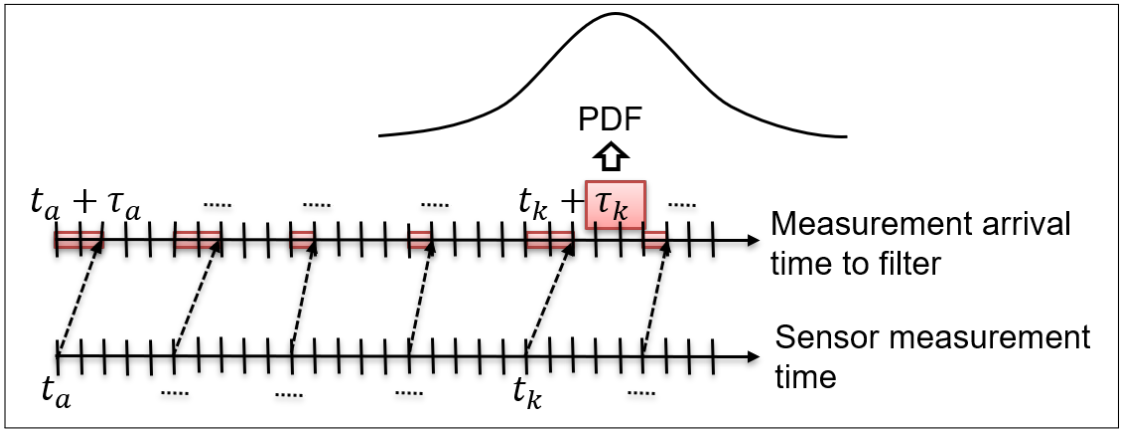


Figure 4.15: If the measurement data arrival is uncertain, the time delay is considered as a probabilistic density function (*e.g.,* Gaussian or Gamma). The probability of the time step in example measurement data is shown in this figure (Figure adapted from [126]).

It is worth noting that one may use timestamps instead of PDFs. For example, one can use online or offline clock synchronisation [129], which can be achieved through Network Time Protocol (NTP) or Global Positioning System (GPS) and IEEE 1588. However, online delay calculations are generally not used in practice for operations of transport protocols [129]. More importantly, time stamp alone is unlikely to provide an efficient solution to the state estimation problem as varying time delay (potentially calculated from the time stamps) will increase the computational overhead of the delay compensation algorithm. Therefore, it is not always necessary to compute precise delay which can be modelled (as described in Section 4.4.2) using known distributions, *e.g.*, Gaussian and Gamma and hence used here in the proposed algorithm.

The probability of the measurement data in k^{th} time step can be expressed as,

$$\begin{aligned}\delta_k &= P\left((t_k - \tfrac{\Delta t}{2}) \leq t < (t_k + \tfrac{\Delta t}{2})\right), \\ &= \int_{t_k+\frac{\Delta t}{2}}^{t_k-\frac{\Delta t}{2}} p(t)dt,\end{aligned}$$

where, $P(.)$ denotes probability and $p(.)$ denotes the PDF.

4.4.2 Delay distributions in the measurement data

The Kalman filter is a state predictor for linear systems with Gaussian noise, but the Extended Kalman filter extended its validity to nonlinear systems with non-Gaussian noise. For Internet-based real-time teleoperation systems, uncertain and variable time delays can cause instability and jeopardize the performance of the system.

The delay distributions have been studied in the literature, and in this work, we have focused on two most commonly reported PDFs, *i.e.*, (1) Gaussian and (2) Gamma distributions. The former one represents general delay modelling when delay distributions are not known [54, 55] and the latter one constitutes Internet-based delay models when the Gaussian model fails in characterising the distribution property or the distribution of the input traffic rates is non-Gaussian [51–53]. These PDFs are defined as follows:

Gaussian distribution: The probability density function of the Normal distribution or Gaussian distribution is:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2},$$

where the parameter μ is the mean or expectation and σ is the standard deviation of the distribution.

Gamma distribution: It is a two-parameter continuous probability distribution. The probability density function in the shape-rate parametrization is

$$f(x) = \frac{\beta^{-\alpha} x^{\alpha-1} \exp(-\frac{x}{\beta})}{\Gamma(\alpha)} \quad \text{for } x > 0 \text{ and } \alpha, \beta > 0$$

where $\Gamma(\alpha)$ is the Gamma function and α and β are shape and rate parameter. The shape parameter for the Gamma distribution specifies the number of events that are being modelled and the scale parameter represents the mean time be-

tween events. These parameters are related to the mean and variance of the delay where

$$\mu = \alpha/\beta,$$
$$\sigma = \beta^2/\alpha,$$

and $\Gamma(\alpha)$ is the gamma function for all positive integers, $\Gamma(\alpha) = (\alpha - 1)!$.

While the most commonly used delay distributions are Gaussian and Gamma distributions, other probability distributions such as uniform distribution could be used to model the uncertain time delay, which is outside the scope of this research. In order to compensate for the time delay, we considered using Gamma distribution in our algorithm assuming the probability of the nature of the time delay as non-Gaussian.

Gamma Distribution is a Continuous Probability Distribution that is widely used to model continuous variables that are always positive and have skewed distributions. The Gamma distribution can model the elapsed time between various numbers of events. The details of the Gaussian and Gamma distribution parameters and other experimental parameters used in this research work are provided in the Table 4.2. It is worth noting that we considered IP latency between cities as a starting point, *e.g.,* two cities 100 miles apart may have a 10-15ms delay [5]. For example, round trip IP latency between London and New York (Trans Atlantic) is around 90ms or Trans-Pacific is 100ms [130]. However, multiple other factors such as system delay and local network delay would add further latency. For these reasons, we used 100 ms or 0.1s as starting delay in our research and then increased it in regular intervals for further investigation. For simplicity, considering 100 miles as a representative unit, we represent 1 time step is equivalent to 10ms (0.01s) and used here for the experiments.

Experimental parameters	Value
Certain delay parameters	
Known or certain delay in time steps (τ)	[10, 15, 20, 25]
Known or certain delay in sec	[0.1, 0.15, 0.2, 0.25] sec
Uncertain delay parameters	
Gaussian parameters for uncertain delay	$\mu = [10, 15, 20, 25]$ and $\sigma = \tau/4$
Gamma parameters for uncertain delay	$\alpha = \beta = \sqrt{\tau}$

Table 4.2: Delay distribution parameters used in the experiments to minimise the effect of time delay during telepresence robot navigation.

4.5 Chapter summary

This chapter proposes a new framework for the development of delay compensation algorithms in telepresence robots and provides the necessary platform for controlled experiments. Development of such a framework is generally complex and requires the integration of multiple hardware and software tool sets, the development of control algorithms for navigation and managing a robot trajectory monitoring system. In this work, we used a commercial differential drive telepresence robot Beam+, a robot operating system for control and navigation, and a VICON motion capture system to capture measurement data. This chapter also provides detailed information on the experimental parameters and details of systematic error correction using the UMBMarc method. The framework is further used in the following chapter (Chapter 5) when developing the delay compensation algorithm with augmented states and Extended Kalman Filter.

5 State estimation using non-linear filter algorithm (AS-EKF)

This chapter proposes a new approach for state estimation assuming uncertain delayed sensor measurements of a telepresence robot during navigation. A new real-world experimental model, based on Augmented State Extended Kalman Filter (AS-EKF), is proposed to estimate the true position of the telepresence robot. The uncertainty of the delayed sensor measurements has been modelled using probabilistic density functions (PDF) [51–55].

The proposed model was successfully verified in our proposed experimental framework which consists of a state-of-the-art differential-drive telepresence robot (Beam plus here) and a motion-tracking multi-camera system (VICON) as described in Chapter 4. The results show significant improvements compared to the traditional EKF that does not consider uncertain delays in sensor measurements. The proposed model will be beneficial to build a real-time predictive display by reducing the effect of visual delay to navigate the robot under the operator's control command, without waiting for delayed sensor measurements.

5.1 Introduction

A telepresence system provides interactive two-way audio and video communication with a remote sender and a receiver for building a communication system between two people in different places. The telepresence system introduces time-varying delays in the reference commands and feedback signals resulting in insta-

bility or poor performance of the system. In this research, a new approach of state estimation is presented that can model and compensate for time delays present in the robot sensor measurements which are uncertain in time during robot navigation.

We hypothesized multiple augmented states in the proposed approach to estimate the true position of a commercially available differential drive telepresence robot. The uncertainty of the delayed sensor measurement was modelled using probabilistic density functions (PDF). This is particularly challenging, especially for a differential drive robot where additional system errors occur due to the kinematics of individual wheels. While there have been several attempts to address the delay problem in mobile robots and teleoperated systems, to the best knowledge of the authors this is the first time such a hypothesis is applied to a commercially available differential drive telepresence robot in a real environment experimental framework.

The time elapsed between making an action decision by the human operator and perceiving the consequences of that action in the environment introduces a time delay. Total time delay can be both certain and uncertain. If the time delay is known or certain, the past state can be predicted by applying backward prediction of the current state. Bar-Shalom [131] proposed an optimal and sub-optimal algorithm for one-step delayed measurement. The extended version for multi-step delayed measurements is also proposed in [132]. In the case of a non-linear system, it needs modifications for state estimation. Larsen *et al.* [133] introduced a method based on extrapolation of a delayed measurement to the present time using past and present estimates of the Kalman Filter. An extension algorithm of [133] is proposed in [128] that interpolating a delayed measurement minimizes the computational time even for significant time delays.

State augmentation has also been used in the time-delayed measurement. Delayed measurement directly corrects the past state and a new prediction of the current state is then obtained from the corrected past state. Challa *et al.*

[134] presented a Bayesian solution to the out-of-sequence measurement (OOSM) problem and provided approximate, implementable algorithms for both cluttered and non-cluttered scenarios involving single and multiple time-delayed measurements. Van Der Merwe *et al.* [135] applied the sigma point Kalman Filter instead of EKF to the augmented technique to fuse latency-lagged observations for non-linear estimation and multiple sensors fusion.

If there is uncertainty in measurement delay, it is hard to predict because the measured time delay may have noise. Julier and Uhlmann [136] suggested a co-variance union algorithm for accommodating time step uncertainty directly into the observation co-variance so that filter consistency is always maintained. Jun *et al.* [137] proposed event-based filtering for time-varying non-linear systems that use probabilities to address uncertain missing measurements. A recursive filtering algorithm is proposed by Zou *et al.* [138] targeting a class of linear time-varying systems of networked sensors for robust signal transmission.

Within the scope, this paper only focuses on estimating overall time delays (uncertain in nature) for robot navigation. The underlying assumption in such cases is that if the time delay is able to be modelled as a form of a distribution, the uncertainty of delay can be modelled. Challa *et al.* [134] proposed a probabilistic data association filter to deal with data association issues arising from the presence of clutter in the OOSM problem. Choi *et al.* [126] proposed a state estimation algorithm incorporating uncertainty of measurement delay. Modelling uncertain delay as a probabilistic density function is accounted for by the proposed estimator, combined with the augmented state Kalman Filter. However, the majority of these algorithms reported simulation-only results that neither considered a real environment nor the techniques were applied to a real robot.

5.1.1 Chapter contributions

While state estimation with augmentation methods have been proposed in the literature, they were not applied in telepresence navigation. The only exception is

Choi *et al.* [126] where the authors simulated their algorithms with the intention to apply to telepresence robot navigation. However, a real-life scenario poses many other additional challenges including system errors, and mechanical errors relating to robot kinematics. In addition to this, a differential drive robot (our chosen industrial telepresence robot Beam plus) imposes additional complexities mainly due to individual wheel controls impacting the robot's kinematics.

To address this we propose an experimental model for state estimation for the navigation of telepresence robots with uncertain and delayed sensor measurements in a non-linear system. We hypothesised and developed the approach by introducing augmented states into the computational model for differential drive telepresence robot navigation. This is verified using a real-world telepresence robot navigation in the laboratory environment using a new experimental framework. The contributions of this work are:

- A delay compensated non-linear state estimation approach considering continuous or uncertain time delay in measurement data. Multiple augmented states, considering delay as a model of Probability Density Function (PDF) in the form of Gaussian or Gamma distribution, is applied within the filtering method to estimate the true robot position from delayed measurements.
- A new real-environment experimental framework is used for telepresence robot navigation to evaluate the performance of the proposed non-linear filter-based state estimator.
- To prove the success of our approach, the proposed model was experimentally verified on a state-of-the-art differential-drive telepresence robot in the real environment using the proposed framework.

The idea and results were reported in the form of conference [139, 140] and journal [141] publications. This chapter provides a detailed mathematical formulation of the proposition, details of the algorithmic development, and a description of experimental results and discussions. To the best of our knowledge, the pro-

posed approach is the first of its kind in compensating for delays in differential-drive telepresence robot navigation.

5.2 State estimation algorithms

Although several methods and algorithms were proposed to address the time delay problem in the telepresence (also *Tele-operated*, see Section 2.1 for details of sub-fields) systems, it is still an open issue that needs to be addressed. The presence of time delay causes instability in the system and poor performance of the robot navigation. As the mobile robot is controlled by a human operator through a communication network in a remote site, the human operator should know the robot's state to control the robot smoothly. If the time delays are not compensated to estimate the robot's position correctly in the remote site, the human operator may cause an accident by crashing obstacles because of the robot's position which the operator inaccurately recognised. Therefore, we designed a non-linear state estimation computational model to estimate the robot's true state by incorporating a new approach for delay compensation.

The state of a robot is a set of position, orientation and velocity, which is the robot's motion over time. This includes the estimation of the state of the robot's kinematic system by combining knowledge from a priori information and sensor measurements. State estimation in dynamical systems is crucial in real-world applications as the true state is unknown and sensors have limited precision, therefore, provide only a sequence of uncertain noisy measurements. It is to be noted that the measurement in the real world comes from various sensors, *e.g.,* ultrasonic, beacon, camera etc. which can be noisy depending on the environment and robot movement.

Commonly used state estimation methods in robot navigation to stabilize non-linear delayed systems include filtering methods [127, 142] *e.g.,* Extended Kalman Filter, Unscented Kalman Filter, Particle filter or Sigma-Point Kalman Filter *etc.* Among them, the Extended Kalman Filter (EKF) is the most widely used algorithm to ac-

quire an estimate of the true robot state from noisy sensor measurements [143].

However, when a filtering processor is connected to a sensor through a network, there is a fundamental communication time. Moreover, if raw sensor data require post-processing, in order to update the state of the dynamical system, additional post-processing time is needed, resulting in a delay between the acquisition of a measurement and its availability to the filter.

5.2.1 Kalman filter

Before we describe the EKF, it might be worth revisiting the Kalman filter in the context of state estimation. Kalman filter at its core is a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean of the squared error. The filter is very powerful in estimations of past, present, and even future states, and it can do so even when the precise nature of the modelled system is unknown. The Kalman filter (KF) is a method based on recursive Bayesian filtering where the noise in your system is assumed Gaussian. The Kalman Filter estimates the state $x \in \mathscr{R}^n$ of a discrete-time controlled process that is governed by a linear stochastic difference equation.

The Extended Kalman Filter (EKF) is an extension of the classic Kalman Filter for non-linear systems where non-linearity is approximated using the first or second-order derivative. As an example, if the states in a system are characterized by multimodal distribution one should use EKF instead of KF.

5.2.2 Extended Kalman Filter

The Extended Kalman Filter is a set of mathematical equations that provides an efficient computational means to estimate the state of a non-linear process. It also supports estimations of past, present, and even future states as the Kalman filter, and it can do so even when the precise nature of the modelled system is unknown. The advantage of the EKF over other non-linear filtering methods is its

relative simplicity compared to its performance.

The process is governed by the non-linear stochastic difference equation assuming the process has a state vector $x \in \mathscr{R}^n$

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}), \tag{5.1}$$

The measurement equation with $z \in \mathscr{R}^m$

$$z_k = h(x_k, v_k), \tag{5.2}$$

Here the non-linear function $f(\cdot)$ in the difference Equation (5.1) relates the state at the previous time step $k-1$ to the state at the current time step k. The non-linear function $h(\cdot)$ in the measurement Equation (5.2) relates the state x_k to the measurement z_k.

x_k represents the current state vector including the previous state x_{k-1}, an control input u_{k-1} and the process noise w_{k-1}.

z_k represents the measurement state vector including the state x_k and the measurement noise v_k.

The random variables w_k and v_k represent the process and measurement noise respectively. They are assumed to be independent of each other, white, with zero mean Gaussian distributions, being Q and R the process and measurement noise co-variance, respectively:

$$p(w) \sim N(0, Q)$$
$$p(v) \sim N(0, R)$$

The state and measurement vector without any type of noise variable at each time step,

$$\tilde{x}_k = f(\hat{x}_{k-1}, u_{k-1}, 0) \tag{5.3}$$

and

$$\tilde{z}_k = h(\tilde{x}_k, 0), \tag{5.4}$$

Where $\hat{x}_k$ is a posteriori estimate of the state at step k.

5.2.2.1 The computational Origins of the Filter

To estimate a process with a non-linear difference and measurement relationships, new governing equations that linearise an estimate about Equation (5.3) and Equation (5.4),

$$x_k \approx \tilde{x}_k + A(x_{k-1} - \hat{x}_{k-1}) + W w_{k-1}, \tag{5.5}$$

$$z_k \approx \tilde{z}_k + H(x_k - \tilde{x}_k) + V v_k, \tag{5.6}$$

where,

- $\tilde{x}_k$ and $\tilde{z}_k$ represents approximate state and measurement vectors,
- A is the Jacobian matrix of partial derivatives of $f(\cdot)$ with respect to x:

$$A_{[i,j]} = \frac{\mathrm{d}f_{[i]}}{\mathrm{d}x_{[j]}} \left(\hat{x}_{k-1}, u_{k-1}, 0\right),$$

- W is the Jacobian matrix of partial derivatives of $f(\cdot)$ with respect to w:

$$W_{[i,j]} = \frac{\mathrm{d}f_{[i]}}{\mathrm{d}w_{[j]}} \left(\hat{x}_{k-1}, u_{k-1}, 0\right),$$

- H is the Jacobian matrix of partial derivatives of $h(\cdot)$ with respect to x:

$$H_{[i,j]} = \frac{\mathrm{d}h_{[i]}}{\mathrm{d}x_{[j]}} \left(\tilde{x}_k, 0\right),$$

- V is the Jacobian matrix of partial derivatives of $h(\cdot)$ with respect to v:

$$V_{[i,j]} = \frac{\mathrm{d}h_{[i]}}{\mathrm{d}v_{[j]}}\left(\tilde{x}_k, 0\right).$$

Now the notation for the predicted error and the measurement residuals are shown in Equation (5.7) and Equation (5.8), respectively.

$$\tilde{e}_{x_k} = x_k - \tilde{x}_k, \tag{5.7}$$

$$\tilde{e}_{z_k} = z_k - \tilde{z}_k. \tag{5.8}$$

In practice we have no access to x_k in Equation (5.7), as it is the actual state vector, i.e. the quantity is trying to estimate. On the other hand, we have access to z_k in Equation (5.8), as it is the actual measurement that is used to estimate x_k.

Using Equation (5.7) and Equation (5.8) we can write governing equations for an error process as

$$\tilde{e}_{x_k} \approx A(x_k - \hat{x}_k) + \varepsilon_k, \tag{5.9}$$

$$\tilde{e}_{z_k} \approx H\tilde{e}_{x_k} + \eta_k, \tag{5.10}$$

where ε_k and η_k represent new independent random variables having zero mean and covariance matrices WQW^T and VRV^T with Q and R.

From Equation (5.8)and Equation (5.9) we can estimate,

$$\hat{x}_k = \tilde{x}_k + \hat{e}_k. \tag{5.11}$$

$\hat{e}_k$ will help to obtain the posterior state estimates for the original non-linear process.

The random variables of Equation (5.9) and Equation (5.10) have approximately the following probability distributions:

$$p(\tilde{e}_{x_k}) \sim N(0, E[\tilde{e}_{x_k}\tilde{e}_{x_k}^T])$$
$$p(\varepsilon_k) \sim N(0, WQ_kW^T)$$
$$p(\eta_k) \sim N(0, VR_kV^T).$$

Given these approximation and letting the predicted value of $\hat{e}_k$ be zero, the Kalman Filter equation used to estimate $\hat{e}_k$ is

$$\hat{e}_k = K_k\tilde{e}_{z_k}. \tag{5.12}$$

By substituting Equation (5.12) back into Equation (5.11) and making use of Equation (5.8) we get,

$$\begin{aligned} \hat{x}_k &= \tilde{x}_k + K_k\tilde{e}_{z_k} \\ &= \tilde{x}_k + K_k(z_k - \tilde{z}_k) \end{aligned} \tag{5.13}$$

A complete set of Extended Kalman Filter estimation equations can be expressed in two parts A. Time Prediction update equations and B. Measurement update equations as described below.

A. Time Prediction update equations

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0) \tag{5.14}$$
$$P_k^- = A_kP_{k-1}A_k^T + W_kQ_{k-1}W_k^T \tag{5.15}$$

As with the basic discrete Kalman filter, the time prediction update equations represent the state and co-variance estimates from the time step k to the time step $k+1$. The Equation (5.14) comes from Equation (5.3), A_k and W_k are the process Jacobians at step k, and Q_k is the process noise co-variance at step k.

B. Measurement update equations

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1} \tag{5.16}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-, 0)) \tag{5.17}$$

$$P_k = (I - K_k H_k) P_k^- \tag{5.18}$$

As with the basic discrete Kalman filter, the measurement update equations correct the state and covariance estimates with the measurement. The Equation (5.17) comes from Equation (5.4), H_k and V_k are the measurement Jacobians at step k, and $R_k$is the measurement noise co-variance at step k [50].

An important feature of the EKF is that the Jacobian H_k in the equation for the Kalman gain K_k serves to correctly propagate only the relevant component of the measurement information.

5.2.3 Implementation on differential drive telepresence robot

Now we take the filter equations for differential drive (telepresence) robots and described them below.

5.2.3.1 The Prediction Model

In the prediction model, the current state (robot pose) is estimated from the previous state. The state vector of the robot pose with respect to a global coordinate frame is

$$x(k) = [x(k), y(k), z(k), \theta(k)]^T$$

Each state vector $x(k)$ includes a co-variance matrix $P(k)$. Control input $u(k)$ is computed using the velocities of the left and right wheels. The state transition function at the next time step is

$$x(k|k-1) = f(x(x-1|x-1), u(k-1), w(k-1))$$

where $w(k)$ represents unpredictable noise. The noise is assumed to be Gaussian with zero mean, $(v(k) = 0)$, and co-variance $Q(k-1)$. From Equation (4.1) to Equation (4.4) the state transition function becomes,

$$f(x(k-1), u(k-1), 0) = \begin{bmatrix} x(k-1) + \Delta V(k-1) * \cos(\theta(k-1) + \Delta\theta(k-1)) \\ y(k-1) + \Delta V(k-1) * \sin(\theta(k-1) + \Delta\theta(k-1)) \\ z(k-1) \\ \theta(k-1) + \Delta\theta(k-1) \end{bmatrix} \tag{5.19}$$

The source of uncertainty is the uncertainty in the angular and linear velocities. We calculate the uncertainty by partial differentiation of Equation (5.19) with respect to $\Delta\theta(k-1)$ and $\Delta V(k-1)$, which gives the following Jacobian,

$$W = \begin{bmatrix} -\Delta V(k-1) * \sin(\theta(k-1) + \Delta\theta(k-1)) & \cos(\theta(k-1) + \Delta\theta(k-1)) \\ \Delta V(k-1) * \cos(\theta(k-1) + \Delta\theta(k-1)) & \sin(\theta(k-1) + \Delta\theta(k-1)) \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{5.20}$$

Another source of uncertainty is the uncertainty in position and orientation. To compute it we make an another partial differentiation of Equation (5.19) with respect to $x(k-1)$, $y(k-1)$, $z(k-1)$ and $\theta(k-1)$, which gives the following Jacobian.

$$A = \begin{bmatrix} 1 & 0 & 0 & -\Delta V(k-1) * \sin(\theta(k-1) + \Delta\theta(k-1)) \\ 0 & 1 & 0 & \Delta V(k-1) * \cos(\theta(k-1) + \Delta\theta(k-1)) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.21}$$

The process noise covariance matrix $Q(k-1)$ depends on two independent sources of error called angular and linear.

$$Q = \begin{bmatrix} \sigma^2_{\Delta\theta} & 0 \\ 0 & \sigma^2_V \end{bmatrix} \tag{5.22}$$

The complete pre-co-variance matrix becomes:

$$P(k|k-1) = A(k)P(k-1|k-1)A(k)^T + W(k)Q(k-1)W(k)^T \tag{5.23}$$

5.2.3.2 The Measurement Model

The motion sensor data readings are used to improve the mobile robot pose estimation. The distance between an obstacle and the robot measured by the i^{th} sensor is computed by the following measurement function

$$h_i(k|k-1) = \sqrt{(x_i - x(k|k-1))^2 + (y_i - y(k|k-1))^2}. \tag{5.24}$$

If the distance between the obstacle and the robot is not known, the measurement function can be computed as follows,

$$h(k|k-1) = \begin{bmatrix} x_i + v(x) \\ y_i + v(y) \\ z_i + v(z) \\ \theta_i + v(\theta) \end{bmatrix} \tag{5.25}$$

The sensor measurement vector z_k is a stack of z_i, k measurements.

$$z_i(k) = h_i(x(k|k-1)) + w_i(k), \tag{5.26}$$

where w_i represents the measurement noise assuming Gaussian with zero mean and variance.

The Jacobian is computed by partial differentiation of Equation (5.24) with respect to $x(k)$, $y(k)$, $z(k)$ and $\theta(k)$,

$$H = \frac{1}{\sqrt{(x_i - x(k|k-1))^2 + (y_i - y(k|k-1))^2}} \begin{bmatrix} x(k|k-1) - x_i \\ y(k|k-1) - y_i \\ 0 \end{bmatrix}^T \tag{5.27}$$

Partial differentiation of Equation (5.25) will be as follows:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.28}$$

Another Jacobian is an identity matrix with size N,

$$V = I(N) \tag{5.29}$$

The measurement co-variance matrix R is a diagonal matrix with the measurement noise variance value on the diagonal.

$$R = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 \\ 0 & 0 & \sigma_z^2 & 0 \\ 0 & 0 & 0 & \sigma_\theta^2 \end{bmatrix} \tag{5.30}$$

$K(k)$ is the optimal Kalman gain computed as follows:

$$K(k) = P(k|k-1)H(k)^T(H(k)P(k|k-1)H(k)^T + V(k)R(k)V(k)^T)^{-1} \tag{5.31}$$

The state estimation and its co-variance in time step k are computed as follows:

$$x(k) = x(k|k-1) + K(k)(z(k) - h(k)) \tag{5.32}$$

$$P(k) = (I - K(k)H(k))P(k|k-1) \tag{5.33}$$

5.3 The algorithm

When sensor measurements are delayed, the current state could not be directly corrected using the current measurement, since a delayed sensor measurement was actually carrying information about a past measurement state. Here, $x(k)$ could not be corrected directly because the measurement values depend on the past measurement state $x(k-\tau)$. Therefore, the past measurement state corresponding to a delayed measurement needed to be determined before using the delayed measurement during the state estimation. The current state also needed to be corrected after correcting the appropriate past state.

5.3.1 Augmented State Extended Kalman Filter (AS-EKF)

In this research, we have used augmentation of states with EKF filter [126] for delay-compensated state estimation of telepresence robots with considering uncertain delayed sensor measurements as depicted in Figure 5.1. We augmented the present and past states into several augmented state vectors to estimate the robot's true position. The current measurement state which contains information on the past measurement states directly corrects the augmented state vectors. In this way, in a delayed system, we determined the corresponding past state in the augmented state vector. After that, the past state was updated using the delayed measurement data and the current state was simultaneously corrected in the augmented state vector. It is to be noted that firstly the algorithm considers a certain time delay which is then extended to compensate for uncertain delays using PDFs.

For a one-time step delay, the prediction equation was modified as

$$\begin{bmatrix} x_{k+1} \\ x_k \end{bmatrix} = \begin{bmatrix} f(x_k, u_k, w_k) \\ x_k \end{bmatrix}$$

where, $\begin{bmatrix} x_{k+1}^T & x_k^T \end{bmatrix}^T$ was the augmented state vector. The measurement equation

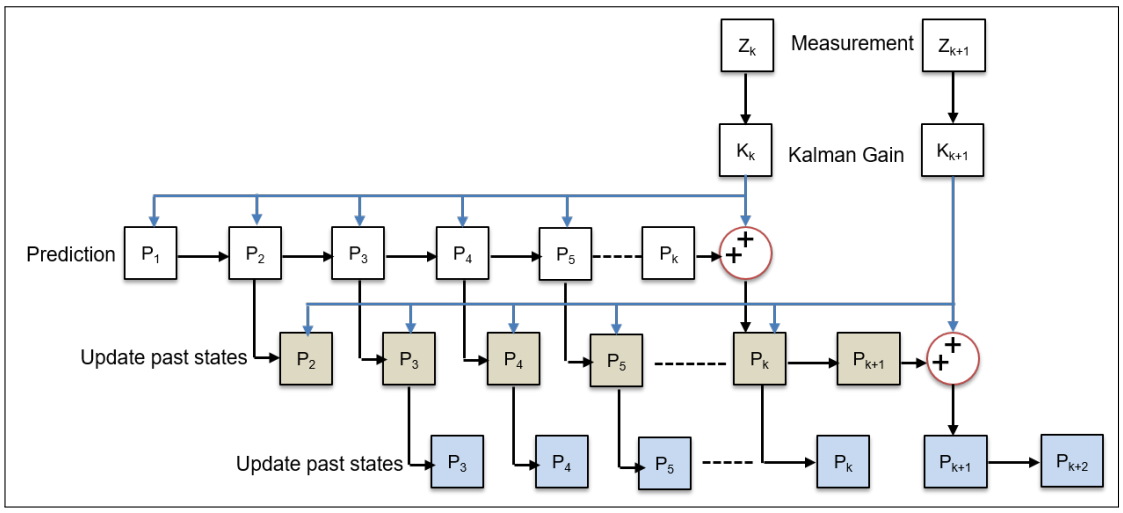


Figure 5.1: Flow diagram of the proposed algorithm. The diagram shows the sequence of steps involved in the AS-EKF model used in this work.

was

$$z_k = h\left(\begin{bmatrix} 0 & I \end{bmatrix}\begin{bmatrix} x_{k+1} \\ x_k \end{bmatrix}, v_k\right)$$

where, I was the identity matrix, the current measurement z_k was used to update $\begin{bmatrix} x_{k+1}^T & x_k^T \end{bmatrix}^T$.

For multi-step delays, the prediction equation defined as

$$\begin{aligned} \mathcal{X}_{(k+1)} &= \begin{bmatrix} f(x_k, u_k) \\ \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & \ddots & 0 & \vdots \\ 0 & 0 & I & 0 \end{bmatrix} X_k \end{bmatrix} + \begin{bmatrix} w_k \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ &\equiv f(\mathcal{X}_k, \mathcal{U}_k, \mathcal{W}_k) \end{aligned} \tag{5.34}$$

where, $\mathcal{X}_{(k)}$ was the augmented state vector defined by $\begin{bmatrix} x_k^T & x_{k-1}^T & \cdots & x_{k-n}^T \end{bmatrix}^T$ and n was the maximum number of delayed time steps. The measurement equation was rewritten as

$$\mathcal{Z}_k = h\left(\begin{bmatrix}\begin{bmatrix}0\\ \vdots\\ I\\ \vdots\\ 0\end{bmatrix}^T & \begin{bmatrix}x_k\\ \vdots\\ x_{k-\tau_k}\\ \vdots\\ x_{k-n}\end{bmatrix}\end{bmatrix}\right) + \begin{bmatrix}0\\ \vdots\\ v_{k-\tau_k}\\ \vdots\\ 0\end{bmatrix} \tag{5.35}$$
$$\equiv h(\mathcal{X}_k, \mathcal{V}_k)$$

where, τ_k represented the time delay, which was less than n, and I was placed at the corresponding time step $k - \tau_k$. If the time delay τ_k is known or certain, the augmented state vector can be estimated recursively via the EKF algorithm.

The EKF algorithm consists of prediction and measurement update stages. In the prediction stage, state prediction was carried out by the prediction (Equation (5.1)). The error covariance was propagated by the Jacobian of the prediction model and the process noise co-variance(Q). The measurement update stage or measurement model was based on the prediction model and the error covariance (Equation (5.2)). The Jacobian of the measurement model and the measurement noise(R) were needed to obtain the Kalman gain (K). The proposed method was implemented in the augmented state vector using the prediction and measurement update stages of the EKF algorithm.

5.3.1.1 Dealing uncertain delays

So far, our model considers certain delays and compensates for the prediction through augmented states. However, in practice, often, delays are unknown (as discussed in Section 4.4), and therefore we extend the AS-EKF to handle uncertain delays. Our hypothesis is that while the delay for each measurement is different, the average delay is measurable by modelling the probability of factors that introduce such delay, *e.g.*, feedback or Internet communication. In this work, the modelling of uncertain delays was done using the PDF in terms of two different

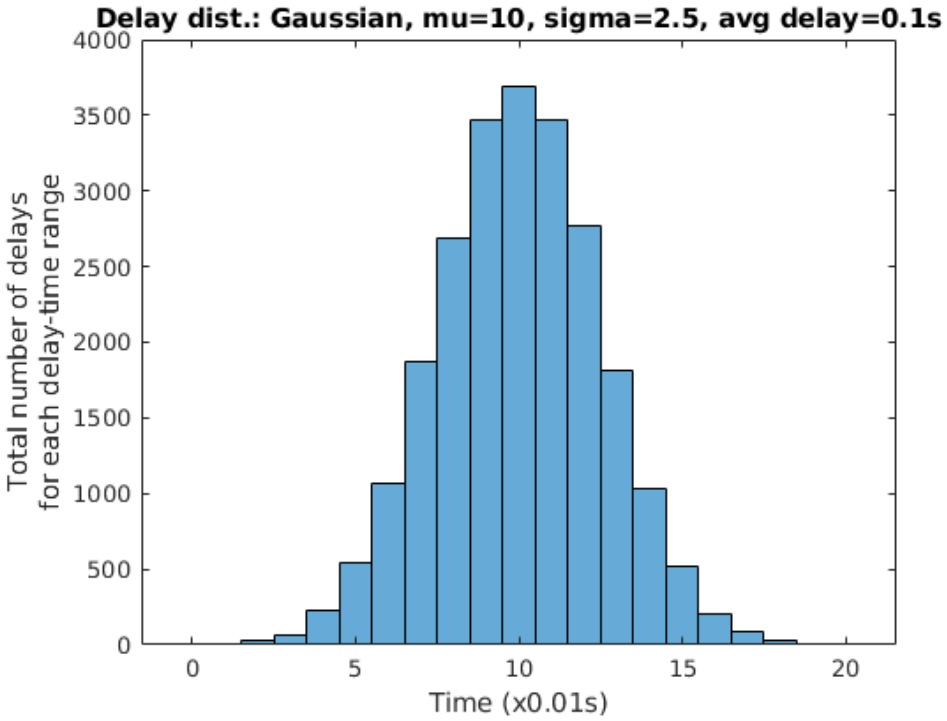


Figure 5.2: Histogram of delay probability density functions for average delay $\tau = 10$ with Gaussian distribution.

delay distributions, *i.e.*, *Gaussian* and *Gamma* (as discussed in Section 4.4.2) to get a consistent state estimator. Examples of such delays, *i.e.*, Gaussian and Gamma are shown in Figure 5.2 and Figure 5.3 (average delay $\tau = 10$).

In extending the proposed algorithm for uncertain delays, we consider such average delay (modelled by PDFs, the peaks in the example figure) as continuous input to the system. This allows us to directly apply the proposed AS-EKF for state estimation in uncertain scenarios. We verify this hypothesis in Section 5.5.2 by simulating various average delays for uncertain time-delay scenarios. The results are promising and assert the fact that the proposed algorithm offers a better and consistent delay-compensated state estimation in both the certain and uncertain delayed system environment.

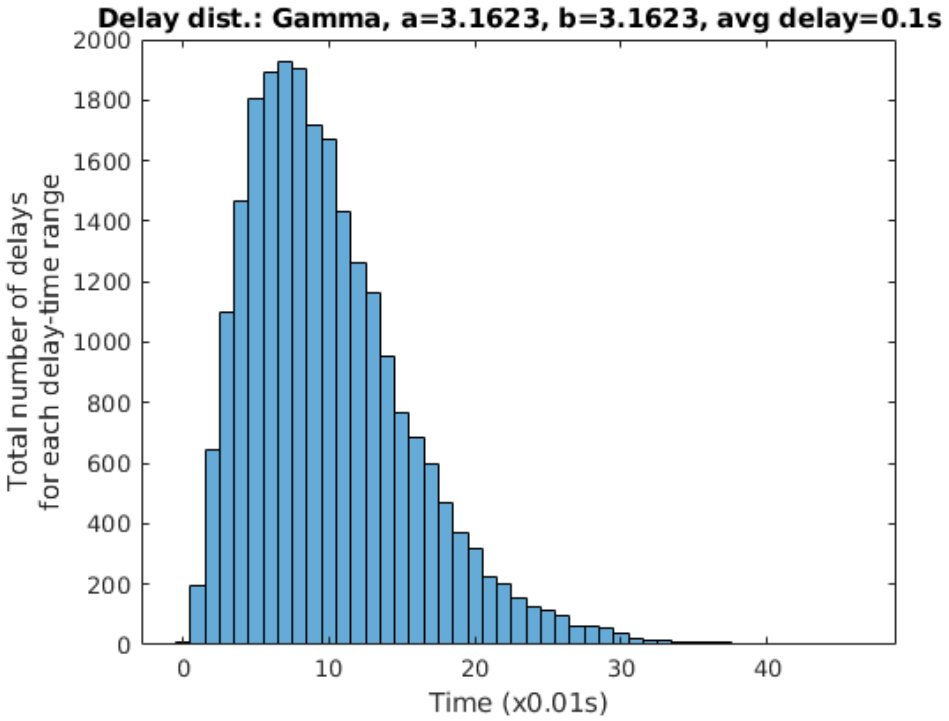


Figure 5.3: Histogram of delay probability density functions for average delay $\tau = 10$ with Gamma distribution.

5.4 Experimental setup

We have evaluated the proposed approach considering delayed robot navigation measurements. Using the proposed framework, a raster scan robot navigation path (*design path*) was created to simulate and study our approach. In this work, we considered various scenarios where the robot might encounter navigation challenges. For example, navigational error in different directional paths (both vertical and horizontal) and, most importantly, during sharp turns in either direction. A raster scan [42] based navigation in fact covers these scenarios and therefore was chosen in this work. The algorithm was written in C++, encapsulating ROS to control the telepresence robot remotely. The real navigation path was monitored and tracked through the VICON motion tracking system. As mentioned previously, VICON has an extremely small error variance, and hence, the VICON output data has been considered as the actual robot path in all our experiments.

We envisage two scenarios, 1) robot navigation in a known or certain delayed environment; and 2) robot navigation in an unknown or uncertain delayed environment. In addition to that, we also considered the measurement for positional data to be noisy. Ideally, to create such scenarios in real life, one would need to arrange a set-up where the local site and remote sites are physically distanced at least in order or hundreds of miles/kilometres so that the physical communication delays are noticeable. In the absence of such a large geographical distance in the lab environment, we simulated the data.

Firstly, the robot velocity was assumed to have white Gaussian noise, and the measurement data was accordingly also delayed by the sensor noise. For this purpose, we have added random position noise to the VICON output data. The noisy positional data is then arranged to insert certain and uncertain delays to simulate various scenarios in this work.

In a real environment scenario, we cannot assume the time delay between sending a control command to the robot and the moment when the received sensor measurement data is entered in the state estimator. We assumed that the measurement time delay was uncertain. In this paper, we applied a state estimation algorithm to obtain the robot's true position calculating and modelling the uncertain time delay as discussed in Section 4.4.2.

5.5 Results and discussions

In order to verify our proposed approach, we initially experimented with an overall simulation followed by the experiments as described in Section 5.4. A raster scan robot path was simulated using MATLAB simulations. Delay was introduced on the measurement values as a unit of time steps. Regular EKF and the proposed delay-compensated approach (AS-EKF) were applied to show the effectiveness of the delay-compensated approach. The results of such simulation are shown in Figure 5.4, where the figure on the top shows the complete path. *True Path* refers to Vicon's observed robot path. The bottom figure is a zoomed version of the se-

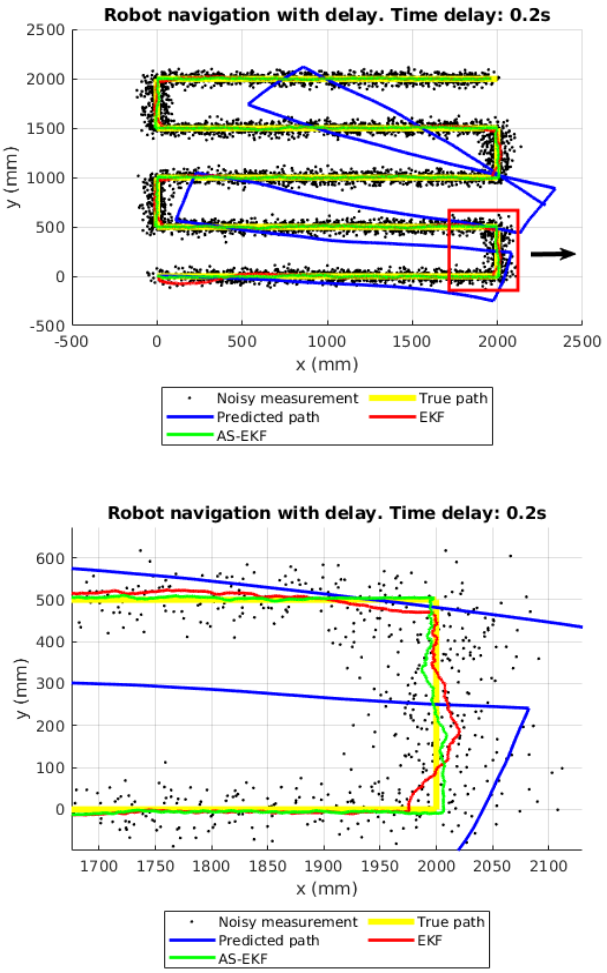


Figure 5.4: Comparison of EKF with AS-EKF estimated robot path with simulated time-delayed measurement data. (*Row 1*) overall robot navigation path; (*Row 2*) zoomed version of the selected path that shows the effectiveness of AS-EKF over regular EKF. *True Path* refers to Vicon's observed robot path.

lected area, which clearly shows regular EKF was unable to handle delay when the robot changed its direction. On the contrary, as expected, the AS-EKF compensated for the delay and closely followed the true robot path. On verification of our approach to the simulations, we performed detailed experiments on the Beam plus telepresence robot using the experiential framework (described in the following subsections).

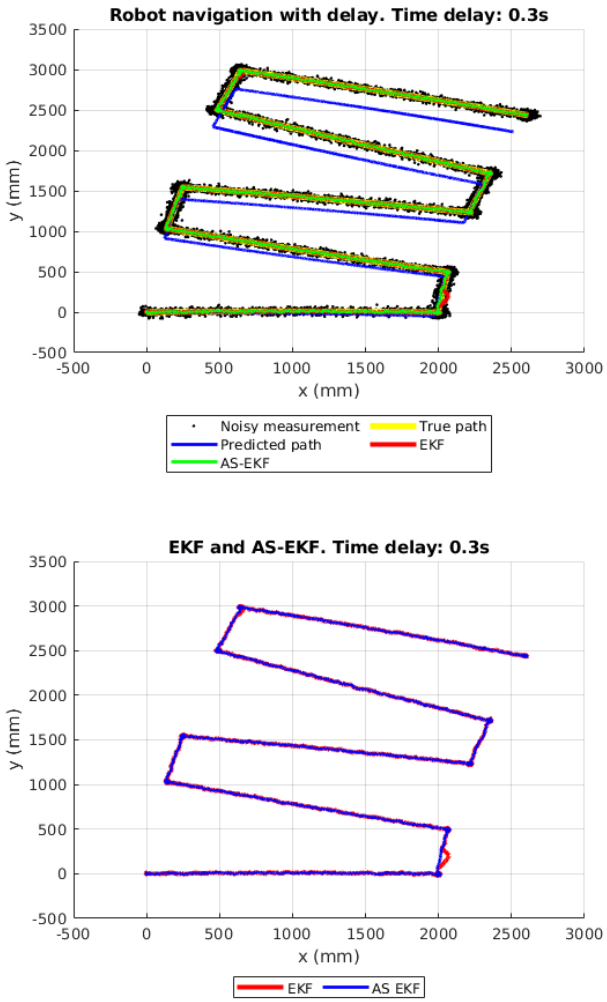


Figure 5.5: Comparison of EKF with AS-EKF estimated robot path with time-delayed measurement data. (*Row 1*) Overall comparison; (*Row 2*) Comparison between EKF and AS-EKF.

5.5.1 Scenario I: Certain time delay

Considering certain time delays in the sensor measurement, we applied both EKF and delay-compensated AS-EKF algorithms as discussed in Section 5.3. To gain an in-depth insight, we have introduced a number of delay in time steps ($\tau = [10, 15, 20, 25]$) corresponding to delay in the equivalent of $[0.1, 0.15, 0.2, 0.25]$ seconds, respectively. The delay parameters are shown in Table 4.1. Results for the corrected navigation path (for $\tau = 30$) are shown in Figure 5.5. The results show

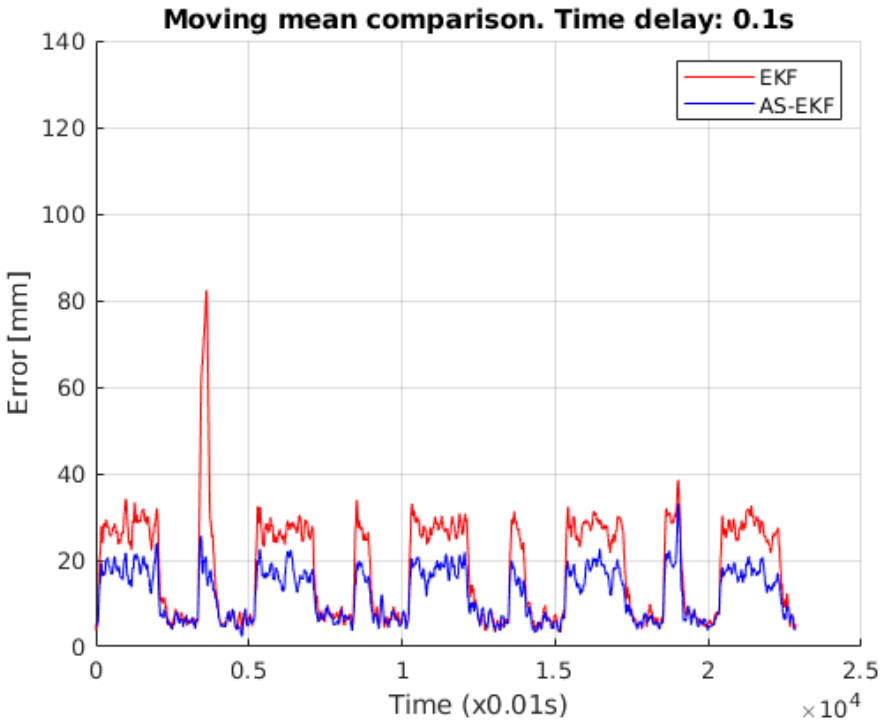
that the AS-EKF algorithm is reducing the linearisation error and compensating for the time delay more precisely. Instead of EKF estimated path, the AS-EKF estimated path is more close to the robot control path.

In order to capture performance for delay compensation, we calculated the error in terms of root mean square error (RMSE) between the VICON measurement (absolute robot path) and estimated path by EKF and AS-EKF, respectively, with respect to measurement time steps. The results for complete paths are shown in Figure 5.6, and Figure 5.7 and average RMSE errors for entire paths are reported in Table 5.1.

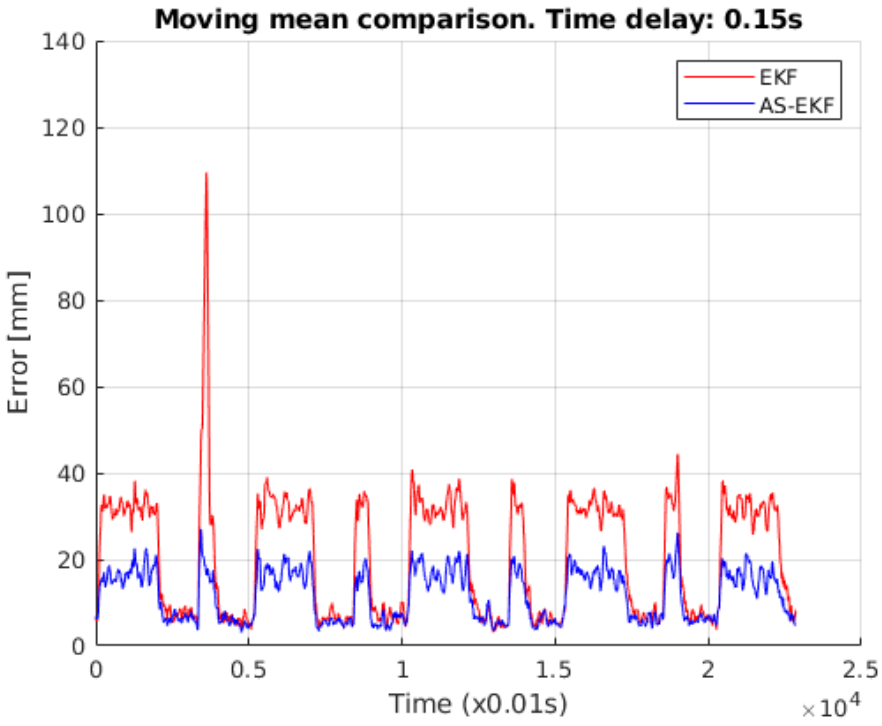
Time	RMSE		Improvement
delay	EKF	AS-EKF	(%)
0.10s	17.60	11.88	32.50
0.15s	20.57	12.01	41.63
0.20s	23.38	11.94	48.92
0.25s	26.17	11.96	54.28

Table 5.1: RMSE error comparison for the certain time delay.

The results show that with the increase of the number of delayed steps in the measurement data, the performance of the EKF algorithm proportionally decreases as the time delay in the measurement data degrades the state estimation accuracy of the algorithm. AS-EKF significantly reduces the error and is maintained at the same level by compensating for the error introduced by the delay. In our experiments, we have achieved improvements of 33% to 54% when considering the delay compensated AS-EKF as opposed to regular EKF.

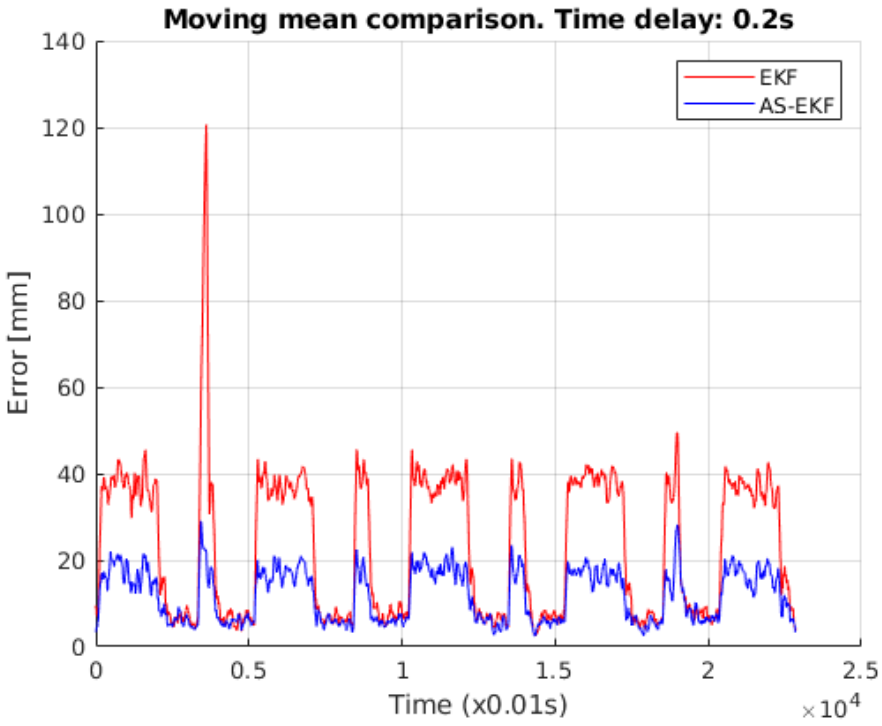


(a) $\tau = 10$.

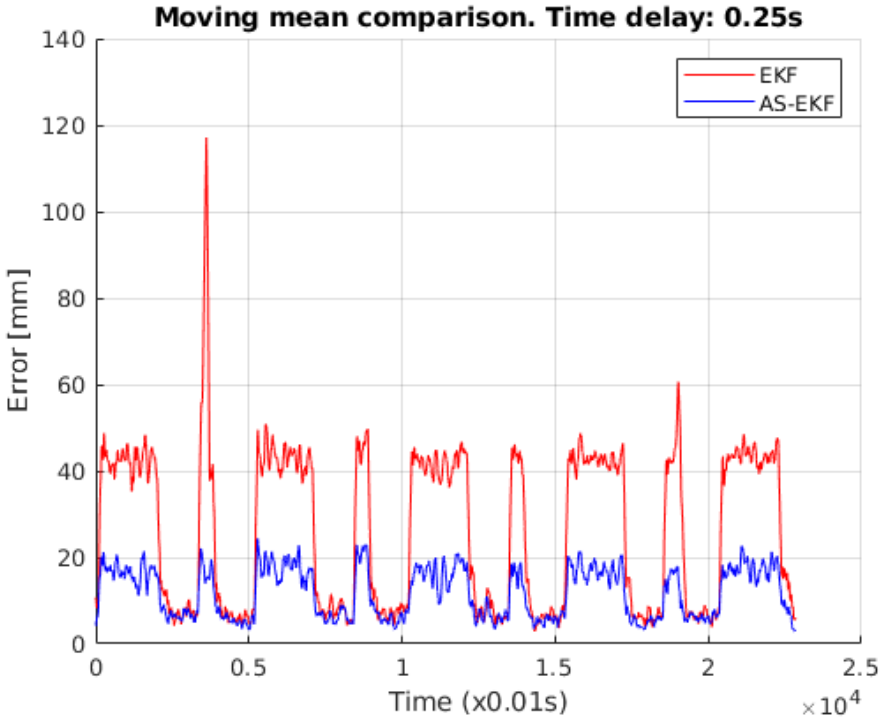


(b) $\tau = 15$.

Figure 5.6: RMSE error comparison for certain time delay $\tau = 10$ and $\tau = 15$ respectively. The red line and blue line represent the RMSE error of EKF and AS-EKF estimation.

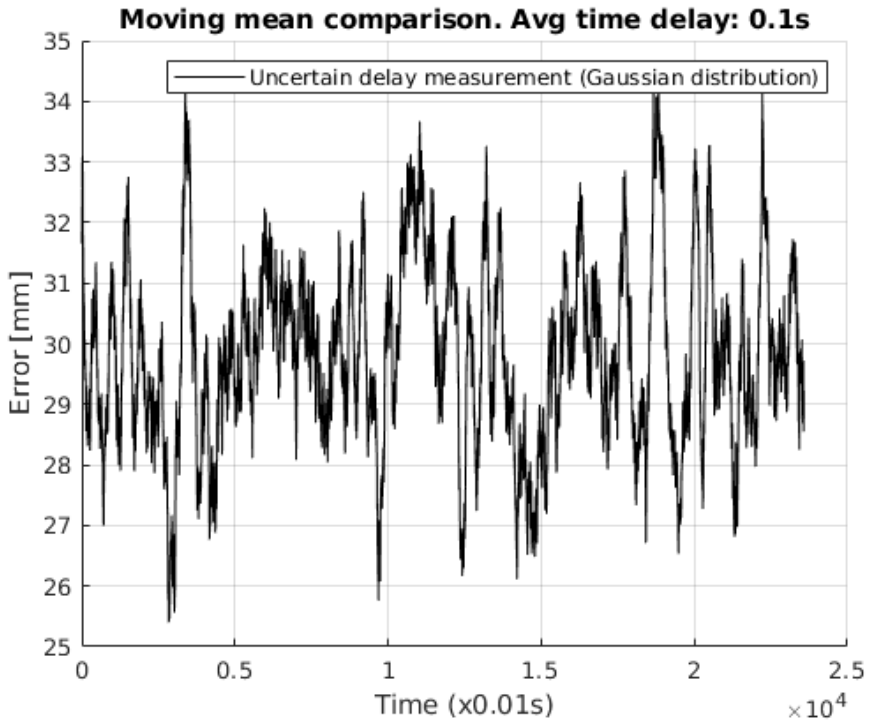


(c) $\tau = 20$.

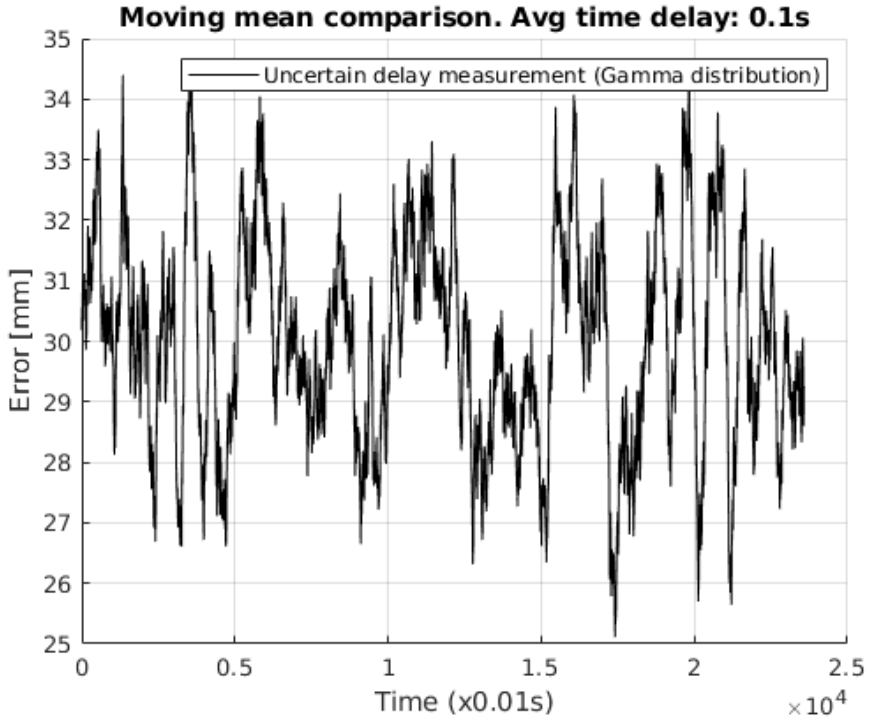


(d) $\tau = 25$.

Figure 5.7: RMSE error comparison for certain time delay $\tau = 20$ and $\tau = 25$ respectively. The red line and blue line represent the RMSE error of EKF and AS-EKF estimation.



(a)



(b)

Figure 5.8: Positional error due to uncertain delay with average delay $\tau = 10$: (a) and (b) represent delay modelled using Gaussian and Gamma distributions, respectively.

5.5.2 Scenario II: Uncertain time delay

As mentioned in earlier sections, certain time delays are rare in real-life environments. Therefore, we consider scenarios with uncertain time delays. As the previous study shows uncertain delays can be modelled using the probability density functions (PDF), such as Gaussian and Gamma distributions, we considered both distributions in simulating delays within measurement values. Random delays with averages similar to the certain time delays are introduced in respective distributions and the distribution parameters were calculated accordingly. The distribution parameters are reported in Table 4.1 and example plots of positional error due to uncertain delay with respect to time steps for both Gaussian and Gamma distributions are shown in Figure 5.8 (average time delay $\tau = 10$). As described in Section 5.3.1 (*Dealing uncertain delays*), we considered the average delays as input to the system and estimated states using the proposed AS-EKF algorithm. Similar to certain delays the results of AS-EKF for uncertain delays were compared against state estimation using EKF only that does not consider delay compensation.

Time	RMSE		Improvement
delay	EKF	AS-EKF	(%)
0.10s	19.11	13.22	30.79
0.15s	31.54	13.08	39.29
0.20s	23.62	12.81	45.77
0.25s	26.67	13.07	50.99

Table 5.2: RMSE error comparison for uncertain time delays with Gaussian distribution.

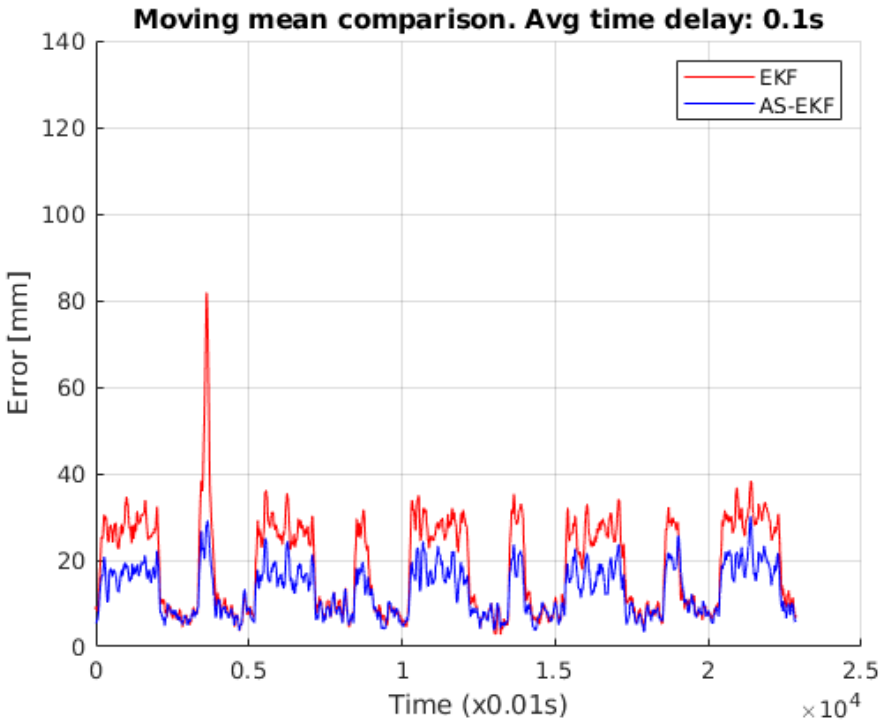
The results for the delay with Gaussian and Gamma distributions are shown in Figure 5.9, Figure 5.10 Figure 5.11 and Figure 5.12, respectively. Similar to the certain delay, we calculated the error between the estimated path and VICON measurements (absolute robot path) and compared for EKF without considering delay compensation and AS-EKF that compensated the delay by assuming the average

Average delay	RMSE		Improvement
in time steps	EKF	AS-EKF	(%)
0.10s	18.53	12.85	30.67
0.15s	21.16	12.78	39.61
0.20s	24.09	12.71	47.24
0.25s	26.60	12.65	52.46

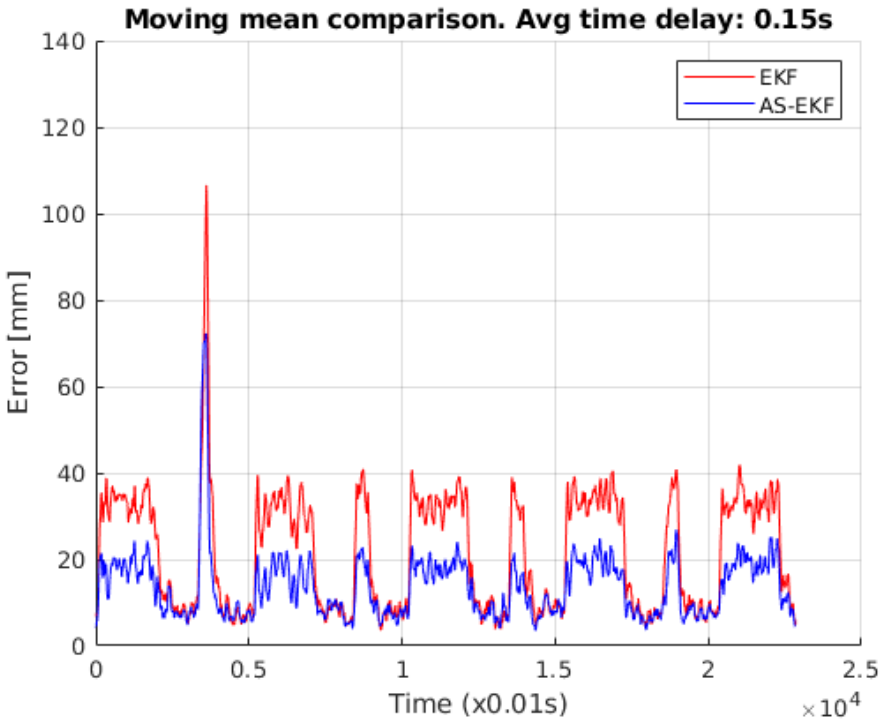
Table 5.3: RMSE error comparison for uncertain time delay with Gamma distribution.

delay in these scenarios. The RMSE error for Gaussian and Gamma distributed delays are shown in Table 5.2 and Table 5.3, respectively. In both cases, we have observed more than 50% improvements.

Finally, we have compared the estimation error for various scenarios, *e.g.,* certain delays and uncertain delays with Gaussian and Gamma distributions. The results are shown in Figure 5.13 for $\tau = [10, 20]$. We have also compared the RMSE errors and reported them in Figure 5.14.

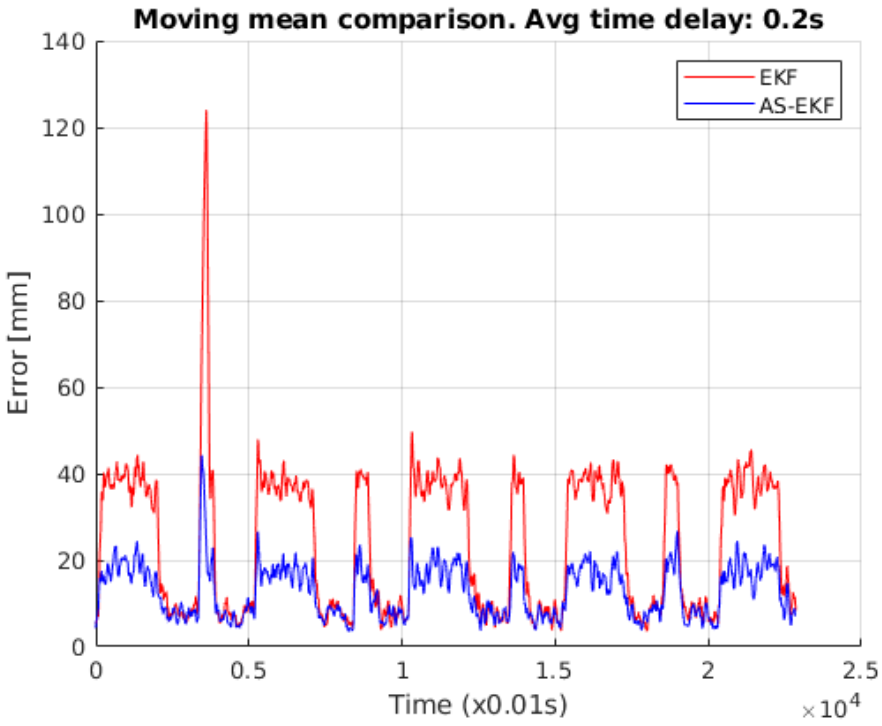


(a) Average $\tau = 10$.

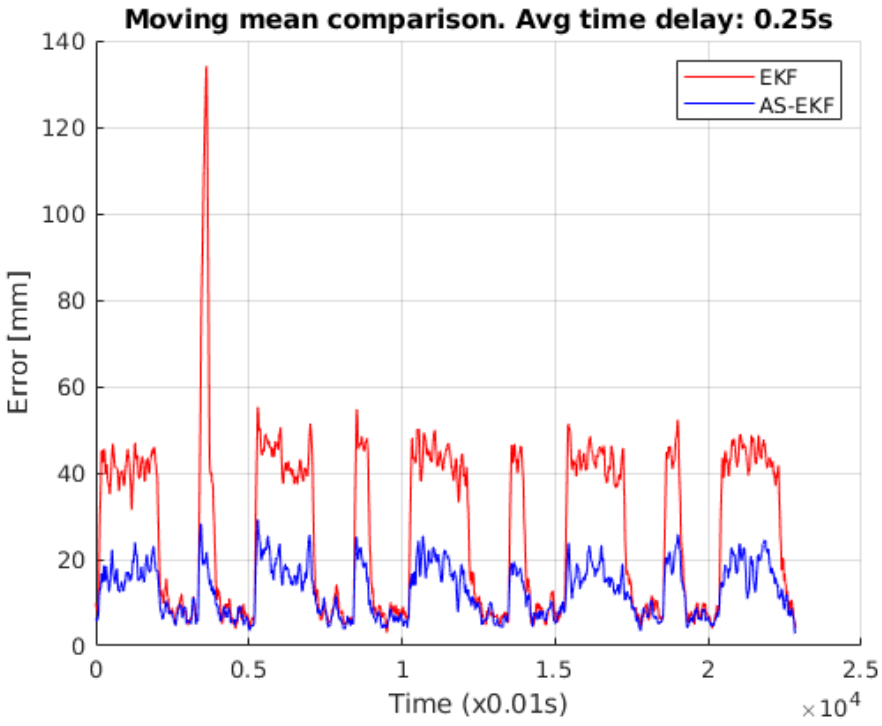


(b) Average $\tau = 15$.

Figure 5.9: RMSE error comparison for uncertain time delay with Gaussian distribution. The red line and blue line represent the RMSE error of EKF and AS-EKF estimation, respectively.

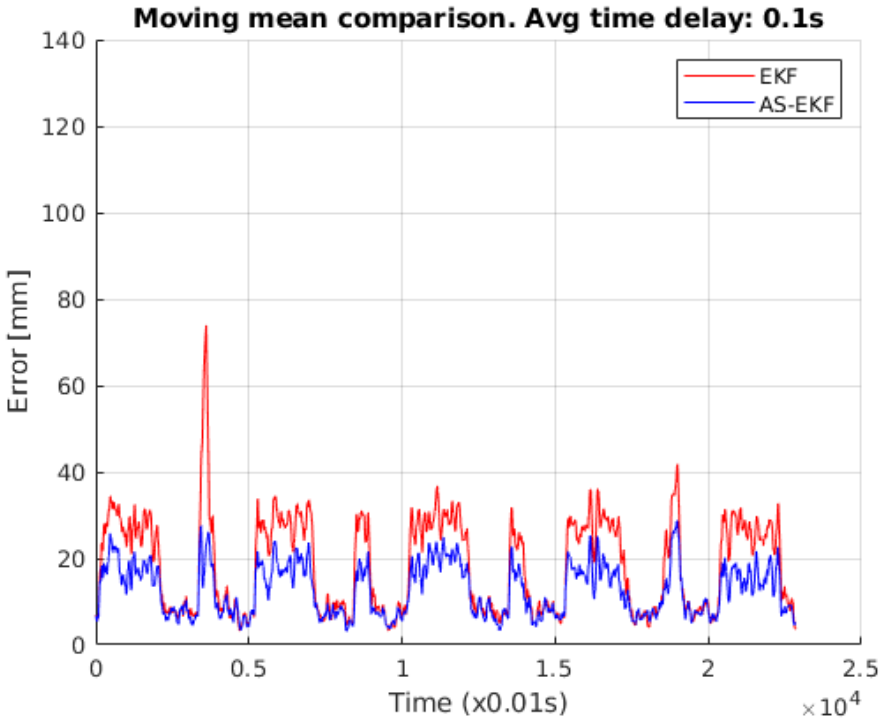


(c) Average $\tau = 20$.

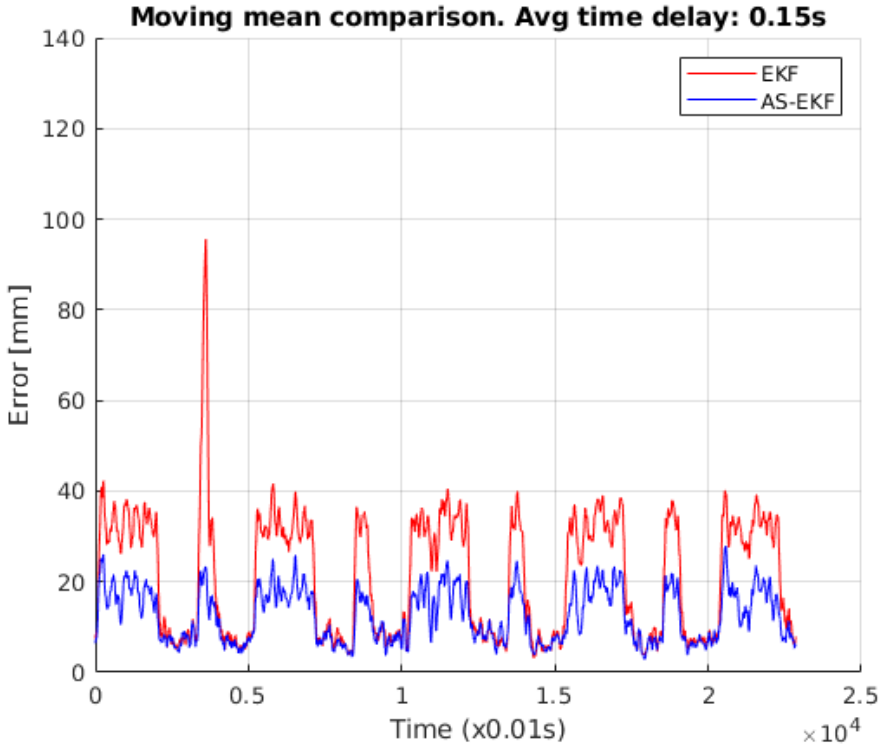


(d) Average $\tau = 25$.

Figure 5.10: RMSE error comparison for uncertain time-delay with Gaussian distribution. The red line and blue line represent the RMSE error of EKF and AS-EKF estimation, respectively.

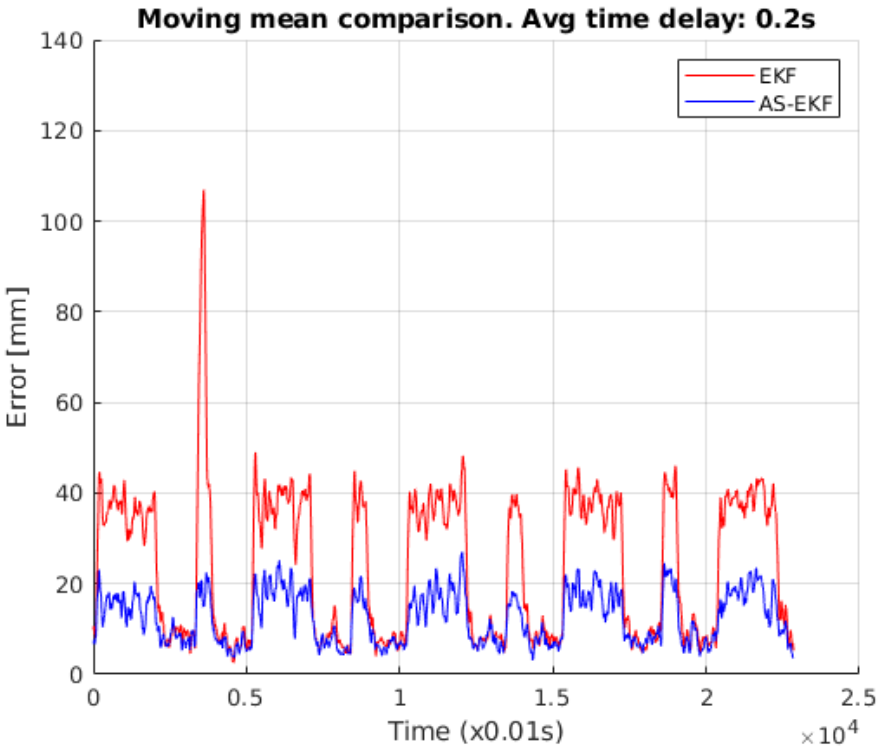


(a) Average $\tau = 10$.

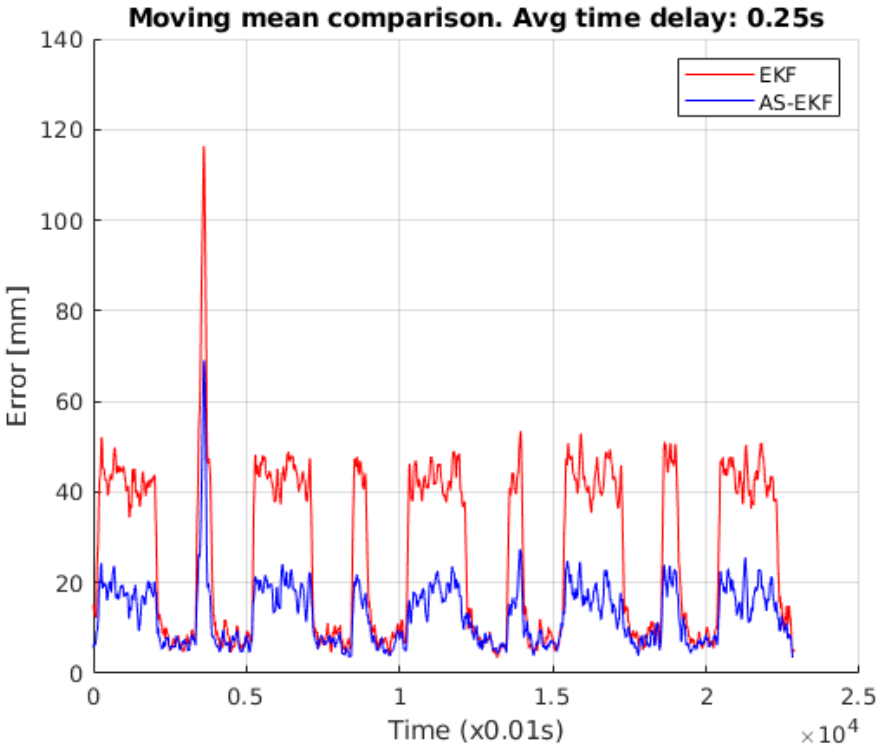


(b) Average $\tau = 15$.

Figure 5.11: RMSE error comparison for uncertain time-delay with Gamma distribution. The red line and blue line represent the RMSE error of EKF and AS-EKF estimation, respectively.

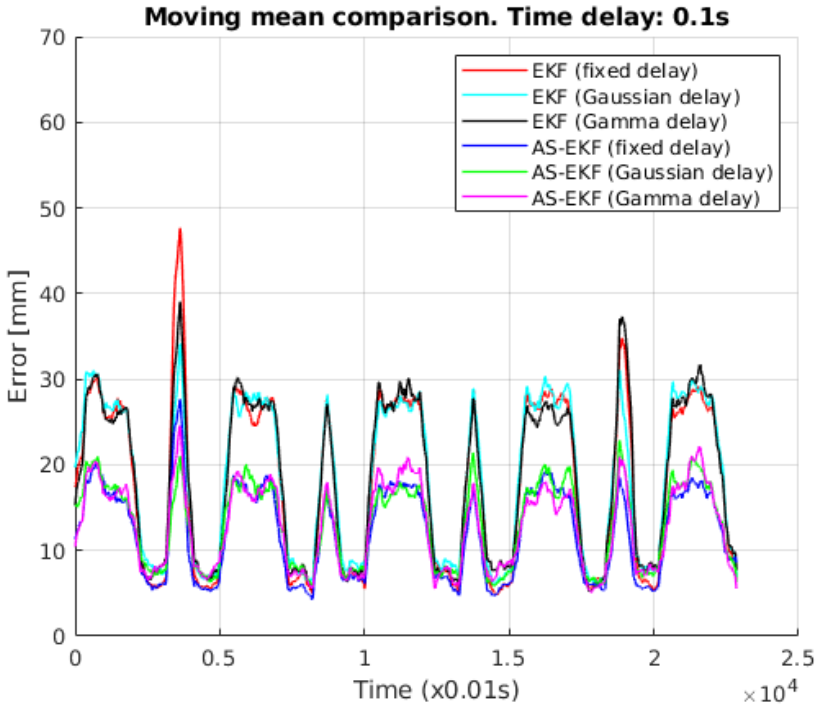


(c) Average $\tau = 20$.

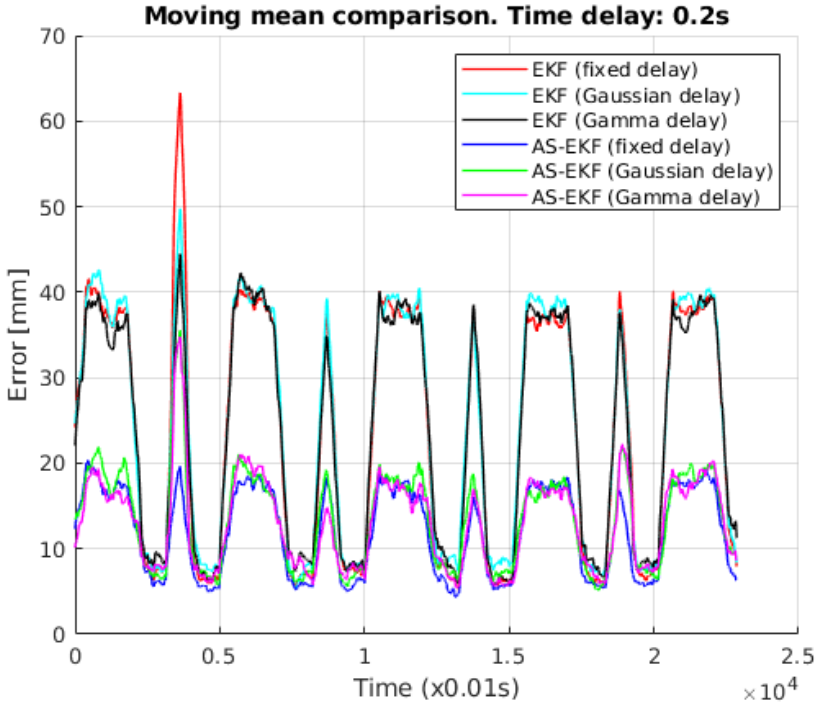


(d) Average $\tau = 25$.

Figure 5.12: RMSE error comparison for uncertain time-delay with Gamma distribution. The red line and blue line represent the RMSE error of EKF and AS-EKF estimation, respectively.



(a)



(b)

Figure 5.13: RMSE error comparison of EKF and AS-EKF estimation. (a) The red, cyan and black lines represent the RMSE error of EKF estimation for the certain delay and uncertain delay with Gaussian distribution and Gamma distribution, respectively. (b) The blue, green and magenta lines represent the RMSE error of AS-EKF estimation for the certain delay, uncertain delay with Gaussian distribution and Gamma distribution, respectively. [106]

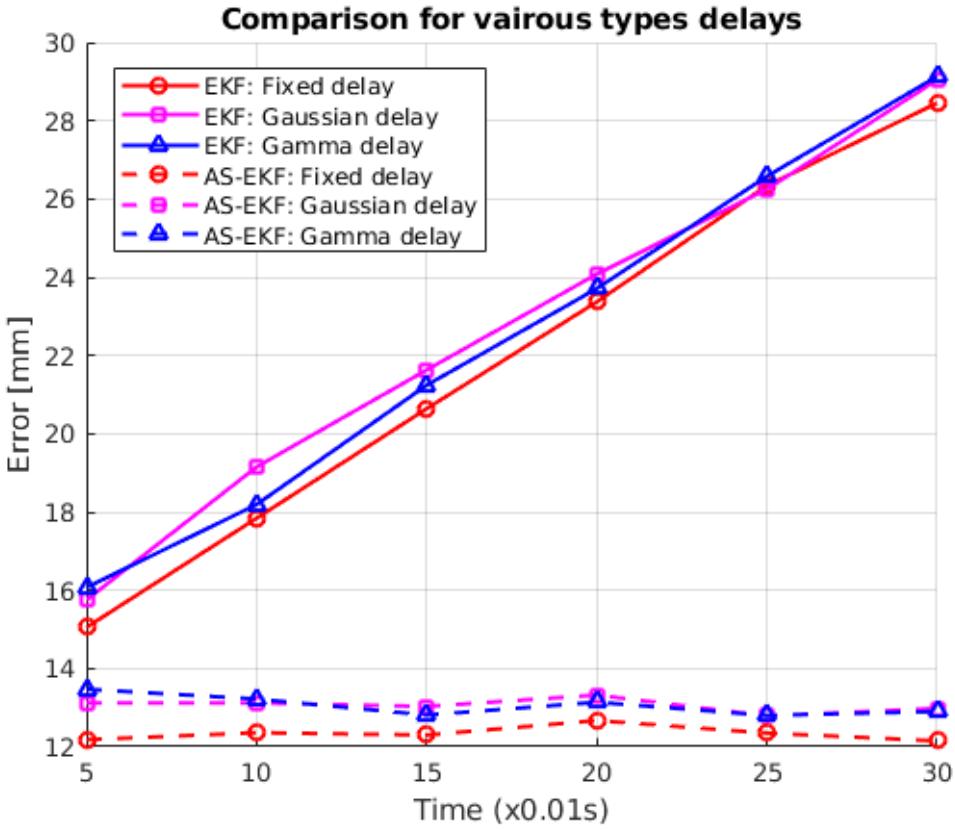


Figure 5.14: RMSE error comparison of EKF and AS-EKF for the certain delay and uncertain time-delay with Gaussian and Gamma distribution respectively. The comparison is made for different delayed time steps. The red, magenta and blue line represent the RMSE error of EKF estimation and the dotted line represent AS-EKF estimation, respectively.

5.6 Chapter summary

In this chapter, a delay-compensated state estimation approach for a telepresence system with uncertain delayed navigation measurement was presented. EKF combined with an augmented state model was successfully executed, estimating the actual robot position and modelling certain and uncertain time delays in the robot navigation. The uncertainty of the time delays was modelled by considering PDF in terms of Gaussian and Gamma distributions. The robot paths estimated by the delay-compensated AS-EKF algorithm and EKF algorithm that does not consider any delay are compared to evaluate the improvement in navigation performance.

The proposed model was experimentally implemented in simulation and verified in the real environment experimental framework with a commercial telepresence robot Beam plus. As the continuation of this work, we build a predictive display (as described in Chapter 6) to address the challenges of mismatch between the predicted state and actual navigation state of the robot. The predictive display is envisaged to show the immediate estimated robot path while the robot is navigating under a delayed network.

6 Robot navigation and pose estimation using predictive technology

This chapter proposes the design and development of a predictive display for telepresence robot navigation used in this research. In earlier chapters, we proposed techniques that address the challenges posed by time delay. The complete telepresence system also requires displays at the local site which provide robot poses to the operator. Due to time delays, there might be a mismatch between the real robot position and the measured position where predictive displays are used to reduce the visual disparity. In addition, predictive displays also provide the required infrastructure for controlled experiments. Real-life telepresence infrastructures are often expensive and not practical and therefore limit research efforts. As an alternative, we believe a simulation would mitigate such issues which can also offer capabilities of predictive displays in a real system. This chapter describes the need for predictive displays in the context of telepresence robot navigation and provides details of the design and development of such a system and how this can be used in our work by simulating similar experiments.

6.1 Introduction

As a general fact telepresence robots suffer significant challenges during navigation in the remote site due to varying communication time delays [3] which are frequently caused by the present state of the network. Moreover, the distance between the human operator and remote sites of the telepresence system intro-

duces time-varying delays adding distortion in the reference commands, response time and feedback signals resulting in instability or poor performance of the system. The time elapsed between making an action decision and perceiving the consequences of that action in the environment introduces an uncertain time delay.

This uncertain time delay produces a visual mismatch between the received navigation state of the robot from the remote site and the current navigation state of the robot at the operator's side which negatively impacts the human operator's performance. Therefore, it is advantageous to compensate for such time delays for robust navigation and manipulation of the telepresence robot.

There are various approaches used in the literature to overcome the time delay in telepresence systems including increasing levels of automation, more sensor on the robot and predictive technology. This chapter focuses on the predictive technology and within the context of this research it includes a state estimation algorithm, display and graphical models to predict the state of the robot based on the robot's delayed current state and commands sent by the operator [8].

The state of a robot is a set of position, orientation and velocity, which is the robot's motion over time. This includes the estimation of the state of the robot's kinematic system by combining knowledge from a priori information and sensor measurements. State estimation in dynamical systems is crucial in real-world applications as the true state is unknown and sensors have limited precision, therefore, provide only a sequence of uncertain noisy measurements.

The aim of this research is to estimate the true state of the robot compensating for any uncertain time delay present in the measurement data and show it on a predictive display at the local site. This chapter focuses on the latter part, *i.e.,* the predictive display that shows the estimated robot pose in real-time in front of the operator while the actual measurement data was delayed which was compensated by a state estimation algorithm. The predictive display helped to increase the operator's performance by addressing the challenges of visual time delay during real-time robot manoeuvring in an unknown environment.

6.1.1 Chapter contributions

While the idea of predictive display is matured for robot navigation only a handful of works are available in the literature [144] that consider delay compensation in the system and hence is an area to be much explored. In this work, we propose the design and develop a predictive display system that has the potential to compensate for the visual time delay by replacing the operator's desired viewpoint. It uses a delay-compensated model which forward predicts robot pose in time and provides the operator with the feeling of being situated at the remote site and directly performing the manipulations.

It is to be noted that the predictive display, developed in this work is based on a simulation where the delay compensation models are incorporated using separate MATLAB programs. We have focused on the development of a proof of the concept which, of course, can be adapted to a real system. There are two primary reasons for simulation-based predictive displays: 1) COVID-19 restriction that limits lab-based work and access to the hardware and 2) the creation of a simulation environment which could be beneficial in replicating an expensive real-life system.

In elaborating on the latter point, it is a well-known truth that real-life telepresence systems are expensive and extremely challenging to establish. This is due to the fact that local and remote sites are generally far apart, often separated by cities, countries, continents or even planets. Therefore any controlled experiments require a simulation environment. Development of our predictive display is, in fact, mitigates this challenge, not only for this thesis but also for the wider research community.

The main contributions of this chapter are,

1. Development of a new predictive display, where we *superpose* information coming from the real system and information coming from the simulation environment in order to compensate or anticipate the time delays. The sys-

tem also shows real and predicted information on the same display.

2. Validation of the predictive display through a series of experiments using a state-of-the-art differential-drive telepresence robot in the simulation environment using real navigation data.

3. Configurable and modular design of the simulation system to allow manipulation and simulation of different types of differential-drive robots by changing the robot's physical and experimental parameters for future usage.

6.2 Related Work

While it may sound repetitive, we briefly discuss the issues of a telepresence robot, especially the time delay again in this chapter as this is necessary to set out the context and background necessary for the predictive display. This section presents relevant related works in this field.

Within any telepresence system, a human operator controls a robot to interact in a remote environment through a communication channel for teleoperations. Literally speaking, a telepresence system is a set of technology which allow a person to feel as if they are present to give the appearance of being present or to have an effect at a place other than their true location. Seamless telepresence experience requires the implementation of human sensory elements such as vision, sound, and remote manipulation. However, the effectiveness of the telepresence system varies by the degree of fidelity, there are factors in telepresence which pose major challenges to precise and reliable robotic control for the human operator in an unknown environment. The main factors which influence navigation are limited field of view, operating multiple cameras and screens, depth perception, low frame rate, video quality and time delay.

Within the scope of this research, we focus only on the time delay or latency which refers to the time gap between the operator's input action and the received measurement response of the robot. The total time delay of the telepresence sys-

tem occurs due to a combination of a number of reasons, such as network switching delays, bandwidth limitation, communication drop-out, hardware processing delays and slow dynamics of the mobile robot. Time delays can be caused by physical limitations such as distance or obstacles between the operator and the robot too. Total time delay can be both certain and uncertain.

Time delay produces a visual mismatch between sending control commands and receiving robot position measurement data. Two different display creates a conflict in human perception. The human operator manoeuvring the robot in an unknown environment includes remote perception and remote manipulation [6]. Remote manipulation depends on the human operator's performance and is limited by the human's motor skills which involve distance estimation, obstacle detection, environment awareness and command generating. Remote perception is also very challenging to cope with the virtual display different from the physical environment. Teleoperation in an unknown environment with time delay is very difficult and highly stressful for the human operator which leads to mental fatigue [6, 145].

A potential solution for time delay is the predictive display. Predictive display using control command and robot state estimation algorithm, immediately display graphically the robot's estimated pose without time delay in the simulation environment. The estimated pose is usually superposed on the display of delayed measurement from the actual robot measurement. Predictive display refers to rendering a visualization of the robot site directly in response to the human operator's control commands, without waiting for the delayed video. Evidently human performance increases (about 20%) with the aid of predictive display in robot navigation [8].

Literature on predictive display for robot navigation can be dissected into the following five categories:

- With and without a priori model of the remote environment,
- SLAM-based predictive displays,

- Multi-sensor input predictive display,
- Computational hardware and
- Human operator performance measurement.

With and without a priori model of the remote environment

Traditionally, predictive displays were realised in highly pre-calibrated settings by superimposing hand-modelled wireframes and solid-model overlays of the robot manipulator and scene objects on top of the delayed video [112]. This approach was developed based on a known environment and a non-moving (fixed) external camera.

More recent work aims at producing photo-realistic predictive displays in less calibrated scenarios where a priori 3D model of the environment is not required [146]. Jin *et al.* [144] presented a vision-based semi-autonomous teleoperation system which is optimized for long-range teleoperation tasks under time delay network conditions. Similar to the previous one, it also does not require prior knowledge of the remote scene. The system initializes with a self-exploration behaviour that senses the remote surroundings through a webcam.

Burkert *et al.* [147] describe an online depth-fusion technique for predictive display that acquires a dense 3D geometry model using a stereo camera. CObzas *et al.* [148, 149] presented an image-based method for predictive display where the scene geometry and appearance are captured, compressed and transmitted to generate immediate feedback in response to the operator's movements.

SLAM-based predictive displays

Hu *et al.* [86] proposed a solution based on sparse 3D points provided by simultaneous localization and mapping (SLAM) and physically generates the correct surface for these point sets. Rachmielowski *et al.* [146] has reconstructed a coarse 3D geometry model using online SLAM. The predictive display was generated by

using the textures from key-frame images and rendering from the current operator viewpoint. This method also does not need a priori 3D model and can be applicable to a variety of unknown environments.

Lovi *et al.* [150] proposed a method using Parallel Tracking and Mapping (PTAM) [151] and free-space carving technique [152] with a telepresence interface and predictive display to generate rendered intermediate image for human control guidance. However, previous images are needed for a projective texture rendering which requires high network consumption and fully human-supervised control which causes difficulties in real-life applications.

Multi-sensor input predictive display

In recent years, algorithms were developed using multi-sensor (*e.g.,* optical images and Light Detection and Ranging (LIDAR)) data fusion techniques for predictive displays. One such system is proposed by Kelly *et al.* [153]. The authors constructed the real-time photo-realistic system by combining LIDAR data with images to reconstruct a 3D model.

Computational hardware

Due to the heavy computations involved in the predictive display systems, attempts were made for hardware optimisations. Schmid *et al.* [154] proposed dense mapping from a UAV using an FPGA (Field Programmable Gate Array) implementation of frame-by-frame dense stereo. On the contrary Hu *et al.* [86] proposed a solution which runs using only a standard low-power CPU. A system is implemented that reduces bandwidth by transmitting geometry and texture information instead of video. The performance is quantitatively characterized by time delays and qualitatively using NASA TLX analysis [155].

Human operator performance measurement

As mentioned earlier, predictive displays aim to help the human operator. A recent study by Dybvik *et al.* [8] shows even with the use of a simple predictive display, the performance can significantly be improved for teleoperated remote vehicle operation ($20\% \uparrow$) or gaming ($30\% \uparrow$). The study considered three distinct conditions for the experiments for 57 subjects: 1) Latency, 2) Latency with predictive display and 3) Baseline (no added latency).

In another study Orlosky *et al.* [156] evaluated the effect of panoramic view reconstruction (as opposed to simple screen) to reduce perceived latency in a humanoid robot. The central idea was to provide flexible head control tasks through the perspective of the robot.

Although various literature describes the development of predictive displays, these are not always easy to replicate due to the constrain of real-life measurements. On the contrary, our research focuses on the creation of the entire pipeline of predictive technology, which includes simulators for the telepresence robots, their control and navigation, provisions for controlled experiments by introducing certain and uncertain delays, state-estimation algorithms and predictive displays. Such a development is easy to replicate and can be used as a simulation framework for any similar telepresence systems. The following sections describe the design process in detail and the frameworks (*i.e.,* RViz and Gazebo) that were used to build this. It is worth noting that the entire development was purposefully built on the simulation mode rather than interacting with a real-life system. This would allow flexibility for wider adaptations.

6.3 Design and development of the proposed predictive technology

In designing the proposed predictive technology, we make use of three major open source software frameworks, namely, 1) Robot Operating System (ROS), 2)

RViz [59] which is a 3D visualisation tool for ROS and 3) a 3D robotics simulator, Gazebo [57]. The design of the proposed work is firmly based on our existing experimental framework, described in Section 4.3 as we see this is a natural extension of the same but in the simulation environment. The new updated version of the experimental framework with the predictive display is shown in the Figure 6.1, where there are two display screens, the first one for two-way communication and the second one for simulation of the predicted robot pose.

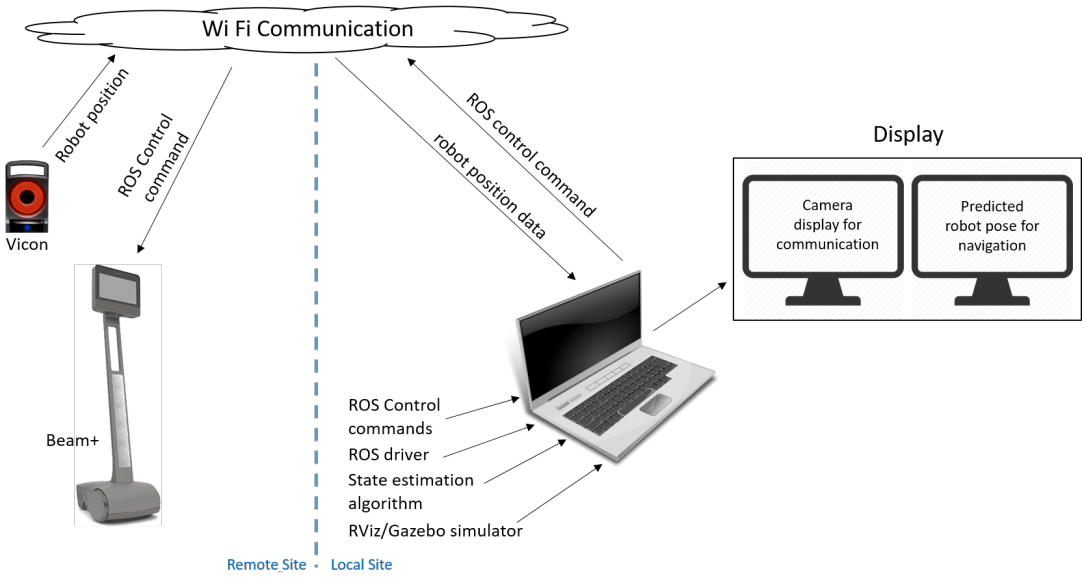


Figure 6.1: The experimental framework included the camera display for two-way human communication and the simulation display with predicted robot position for navigation.

Similar to the real environment, in this work, ROS is also used for path formation and control of the telepresence robot in the simulation environment. In modelling the time delay we simulate it by adding various time delays and apply our AS-EKF-based state estimation algorithm (see Section 5.3) to verify the outcome. As previously stated one of the major advantages of such a simulation environment is to visualise the outcome and the same is used to visualise the issues related to time delay and its mitigation using the AS-EKF. A real-like fictitious environment was created in this case using Gazebo and real-like observations were done by inserting a live camera along with the robot model.

The rest of this section comprises some necessary details of the software frameworks, RViv and Gazebo and various design, and development stages followed by some experimental results that were produced using the predictive technology.

6.3.1 RViz

ROS visualisation tool, abbreviated as RViz, is considered to be a powerful 3D visualization tool for ROS and is used in this work. In summary, it provides a convenient Graphical User Interface (GUI) to visualise the robot model, display and log information from the robot's sensors, and also allow replay of the (logged) sensor information. This is useful in building a simulator and aids the user by visualizing what the robot is seeing or doing. Therefore it enables the user to debug a robot application from sensor inputs against the planned (or unplanned) actions.

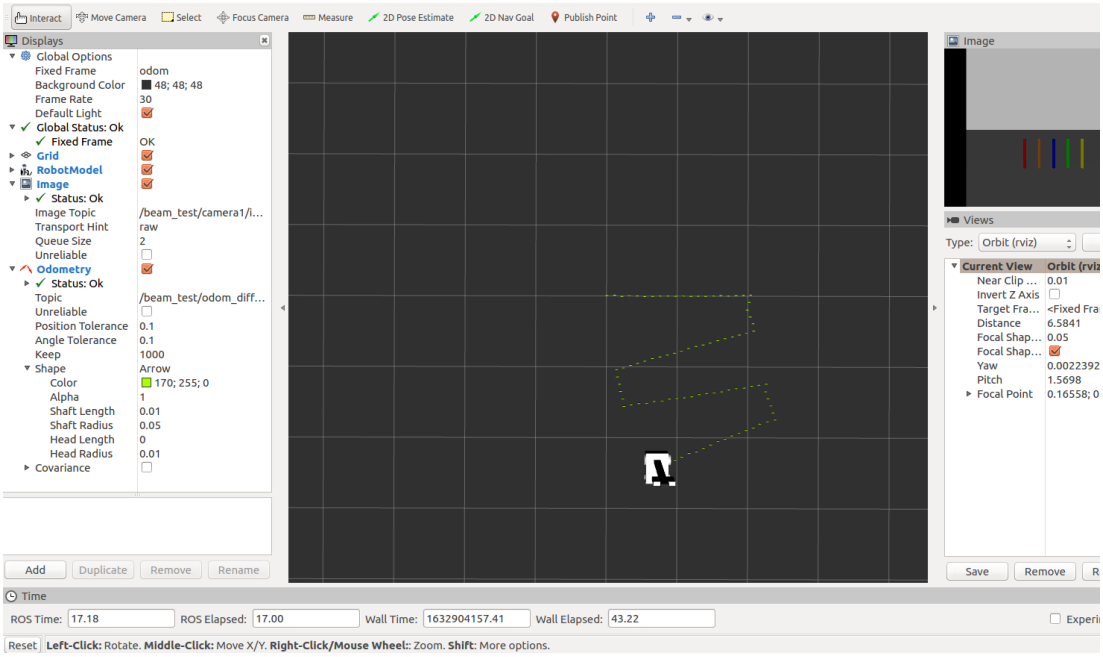


Figure 6.2: Example of the experimental environment in RViz built in this work to visualise the robot model, robot path and the camera display information.

As a generic description of RViz, it is capable of displaying 3D sensor data from stereo cameras, lasers, Kinects, and other 3D devices in the form of point clouds or depth images. 2D sensor data from webcams, RGB cameras, and 2D laser rangefinders are viewed in RViz as image data. It is to be noted that we only make

use of the 2D optical camera as a sensor to monitor the environment.

When RViz is integrated with an actual robot, if an actual robot communicates with a workstation running RViz, RViz will display the robot's current configuration on the virtual robot model. For example, if we fix a camera in the robot model in the Gazebo (as a simulator robot environment which is also used in our work), the camera information will be visualised in RViz. From the camera data, one can build a navigation path and it can also be used for auto navigation. In addition, in RViz one can access and graphically represent the values using a camera image. This information is useful to build point clouds and depth images. As a general functionality of RViz, one can select many displays to be viewed in RViz with data from different sensors [157].

In this work, we used RViz in conjunction with Gazebo to visualise the robot navigation path, view the environment by accessing the robot's camera and log navigation path information all of which are necessary to build the predictive display.

6.3.2 Gazebo

Similar to RViz, we briefly describe Gazebo in parts that are necessary to describe our work. In general, the Gazebo framework, an open-source solution, is used to create 3D scenarios on the computer with simulated robots, obstacles and other objects. The gazebo is also the default simulator used in the ROS framework. Although two separate projects, they are connected using gazebo_ros_pkgs. Details of this package can be found in its tutorial [57] which was heavily used in developing the proposed framework. It contains plugins that interface ROS and Gazebo. Gazebo accepts plugins developed in C++ alongside its external API (also in C++). These plugins are then attached to the simulated robot and provide easy ROS communication, such as both publish and subscribe control commands. The gazebo is a 3D simulator, while ROS serves as the interface for the robot. Combining both results in a powerful simulation system [58].

In our work, we built a Beam plus robot differential drive plugin (with the same configuration as the real robot that was used in this thesis) which was used to apply appropriate commands to the robot's motors. An example of our experimental environment is shown in Figure 6.3. At this juncture, it is probably worth noting that all the experiments were performed on Ubuntu 16.04 operating system, with ROS Kinetic, Gazebo 7 version.

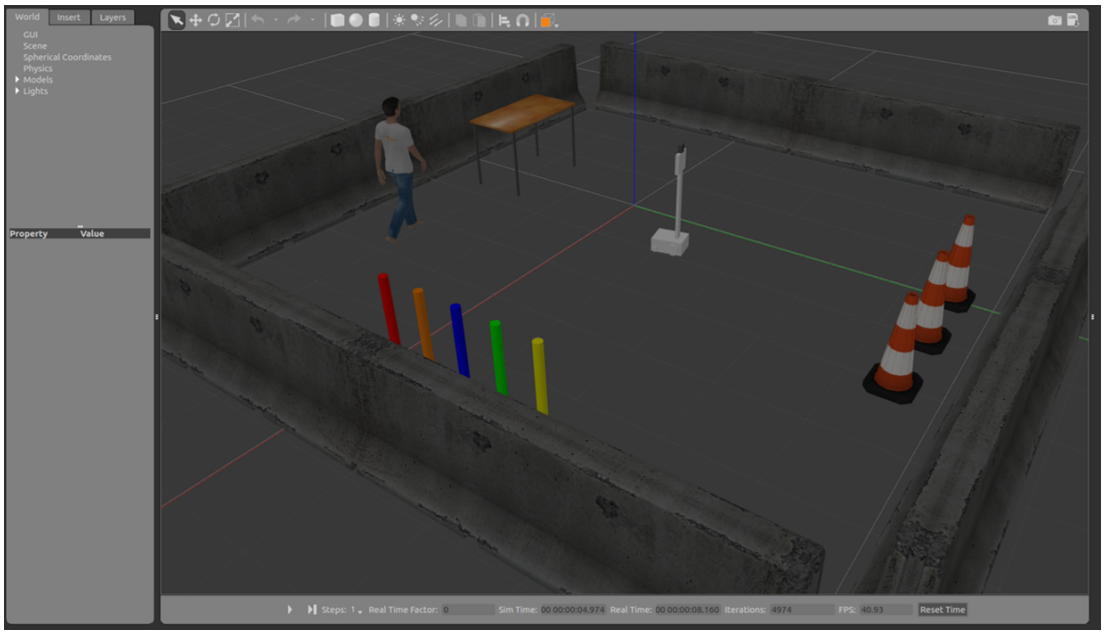


Figure 6.3: Example of the experimental environment in Gazebo simulator built in this work to visualize the real-time 3D scenario of the telepresence robot navigation on the computer controlled by the human operator.

6.3.2.1 Gazebo architecture and relevant libraries

Gazebo uses a distributed architecture with separate libraries for physics simulation, rendering, user interface, communication, and sensor generation. Additionally, the gazebo provides two executable programs for running simulations: 1) a server referred to as *gzserver* for simulating the physics, rendering, and sensors and 2) a client called *gzclient* that provides a graphical interface to visualize and interact with the simulation. The client and server communicate using the gazebo communication library.

In the Gazebo system, Gazebo Master is essentially a topic name server. It provides name lookup and topic management. A single master can handle multiple physics simulations, sensor generators, and GUIs. Almost all subsequent libraries use the communication library. It acts as the communication and transport mechanism for Gazebo. It currently supports only publishing/subscribing, but it is possible to use RPC with minimal effort.

The physics library provides a simple and generic interface to fundamental simulation components, including rigid bodies, collision shapes, and joints for representing articulation constraints. This interface has been integrated with four open-source physics engines: Open Dynamics Engine (ODE) [158], Bullet [159], Simbody [160] and Dynamic Animation and Robotics Toolkit (DART) [161]. A model described in the Simulation Description Format (SDF) using XML can be loaded by each of these physics engines. This provides access to different algorithm implementations and simulation features.

The rendering library uses Object-Oriented Graphics Rendering Engine (OGRE) [162] to provide a simple interface for rendering 3D scenes to both the GUI and sensor libraries. It includes lighting, textures, and sky simulation. It is possible to write plugins for the rendering engine.

The sensor generation library implements all the various types of sensors, listens to world state updates from a physics simulator and produces output specified by the instantiated sensors. The GUI library uses Qt to create graphical widgets for users to interact with the simulation. The user may control the flow of time by pausing or changing the time step size via GUI widgets. The user may also modify the scene by adding, modifying, or removing models. Additionally, there are some tools for visualizing and logging simulated sensor data.

The physics, sensor, and rendering libraries support plugins. These plugins provide users with access to the respective libraries without using the communication system. Deploying ROS Plugin for Gazebo helps to implement a direct communication interface to ROS, thus controlling the simulated and the real robots

using the same software. This provides an effective simulation tool for testing and development of real robotic systems [163].

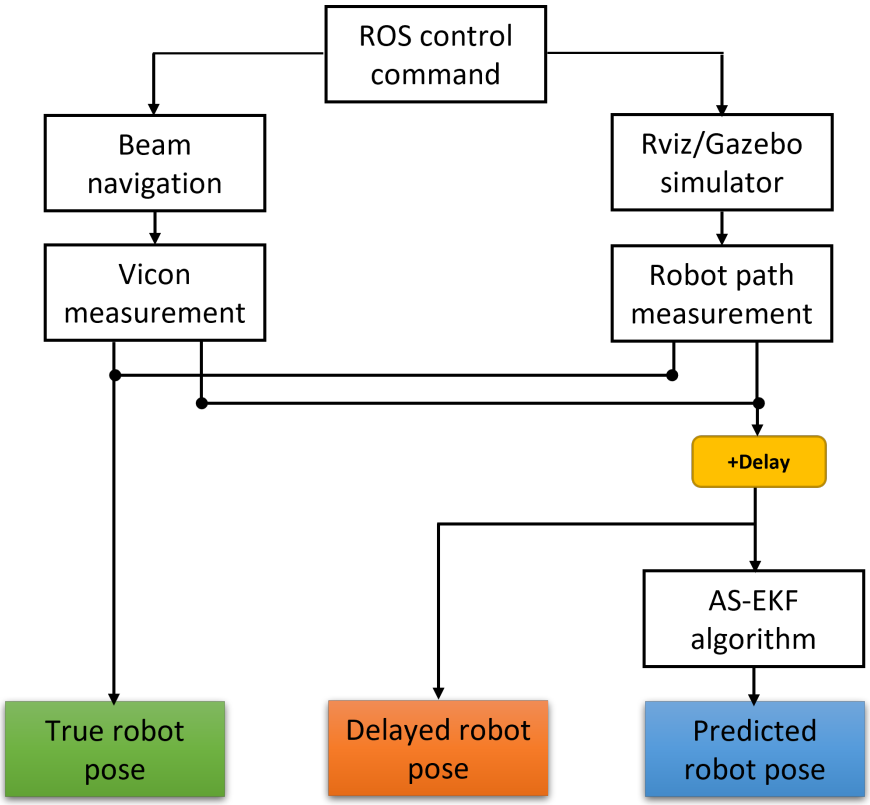


Figure 6.4: Flow diagram of the proposed experimental environment including three simulation displays for robot's true position, delayed sensor measurement and predicted robot pose.

6.3.3 Framework components

The proposed framework contains multiple components. Although they are driven by the necessity of this research, we aim to develop it in such a way that it is modular and could be adopted easily for any other type of telepresence navigation system. The requirements of the proposed framework include

- Creation of a simulation environment/world which also incorporates real-world equivalent items such as obstacles,
- Creation of a differential drive telepresence robot resembling the similar look and configuration of the Beam plus real robot that was used so far in this

work,

- Robot control and navigation through either ROS commands or through programs (for controlled experiments),
- Tracking and logging of navigation paths and
- Visualisation of the remote environment through camera sensor, and
- Inclusion of external navigation data (*i.e.,* VICON data from real Beam plus) in the simulation environment,

An overview flow diagram of the proposed framework is shown in Figure 6.4. In our work, we have modelled the delay by incorporating various types of delays (both certain and uncertain) which are then compensated with previously proposed AS-EKF algorithm (see Chapter 5). Finally, the navigation paths are shown in the simulation environment for various scenarios, *i.e.,* true robot pose, delayed robot pose and delay compensated predicted robot pose with an expectation that predicted and true poses are close to each other.

As a workflow at first in the simulator, the robot receives commands of the goal pose and the predefined trajectory via ROS control. The commands steer the simulated robot in the Gazebo environment. Gazebo then sends the robot pose information to the RViz to visualise the robot's trajectory. We have introduced a small time delay in the logged robot position data (both from the simulation and real-life VICON-captured robot data) and used it as the delayed measurement data. The delayed measurement data is processed by the AS-EKF-based state estimation algorithm to minimize the time delay. The algorithm generated predicted robot pose displays on the RViz.

Within the scope of this work, we intend to make a proof of the concept and therefore the development was done using both the ROS-RViz-Gazebo environment for simulation and visualisation whereas the delay modelling and AS-EKF were done in MATLAB. Integration of both and the creation of a single environment is considered to be future work.

6.3.4 Building the simulation environment

A robot simulation system is used to verify the output results, creating a virtual experimental environment for the physical robot depending on the actual robot specifications. In this research, we presented a simulation environment same to the real environment experimental framework based on ROS and Gazebo. We created the robot models precisely under Gazebo, the code developed for the simulation process and the state estimation predictive display directly implemented in the real robot without modifications. The simulation system includes five submodels,

- Building the robot model,
- Building the world model,
- Control plugins,
- ROS communication with Gazebo, and
- Visualization in Gazebo and RViz

6.3.4.1 Building the robot model

We have developed the simulated robot model which is a differential drive robot using the real physical specifications of the Beam plus robot. We built a Unified Robot Description Format (URDF) file for the robot that will describe the main components of our robot and enable it to be visualized and controlled by ROS tools, *e.g.,* RViz and Gazebo. Within the RViz visualization tool, we can view our URDF file as we built it in increments. When the visual model has completed, we can modify the URDF file to use in the Gazebo environment. In Gazebo, we can view the effects of physics on our model as we move the model around the 3D environment.

The overall description of the UDRF model (developed in this work) is shown in Figure 6.5. The components of the URDF model and the order of adding features during the development are described below.

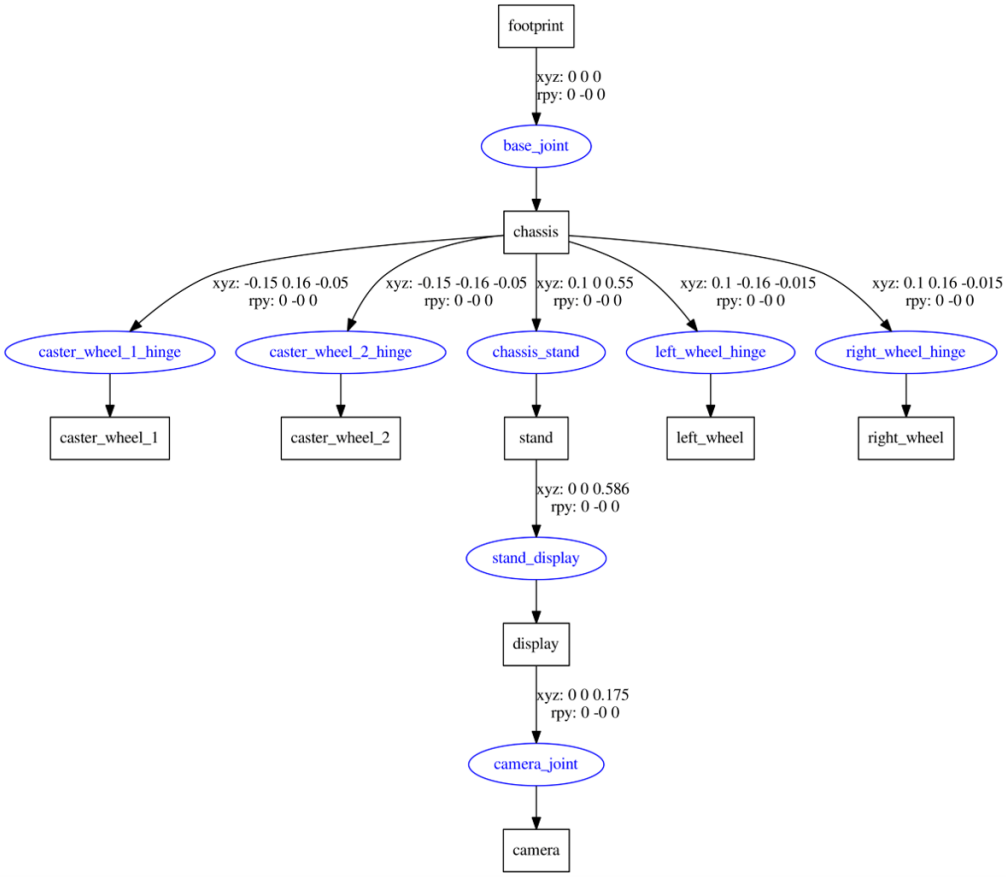


Figure 6.5: URDF description of the experimental robot, *i.e.,* Beam plus model with links and joints in the simulation environment.

Add a link A link contains the physical properties of one body of the model. This can be a wheel or a link in a joint chain. Each link may contain many collision and visual elements. The Beam plus model consists of nine links including footprint, chassis, four wheels, display stand, screen and camera. The steps for adding various links are given:

1. **Set the collision element:** A collision element encapsulates a geometry that is used for collision checking. This can be a simple shape or a triangle mesh.

2. **Set the visual element:** A visual element is used to visualize parts of a link. A link may contain 0 or more visual elements.

3. **Set the inertial properties:** The inertial element describes the dynamic properties of the link, such as mass and rotational inertia matrix.

4. **Add sensors:** A sensor collects data from the world for use in plugins. A link may contain 0 or more sensors.

5. **Add light elements:** A light element describes a light source attached to a link. A link may contain 0 or more lights.

Add a joint A joint connects two links. A parent and child relationship is established along with other parameters such as the axis of rotation, and joint limits. The beam plus model has eight joints few were fixed and few were continuous in nature.

Add plugins A plugin is a shared library created by a third party to control a model. The robot model is a differential drive robot so we have used a differential drive plugin to control the drive wheels and a camera plugin to capture the live camera data. In our case, we developed a customised model plugin to navigate the robot using both a computer keyboard and coding in C++ for a different type of path formation. Such a plugin is useful for any controlled experiments.

URDF model development The code defines the robot model named beam. The model contains nine links. These are nominally named link A, link B and link C, which connect via two joints, named joint A and joint B. The `<parent>` and `<child>` elements of the joints identify how the links connect to each other: link A connects to link B and link B connects to link C. link A has no parent link—that is, it appears in <joint> elements as a child element only—and is, therefore, the root link.

The `<inertial>` element of link A defines the mass and moments of inertia (`ixx`, `iyy`, `izz`) of the link. The products of inertia (`ixy`, `ixz`, and `iyz`) are unspecified and have the URDF default value of zero. The visual element of link A defines the geometry type and material colour for use in the model visualization. The geometry in this case is a box with a width and thickness of 0.5 m and height of 0.1

m. The <origin> elements of the link <inertial> and <visual> specify the transforms from the link reference frame to the inertial and visual reference frames. Similar elements apply to link B and link C.

The type attribute of the <joint> elements defines the joints as continuous—a type of revolute joint without motion limits. The <origin> element specifies the location of the joint relative to the reference frame of the parent link element. For example, the <origin> element of joint A offsets the joint 0.05m along the -Z axis relative to the origin of the link A reference frame. The axis element nested inside each joint element defines the rotational axis of the joint as the Cartesian vector $[0, 1, 0]$, or +Y.

The figure (Figure 6.5) shows the components of the model, *i.e.,* the links and joints and the various frames they contain. R denotes a link reference frame, I a link inertial frame, and V a link visual frame. J denotes a joint reference frame—by definition held coincident with the reference frame of the child link. The inertial and visual frames are offset to the centres of the links and the joint frames to their lower edges.

The code was developed in a standard Gazebo environment where UDRF and other descriptions are often encoded within XML (Extensible Markup Language) formats. For completeness, an example code snippet is shown in Figure 6.6.

6.3.4.2 Building the world model

The term world used to describe a collection of robots and objects such as buildings, tables, lights and global parameters including the sky, ambient light, and physics properties. The world included both static and dynamic objects. Static objects only have collision geometry. All objects which are not meant to move have been marked as static, which was for performance enhancement. Dynamic objects have both inertia and collision geometry. We used Gazebo's model database which is a repository of all types of models including robots, tables, buildings, obstacles, persons etc.

```
<?xml version="1.0" ?>
<robot name="beam_test">
<!-- ############## Plugins ############### -->
  <gazebo reference="chassis">
    <material>Gazebo/White</material>
  </gazebo>
<!-- ****************************** -->
  <gazebo reference="caster_wheel_1">
    <mu1>0.0</mu1>
    <mu2>0.0</mu2>
    <material>Gazebo/White</material>
  </gazebo>
  <gazebo reference="caster_wheel_2">
    <mu1>0.0</mu1>
    <mu2>0.0</mu2>
    <material>Gazebo/White</material>
  </gazebo>
  <gazebo>
<!-- ****************************** -->
    <plugin filename="libgazebo_ros_control.so" name="gazebo_ros_control">
      <robotNamespace>/beam_test</robotNamespace>
      <legacyModeNS>true</legacyModeNS>
    </plugin>
  </gazebo>
  <gazebo>
<!-- ****************************** -->
    <plugin filename="libgazebo_ros_diff_drive.so" name="differential_drive_controller">
      <legacyMode>false</legacyMode>
      <alwaysOn>true</alwaysOn>
      <updateRate>100</updateRate>
      <leftJoint>left_wheel_hinge</leftJoint>
      <rightJoint>right_wheel_hinge</rightJoint>
      <wheelSeparation>0.463</wheelSeparation>
      <wheelDiameter>0.15</wheelDiameter>
      <torque>20</torque>
      <commandTopic>beam_test/cmd_vel</commandTopic>
      <odometryTopic>beam_test/odom_diffdrive</odometryTopic>
      <odometryFrame>odom</odometryFrame>
      <robotBaseFrame>footprint</robotBaseFrame>
    </plugin>
  </gazebo>
<!-- ****************************** -->
  <gazebo reference="camera">
    <material>Gazebo/Grey</material>
    <sensor name="camera1" type="camera">
      <update_rate>30.0</update_rate>
      <camera name="head">
```

Figure 6.6: Example code snippets of robot model developed for the simulation.

6.3.4.3 Control plugins

A plugin is a chunk of code that was compiled as a shared library and inserted into the simulation. The plugin has direct access to all the functionality of Gazebo through the standard C++ classes. Plugins are useful because they let developers control almost any aspect of Gazebo, are self-contained routines that are easily shared and can be inserted and removed from a running system.

Plugins were useful to programmatically alter the simulation such as moving models, responding to events, insert new models given a set of preconditions. It makes a fast interface to the Gazebo, without the overhead of the transport layer

(no serialization and deserialisation of messages). There were currently six types of plugins such as, *World*, *Model*, *Sensor*, *System*, *Visual* and *GUI*. Each plugin type is managed by a different component of Gazebo. For example, a *Model* plugin is attached to and controls a specific model in Gazebo. Similarly, a *World* plugin is attached to a world and a *Sensor* plugin to a specific sensor. The *System* plugin is specified on the command line and loads first during a Gazebo startup. This plugin gives the user control over the startup process.

6.3.4.4 ROS communication with Gazebo

To achieve ROS integration with the stand-alone Gazebo, a set of ROS packages named `gazebo_ros_pkgs` provides wrappers around the stand-alone Gazebo. They provide the necessary interfaces to simulate a robot in Gazebo using ROS messages and services. The ROS APIs allow users to modify and get information about various aspects of the simulated world.

6.3.4.5 Visualize in Gazebo and RViz

As previously mentioned RViz (ROS visualization) is a 3D visualization software tool for robots, sensors, and algorithms. It enables seeing the robot's perception of its world (real or simulated). The gazebo is a 3D robot simulator. Its objective is to simulate a robot, giving a close substitute for how a robot would behave in a real-world physical environment. The main difference between the two is Gazebo enables the user to see what is happening along with the environment (as if in the real environment) whereas RViz provides key information about the robot including its navigation path or sensor measures, *i.e.,* camera images [164].

In our case, we used both as 1) Gazebo is necessary to simulate a real-like environment and 2) RViz provides the necessary tools to log the robot positions and view the world from the robot's perspective. Together they form the **Predictive Display** as intended in this work.

With the development of this framework, we now conduct a series of experi-

ments that resembles experiments that were done in earlier chapters. The experimental setups and the results are shown in the following sections.

6.4 Experimental results and discussions

We performed a series of controlled experiments using the proposed predictive technology framework. While the primary aim of this chapter is the design and development of the framework, this section verifies the usefulness of the same where we conduct a set of experiments, similar to the ones that were reported in the previous chapter (with real Beam plus).

6.4.1 Experimental flow and parameters

In all experiments a common workflow / experimental flow was maintained which includes the following steps:

1. Robot navigation using ROS,
2. Logging of robot pose captured through the navigation,
3. Delay modelling by adding certain or uncertain delays,
4. Delay compensation using the previously proposed AS-EKF algorithm,
5. Visualisation of the robot navigation in the simulated environment through the Gazebo, and
6. Visualisation of robot path and robot view of the environment using a camera in RViz.

As previously mentioned, this work focuses on the proof of the concept rather than a complete integrated system. Therefore, some steps, *step 3* and *step 4* in the above list are performed using MATLAB. The output of *step 2*, *i.e.,* robot pose was exported to a CSV file using a suitable *rostopic* command, which is then used in MATLAB. The output of *step 4*, *i.e.,* predicted robot pose after AS-EKF, are again

exported through a CSV file which is then taken as the input to Gazebo and RViz through a custom-built plugin.

In creating the robot model in Gazebo, we used robot parameters the same as Beam plus. A similar raster scan navigation path was created pragmatically using another custom-built C++ plugin in Gazebo. Two sets of results are obtained in various delayed scenarios:

- Time series average error between the original robot path and delayed as well as corrected robot path, and
- How robot sees its surrounding environments in the case of the original path delayed path and corrected path.

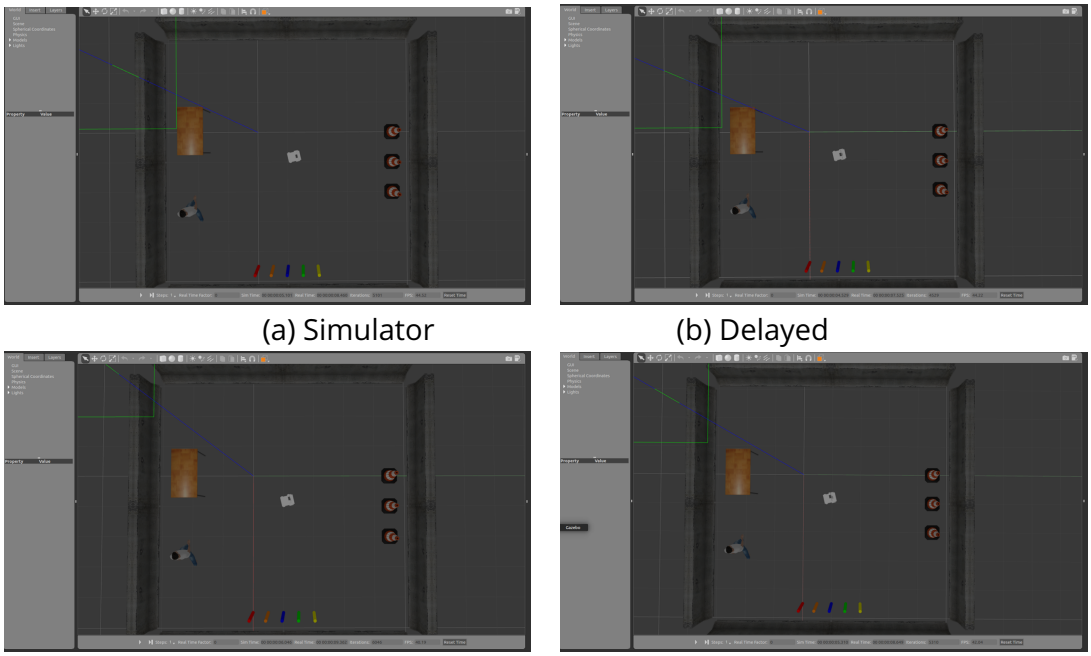
For the first set of results, it is expected the overall mean square error would be reduced significantly and this should be reflected in the visualisation. For the latter part, we fixed the robot after a certain step where one could see the surrounding items in a manner that there is a visible difference between the original and delayed path whereas after AS-EKF correction it is closer to the original path.

The following subsection reported such results for both certain and uncertain delays. As with the previous chapter (Chapter 5), we consider two types of uncertain delays, 1) Gamma and 2) Gaussian, respectively. For a reasonable visualisation, we introduced an average of 50 time step delays in all cases which would translate to a 0.1s delay in reality. However, as an experimental parameter, this can be varied to simulate various other scenarios.

6.4.2 Scenario I: Certain time delay

In the case of a certain time delay, we have introduced a fixed number of time steps delay ($\tau = 50$) for each robot pose and measured the moving mean errors between the original and delayed, EKF and AS-EKF prediction of the robot pose. All other parameters are kept as in the previous experiment. Figure 6.7 provides the top view visuals of the robot position in the Gazebo simulation environment for all four scenarios.

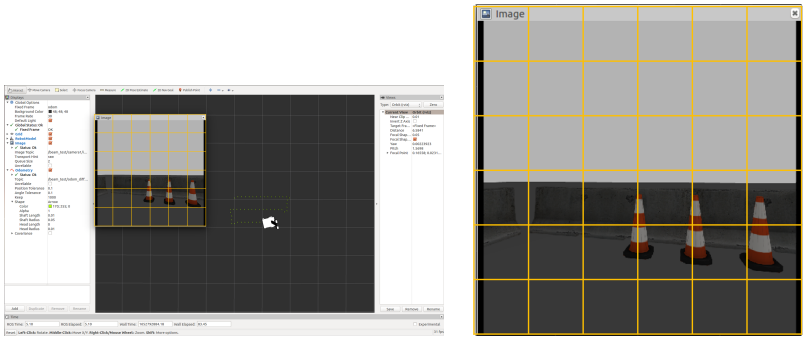
In order to visualize the environment from the robot camera, we used RViz and the results are shown in Figure 6.8. It is evident that there is a difference between the original robot pose and the delayed robot pose on what the robot can see at that point in time. As expected EKF can not improve the prediction. However, AS-EKF performed well and compensated for the delay and offer a near-original robot pose. Finally, we also measure and compare the RMSE error between EKF and AS-EKF in Figure 6.9, which shows expected significant improvements in the case AS-EKF.



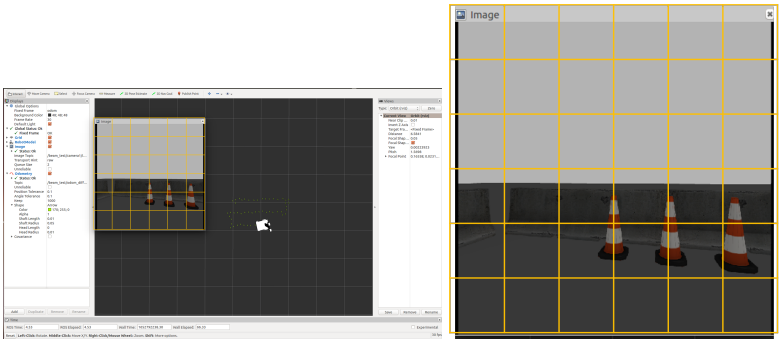
(a) Simulator (b) Delayed

(d) AS-EKF (c) EKF

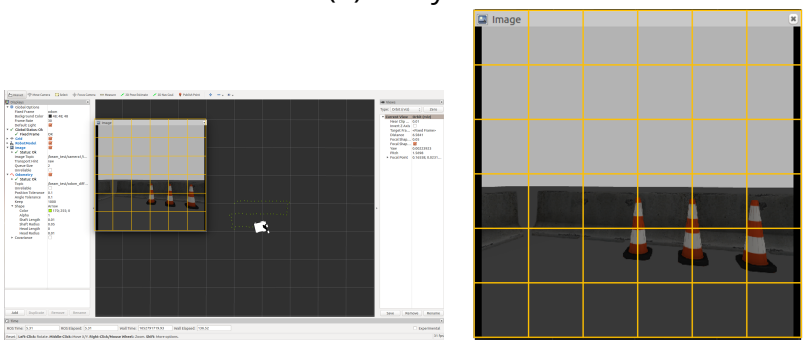
Figure 6.7: Simulation environment in Gazebo for the certain time delay ($\tau = 50$ steps). (a), (b), (c) and (d) represent robot positions within the simulation environment without delay, with delay, and compensation with EKF and AS-EKF, respectively. The visualisation and results indicate that it can emulate the real experimental set-up and introduce a delay for controlled experiments along with applying EKF/AS-EKF for time delay compensation.



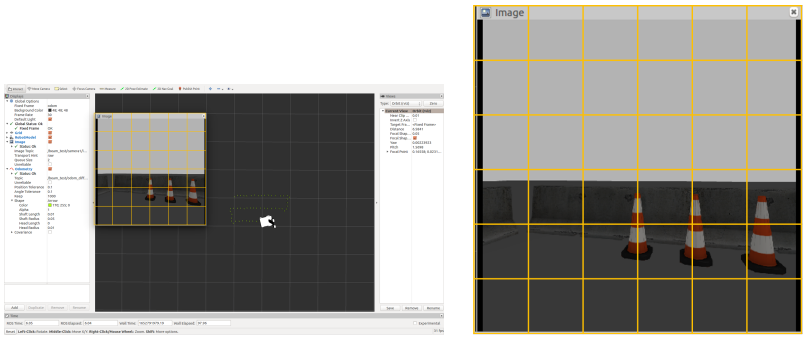
(a) Simulator



(b) Delayed



(c) EKF



(d) AS-EKF

Figure 6.8: Display in RViz for the certain time delay ($\tau = 50$ steps) along with the robot view of the environment. Figures in the left column show the robot positions in the simulation environment for (a), (b), (c) and (d) representing without delay, with delay, compensation with EKF and AS-EKF, respectively. The right column shows the robot's view of the environment which indicates the original without delay and AS-EKF has a very close environmental view proving the effectiveness of the proposed algorithm and the use of this simulator.

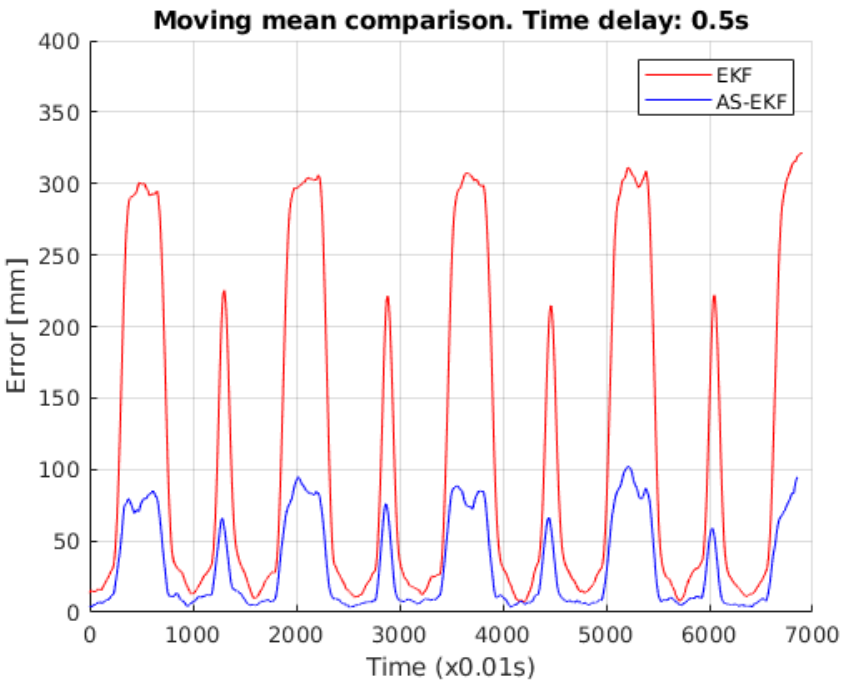


Figure 6.9: RMSE error comparison of EKF and AS-EKF estimation with a certain delay. Legend red and blue represent results for EKF and As-EKF, respectively with AS-EKF showing significant improvements with less error.

6.4.3 Scenario II: Uncertain time delay

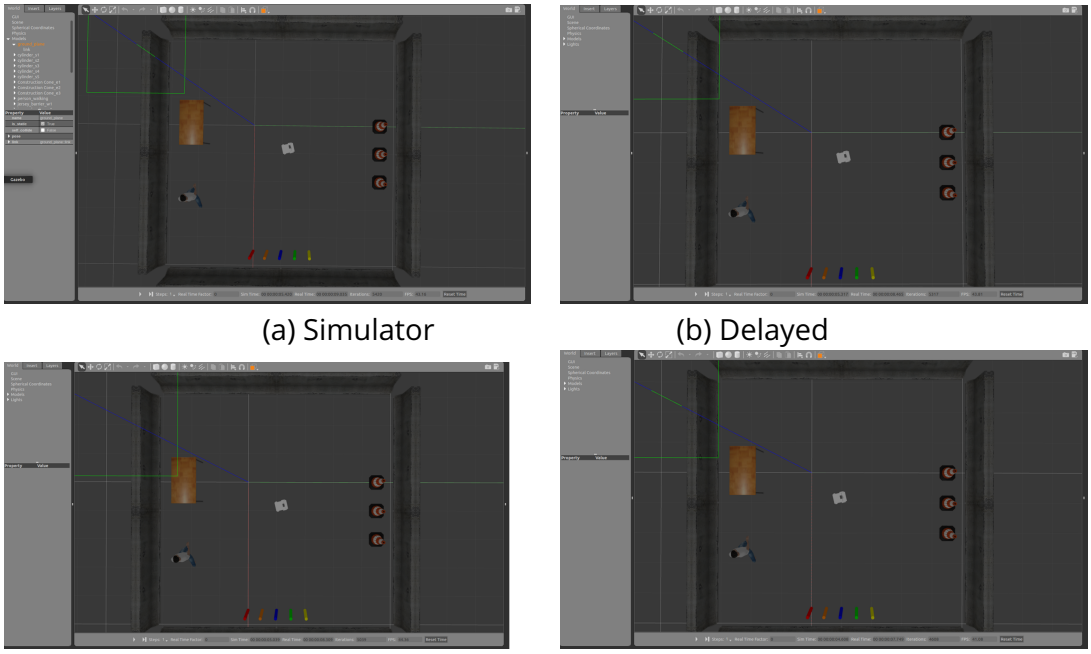
For uncertain delay, we have modelled the delay using Gamma and Gaussian distributions with an average delay of $\tau = 50$ steps. Similar to Scenario I, we have captured the visualisation in Gazebo and RViz and compared the error or differences in the visualisation for the original path, delayed path, and EKF and AS-EKF predictions.

Gazebo visualisations are shown in Figure 6.10 and Figure 6.13 for Gamma and Gaussian distributions, respectively. Similarly, RViz visualisations are shown in Figure 6.11 and Figure 6.14. Finally, the comparison between the EKF and AS-EKF performances is shown in Figure 6.12 and Figure 6.15, for Gamma and Gaussian distributions, respectively. Finally, Table 6.1 shows the RMSE error comparison for all three scenarios.

Evidently, as expected in all cases, AS-EKF consistently performed well and also show better pose predictions which are close to the original robot path. The results also show the capabilities of the proposed predictive technology framework when real visualisation and controlled experiments are plausible in the simulation environment.

Delay type	**EKF**	**AS-EKF**	**Improvement**
Certain	114.82	30.84	73.15%
Uncertain (Gamma)	111.73	30.62	72.60%
Uncertain (Gaussian)	112.15	31.79	71.65%

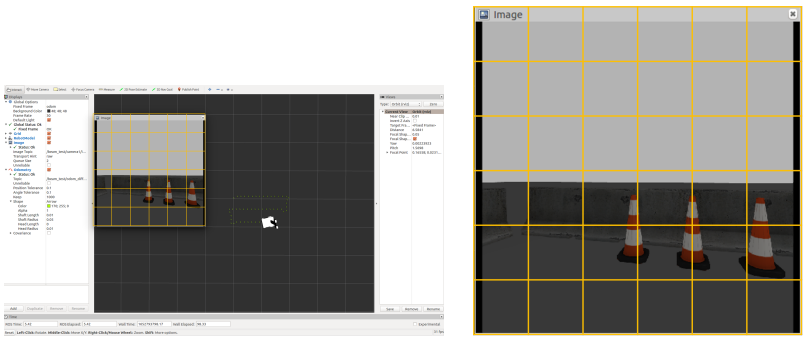
Table 6.1: RMSE error comparison for certain and uncertain time delays (average $\tau = 50$). AS-EKF shows major improvement over EKF which does not consider the delay in its filtering steps.



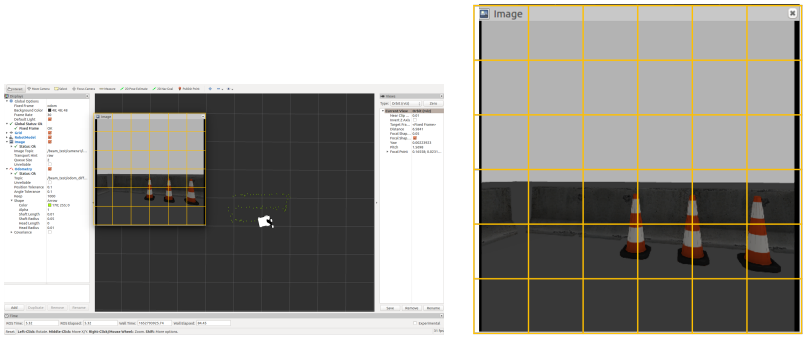
(a) Simulator (b) Delayed

(d) AS-EKF (c) EKF

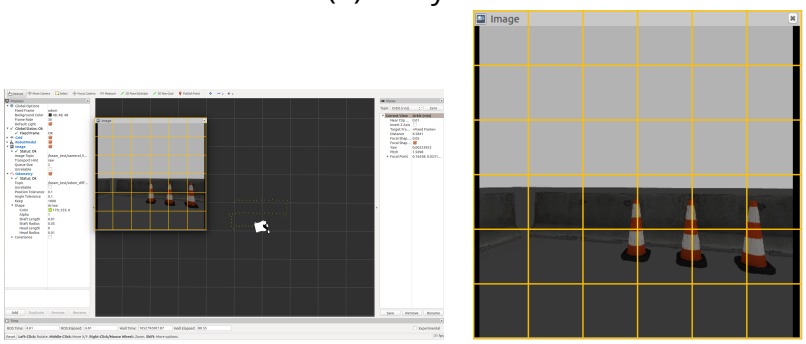
Figure 6.10: Simulation environment in Gazebo for uncertain time delay with Gamma distribution (average $\tau = 50$ steps). (a), (b), (c) and (d) represent robot positions within the simulation environment without delay, with delay, and compensation with EKF and AS-EKF, respectively.



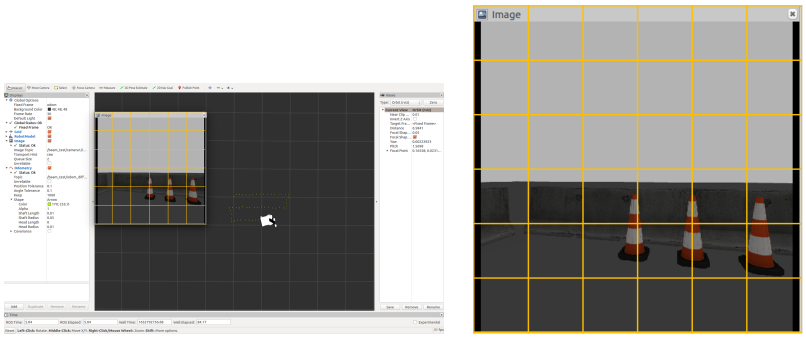
(a) Simulator



(b) Delayed



(c) EKF



(d) AS-EKF

Figure 6.11: Display in RViz for uncertain time delay with Gamma distribution (average $\tau = 50$ steps) along with the robot view of the environment. Figures in the left column show the robot positions in the simulation environment for (a), (b), (c) and (d) representing without delay, with delay, compensation with EKF and AS-EKF, respectively. The right column shows the robot's view of the environment which indicates the original without delay and AS-EKF has a very close environmental view proving the effectiveness of the proposed algorithm and the use of this simulator.

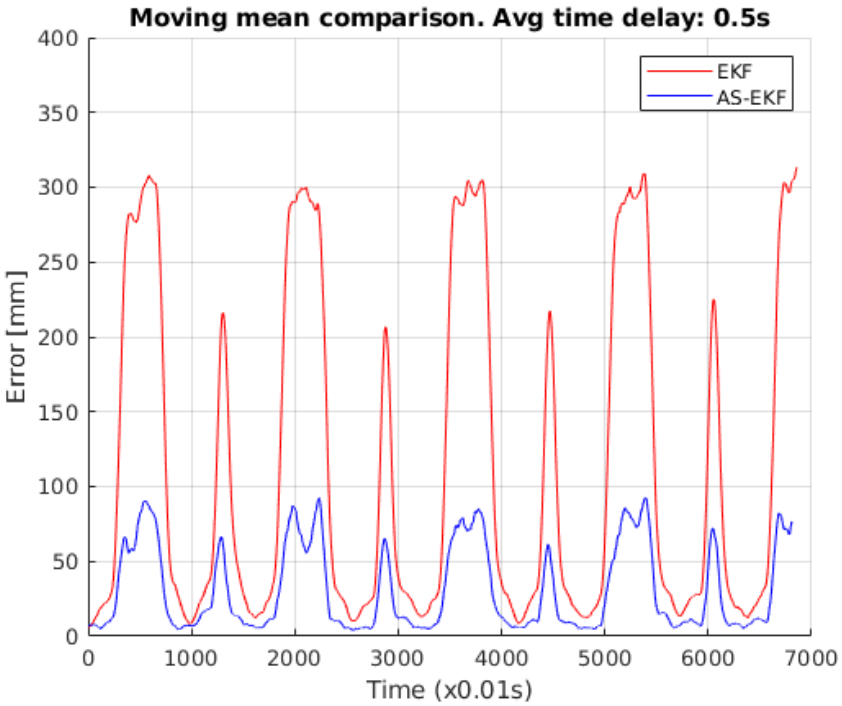
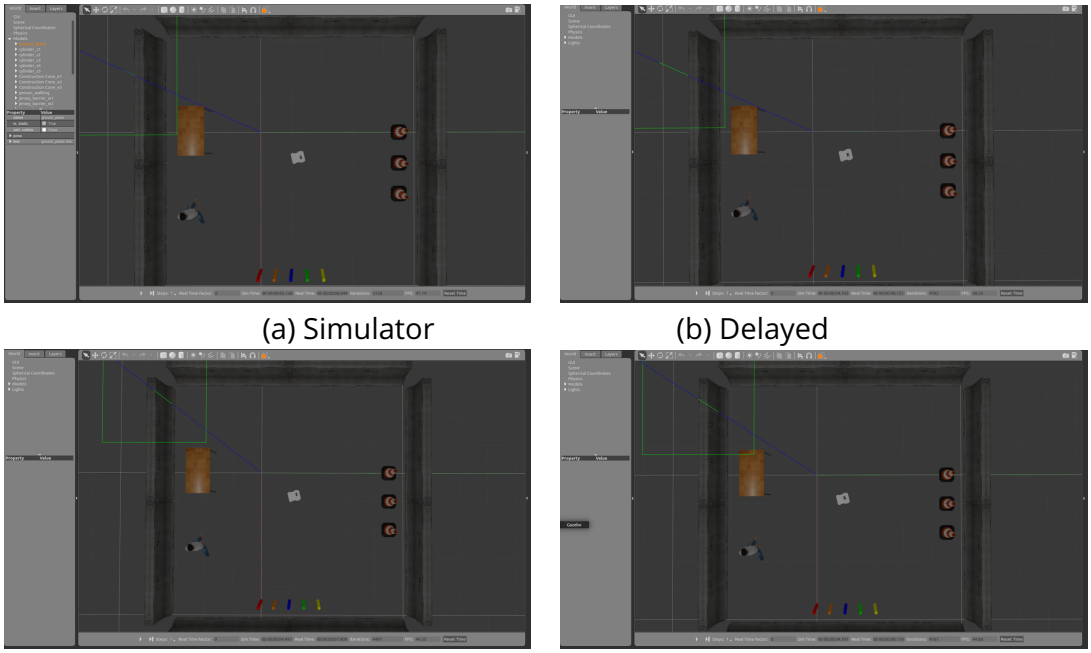


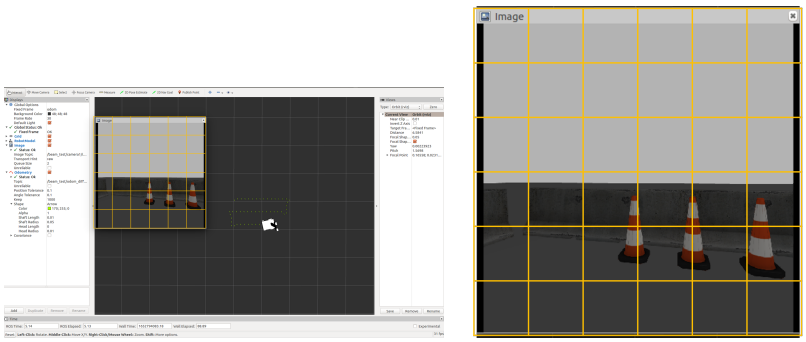
Figure 6.12: RMSE error comparison of EKF and AS-EKF estimation with uncertain delay (Gamma distribution). Legend red and blue represent results for EKF and As-EKF, respectively with AS-EKF showing significant improvements with less error.



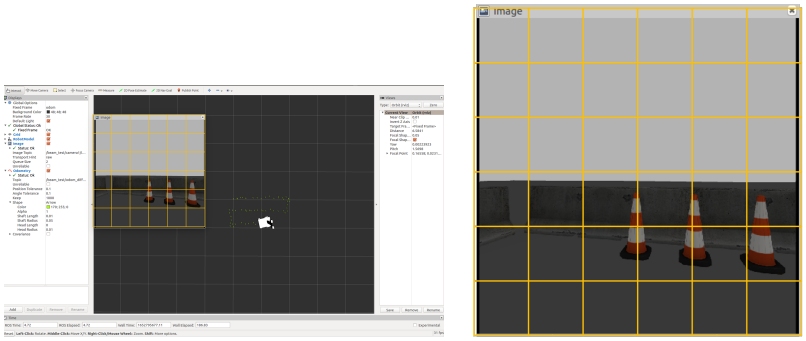
(a) Simulator (b) Delayed

(c) AS-EKF (d) EKF

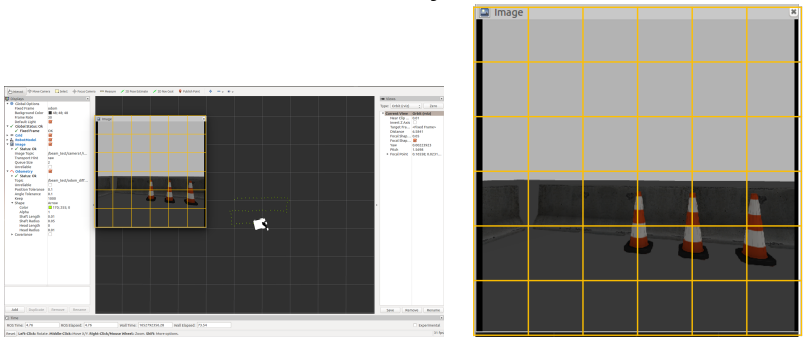
Figure 6.13: Simulation environment in Gazebo for uncertain time delay with Gaussian distribution (average $\tau = 50$ steps). (a), (b), (c) and (d) represent robot positions within the simulation environment without delay, with delay, and compensation with EKF and AS-EKF, respectively.



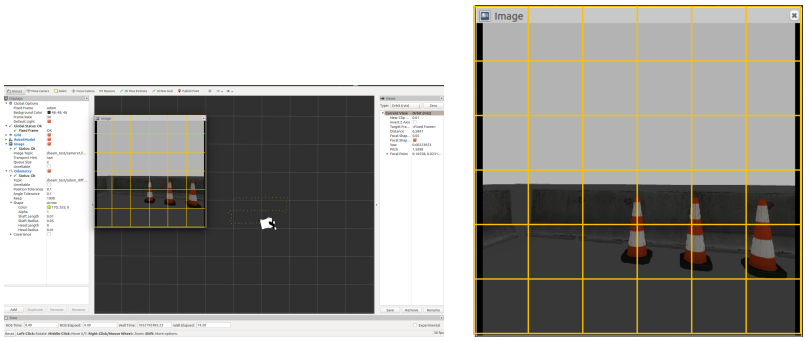
(a) Simulator



(b) Delayed



(d) EKF



(c) AS-EKF

Figure 6.14: Display in RViz for uncertain time delay with Gaussian distribution (average $\tau = 50$ steps) along with the robot view of the environment. Figures in the left column show the robot positions in the simulation environment for (a), (b), (c) and (d) representing without delay, with delay, compensation with EKF and AS-EKF, respectively. The right column shows the robot's view of the environment which indicates the original without delay and AS-EKF has a very close environmental view proving the effectiveness of the proposed algorithm and the use of this simulator.

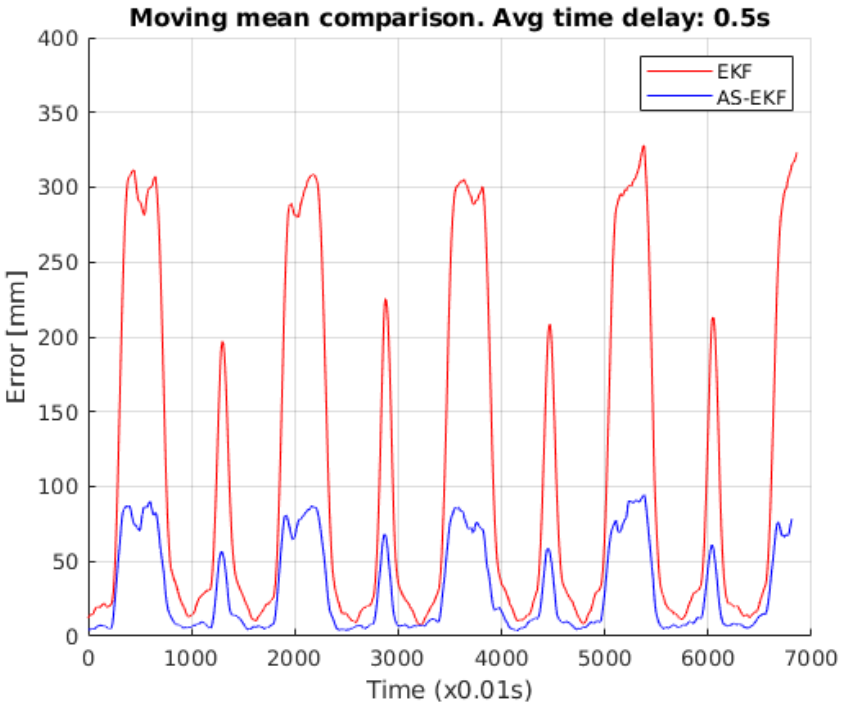


Figure 6.15: RMSE error comparison of EKF and AS-EKF estimation with uncertain delay (Gaussian distribution). Legend red and blue represent results for EKF and As-EKF, respectively with AS-EKF showing significant improvements with less error.

6.5 Chapter summary

This chapter proposes the design development of the predictive technology for differential drive telepresence systems. The overall system has three major components, namely, 1) ROS, 2) RViz and 3) Gazebo. RViz and Gazebo are used for visualisation whereas ROS is used to control the robot.

A differential drive telepresence robot that resembles the Beam plus looks and configuration was developed in Gazebo and controlled through ROS using custom-built plugins. Raster scan-based robot navigation was done using a C++-based plugin and the robot pose was captured. This original robot pose was then used for delay modelling with certain and uncertain delays (with both Gamma and Gaussian distributions). The delays are then compensated using standards EKF and previously proposed AS-EKF algorithms. Finally, the outputs are visualised in Gazebo as well as in RViz to replicate the real-life telepresence system with predictive display.

Results for various scenarios show the capability of the proposed predictive technology simulation framework and the superiority of the AS-EKF algorithm. Such a framework is beneficial for two purposes, 1) predictive display and 2) conducting controlled experiments in the simulation environment. Especially the latter is beneficial to the wider research community as a real-world telepresence system is expensive and often unrealistic to realise in research environments.

7 Conclusions and future work

In concluding the work carried out in this thesis, this chapter discusses the outcome of the chapters along with the overall concluding remarks. As expected the conclusions lead to further research directions that are noted in Section 7.2 on the future work. The future work also includes details of initial experiments on robot navigation with Visual Simultaneous Localization and Mapping (VSLAM).

7.1 Conclusions

This thesis conducted research on the robust navigation of telepresence robots. Telepresence systems are necessary for multiple applications areas where human physical presence is either not possible, dangerous or unnecessary. Such applications include space applications, military applications, telesurgery, industrial work or more recently telepresence in an office environment, hospitals or individual homes. We are largely interested in the latter application and focused on their robot navigation issues, especially time delay.

Time delays could be caused by a number of factors such as delays due to communication, system hardware, computational overheads or even latency due to physical distances. Modelling such delays and a compensation mechanism is of our interest, and this thesis proposed strategies to mitigate them. However, in order to design, develop and verify new algorithms, requires a reasonable telepresence framework to conduct experiments. Thus we built them by two means,

1) a real-like scenario where we used an off the shelve state-of-the-art differential drive telepresence robot Beam+ along with a VICON multi-camera system for tracking and 2) an end-to-end simulation environment using open-sourced software such as Gazebo and RViz. In both cases, the robot operating system, ROS, played a fundamental role to control and navigate the robot and measuring and log necessary sensor outputs.

The core of this work focuses on an algorithm that compensated for the time delay. This was achieved by proposing the algorithm called Augmented State Extended Kalman Filter (AS-EKF). In this process, we used EKF as the baseline and incorporated augmented state models in order to estimate and compensate for the errors caused due to the delay. The proposed algorithm successfully executed estimating actual robot position modelling certain and uncertain time delays in the robot navigation. The uncertainty of the time delays was modelled by considering PDF in terms of Gaussian and Gamma distributions. The results show major improvement over baseline EKF which does not consider time delays.

The thesis consists of seven chapters which include the problem statement, experimental frameworks, a new algorithmic proposal and future research directions.

- **Chapter 1** provided necessary introduction and outline of the thesis.
- **Chapter 2** presented a general overview of the telepresence systems, their application areas and the background that is necessary for this research.
- **Chapter 3** presented the state-of-the-art analysis with a focus on robot navigation in local and remote sites, the role of communications and challenges and mitigation strategies around time delay issues.
- **Chapter 4** described the real-world experimental framework that consists of an off shelve Beam plus telepresence robot, VICON motion tracking camera system, and methodologies for robot control and navigation using ROS, correction of systematic error using UMBMark method and other measure-

ments.

- **Chapter 5** proposed a new approach for state estimation assuming uncertain delayed sensor measurements of a telepresence robot during navigation. A technique based on Augmented State Extended Kalman Filter (AS-EKF), is proposed to estimate the true position of the telepresence robot. The uncertainty of the delayed sensor measurements has been modelled using probabilistic density functions (PDF) of Gamma and Gaussian distributions.

- **Chapter 6** presented the design and development of a simulation environment for robot navigation and pose estimation using predictive technology. This development incorporates the predictive display which is necessary for human operators at the local site. Additionally, the simulation environment allows conducting controlled experiments as a real-world telepresence system is expensive and often not so plausible to develop.

- Finally, **Chapter 7** concluded the thesis and discussed future work. In this context, initial experiments were conducted by proposing a visual SLAM-based experimental set-up which would allow robot pose estimation in unknown environments. Further research directions were discussed especially when such measurements (*i.e.,* pose from visual SLAM) are noisy and potentially erroneous.

7.1.1 Key conclusions

While telepresence systems exist for decades for various applications, adaptation to consumer life is fairly recent with the introduction of telepresence robots in offices or care homes. This has been even more relevant in today's world when the COVID-19 pandemic restricts travel and physical human interactions. However, there are a few problems that remain regarding the robust navigation of robots in the remote environment. Time delay is one of the key challenges and this thesis proposed solutions to address that.

Traditional EKF often fails to consider the issues with a time delay as it largely concerns in addressing positional errors in a noisy environment. However, we showed that the issues could be addressed by incorporating augmented states by extending traditional EKF that compensate for the time delay. We developed a new algorithm (AS-EKF) that demonstrates the improvement up to 54% for a certain time delay (for a time delay of 25 steps).

Now considering a real environment where certain time delays are not expected, we considered uncertain time delays. Uncertain time delays are modelled using two PDF distributions representing various time delays in such systems and discussed in detail in Section 4.4.2). The proposed AS-EKF considered those uncertain delays and reports major improvements over standard EKF, over 50% for both Gaussian and Gamma distributions (for 25 steps time delay, same as *certain*). This in fact demonstrates the robustness of the proposed algorithm for wider application scenarios.

The other main proposition of this work is the creation of predictive technology in a simulation environment. It is well understood that the realisation of a telepresence system is often implausible for various reasons including cost and infrastructure requirements. Even with such a system, conducting controlled experiments are hard to achieve. We not only overcome these challenges through the design and development of a framework with the use of RViz and Gazebo but also provide a framework to the research community for wider adaption. We demonstrated that one can reproduce similar results in the simulation environment which also provides a platform for predictive display in a telepresence system.

7.2 Future work

This thesis conducted research in improving telepresence robot navigation in time-delayed conditions. In order to compensate for certain and uncertain delays we proposed (in Chapter 5) an Augmented State Extended Kalman Filter (AS-EKF) that shows significant improvement over traditional EKF. The development and verification of the algorithm were achieved in two parts,

- With the use of an off-the-shelve telepresence robot Beam plus (controlled using ROS) along with its experimental framework that includes a state-of-the-art VICON motion tracking system and
- A custom-built simulation environment that uses open-source visualisation and modelling software RViz and Gazebo, respectively.

The algorithm development relies on various delays that were artificially incorporated based on the delay models in a similar situation as available in the literature. Therefore as a natural way forward would be to use a real-world system that by nature incorporates unknown delays and positional errors.

In addition, so far our experimental setups used either external measuring equipment,*i.e.,* VICON or tools available in the simulator to measure the robot pose. In practice, neither of them is available for a real system deployment. That indicates the need for an in-situ measurement sensor(s) that can provide robot pose information.

We understand that controlled experiments in a real-world experimental set-up are challenging (if not impossible). However, considering the above scenarios, one could think of future work in a direction that can realise such an experimental framework and try to verify the proposed delay compensations technique.

In this work, such an attempt was made using a new set-up where a camera was installed on top of the Beam plus telepresence robot and visual SLAM was applied to measure the robot's pose. Admittedly it encountered a set of challenges includ-

ing the COVID-19 lockdown over the past years when lab access was restricted and therefore the plan did not go forward as expected. As an alternative, this thesis develops the simulation environment and continued the work.

However, we believe it is important to report such an effort as this will help future researchers starting from information regarding the set-up and initial measurement observations. The section below provides a brief background of visual SLAM, the new experimental set-up and observation of initial robot pose measurement data.

7.2.1 Telepresence robots with visual SLAM

The work aimed at navigating Beam plus telepresence robot using ROS along with a camera attached to it which can provide a robot pose in an unknown environment. For this purpose, we rely on existing ORB-SLAM2 and capture robot pose which is essentially a delayed and noisy measurement of the camera sensor. The original intention was to apply the AS-EKF algorithm on such noisy camera measurements.

7.2.1.1 Visual Simultaneous Localization and Mapping

State estimation in an unknown environment is very challenging without knowledge about the environment and the current pose of the robot. Visual SLAM became a very practical approach to solving this problem. A complete visual SLAM framework consists of four parts: *(1)* tracking front end, *(2)* optimization back end, *(3)* loop closure detection and *(4)* map construction. The implementation method of Visual Odometry is divided into *(A)* feature-based method and *(B)* direct method.

Davison *et al.* [165] first proposed a filter-based monocular visual SLAM (Mono SLAM) system that estimates sparse feature points and camera pose. The nonlinear error model and large computations restricted its application. Strasdat *et al.* [166] demonstrated that key frame bundle adjustment-based techniques which are more accurate per computational time than filtering. Klein and Murray [151]

proposed Parallel tracking and mapping (PTAM) system which is a feature-based SLAM algorithm based on nonlinear optimization back end. Mur-Artal *et al.* [167] improved PTAM and proposed the ORB feature-based slam system, called ORB-SLAM. The system is fast for real-time accurate tracking and mapping. Engel *et al.* [168] proposed a large-scale direct monocular SLAM (LSD-SLAM). Compared to other existing direct methods, it reconstructs large-scale semi-dense maps and highly accurate three-dimensional maps in real time.

Newcombe *et al.* [169] integrate all the depth data and image information from Kinect into a dense volumetric model to reconstruct the 3D model of the global map. Henry *et al.* [170] use a joint optimization algorithm to apply RGB-D cameras to the robot field in indoor environments. Kerl *et al.* [171] proposed a visual SLAM method based on a direct dense RGB-D camera. This method combines dense tracking with keyframe selection and poses graph optimization that minimizes the photometric and depth error. Compared to the feature-based method, the direct method is faster and directly recover the camera pose. It has high robustness to the photo-metric error of the image without feature extraction. However, in the case of geometric noise, the algorithm performance decreases quickly.

In this work, we used a recent ORB-SLAM method proposed by Mur-Artal *et al.* [172] due to its comprehensive SLAM framework which is capable of real-time parallel tracking, mapping, loop closing and re-localization. Therefore, this method is efficient for mapping and localizing a wide range of environments and is suitable for our telepresence robot navigation in an unknown environment.

7.2.1.2 Methodology

In this work, we use a camera mounted on top of our telepresence robot and apply the video feed to a visual SLAM algorithm to measure the robot's position. As this measurement is often noisy and time-lagged or delayed, this does not provide the true state of the robot. In order to improve the state estimation one could apply the proposed AS-EKF algorithm on the measured states and compare the

performance with real robot position captured using a multi-camera VICON setup. In this context, it is sensible that firstly we summarize the ORB-SLAM2 algorithm.

ORB-SLAM2: ORB-SLAM2 [172] which uses monocular, stereo and RGB-D cameras contain three threads which work in parallel, tracking, local mapping and loop closing. The tracking thread is used to localize the camera with the extracted features for every frame and to the local map and minimize the re-projection error by applying motion-only bundle adjustment. The second thread local mapping is used to manage and optimise the local map through local bundle adjustment. The last thread is loop closing which performs pose-graph optimization to correct the drift and detect large loops. After loop closing a fourth thread was introduced to perform full bundle adjustment of the entire map to compute the optimal structure and motion solution.

ORB-SLAM2 embedded a place recognition module for re-localization, if the system has lost track of where it is or for re-initializes if there is already a mapped scene. Another important aspect of ORB-SLAM2 is the co-visibility graph, which is used to link any two key-frames that have similar observations of points. The graph structure is used to retrieve local windows of key-frames which enables the tracking and local mapping to operate locally. For tracking, mapping and visual place recognition the ORB-SLAM2 uses ORB features, which are robust to capture environmental changes and fast to extract and match allowing for real-time operation. We capture the output from key-frame mapping to extract the robot position including x,y,z and θ which then can be used as the input to the AS-EKF framework for further processing. An example output from ORB-SLAM2, captured during our experiment in the lab environment is shown in Figure 7.1.

Through the ORB-SLAM2 we predicted the camera pose and hence the robot pose which is used as measurement data in the AS-EKF algorithm. With the empirical evidence, we assume that there is an amount of time difference between sending a control command and the moment when the measurement data enters the filter.

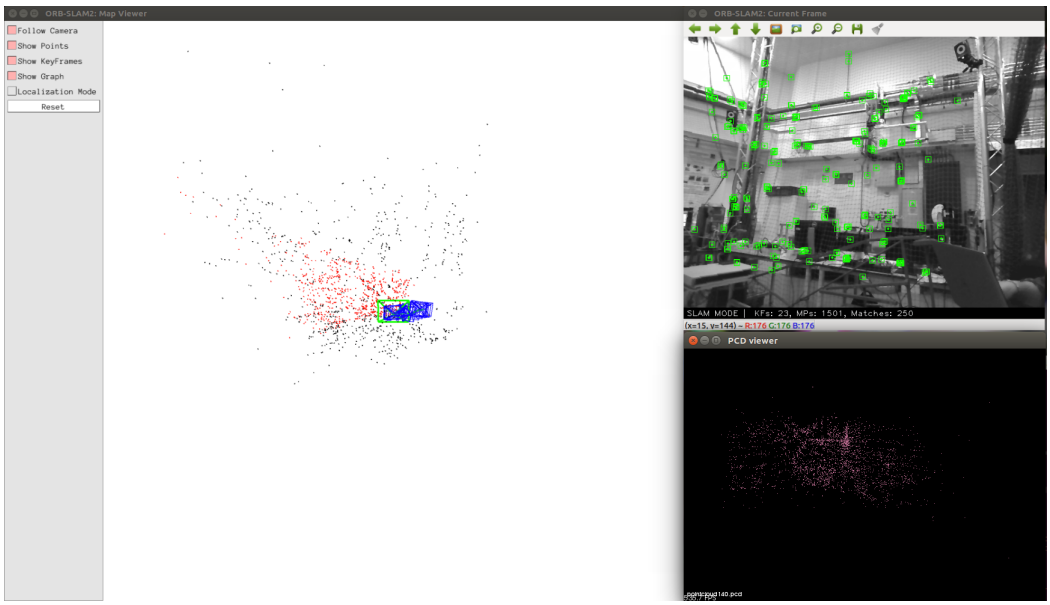


Figure 7.1: Example output from ORB-SLAM2 captured in the lab environment during the experiment. The left-hand column shows the robot motion history, row 1 of column 2 is the reference points detected by SLAM and row 2 is the point cloud view.

7.2.1.3 Experimental framework

In this work, we have used a Beam plus differential drive mobile robot. As described in earlier chapters the Beam plus robot produced an enormous amount of dead-reckoning errors during experimental navigation which are corrected with the UMBMark method. ROS is used to navigate and control the robot.

This setup is now extended to enable visual SLAM-based robot pose measurement by incorporating a 3D camera sensor. We captured camera data using ASUS Xtion2 3D sensor [173] mounted on top of our Beam plus. The experimental set-up for Beam plus and the camera is shown in Figure 7.2. The camera was connected through the ROS packages with the host computer. The camera was used to map the environment and extract the robot's pose. We captured the robot's absolute navigation data using the VICON motion capture system. We have attached some retro-reflective markers on the robot to represent it as a rigid body. VICON cameras were used to record the motion movement of the robot.

The proposed algorithm was implemented in a Linux-based computer as a host

Figure 7.2: Telepresence robot Beam plus with ASUS Xtion-2 camera mounted for visual SLAM measurement in our experiment.

computer. The host computer is connected to the telepresence robot using the ROS driver which created a ROS bridge between them. Through the ROS bridge, we send velocity commands to navigate the robot. The control command directed the robot to form a one-by-half-meter raster-scan navigation path. In order to localize the robot's position, the Xtion2 camera was used. The camera was attached to the robot head and using the ORB-SLAM2 method the lab environment was mapped and the robot's location was tracked through the camera position. The VICON camera which captured absolute navigation data (assuming negligible variance) of the robot was used to confirm the accessibility of the proposed algorithm. All the experimental works were carried out in a real laboratory environment. The overall experimental framework is depicted in Figure 7.3.

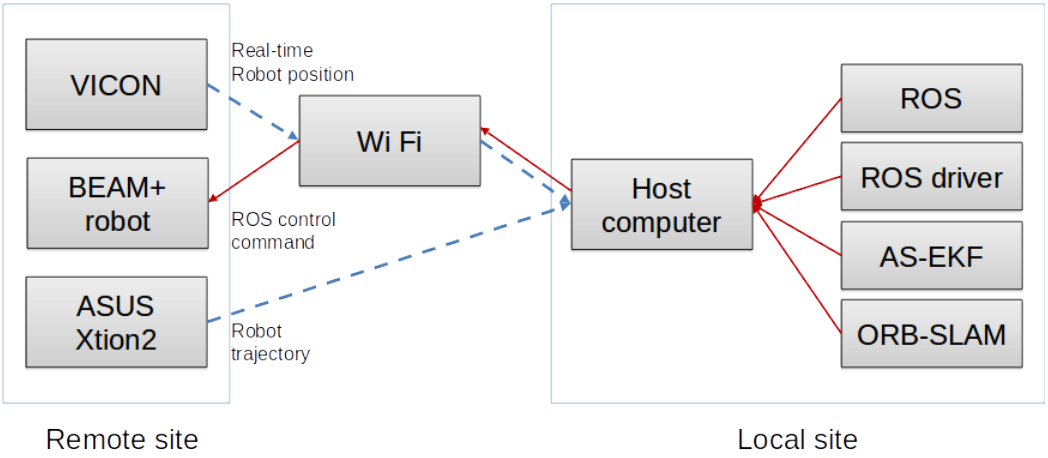


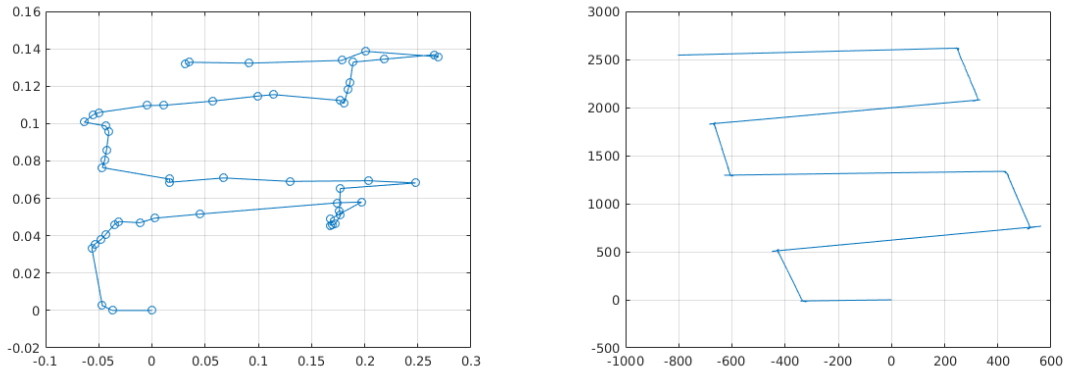
Figure 7.3: Visual SLAM experimental framework with a state-of-the-art off-the-shelve telepresence robot Beam plus.

7.2.2 Discussions and future work

With the initial set up we acquired experimental SLAM and corresponding VICON data. An example of data is shown in Figure 7.4. The observation indicates erroneous measurements of robot pose using the visual SLAM. We did not apply any corrective algorithm which is considered to be future work. Based on the initial observation following research directions are advised:

- Measurement data point using visual SLAM is extremely limited (<100) compared to the number of VICON data points (>10,000). Thus one can consider data augmentation through various interpolation techniques. This would potentially improve the quality of the input measurement data to any filtering algorithm.

- Due to uncertain and challenging environments sometimes SLAM fails to generate any measurement points or even produces extremely noisy and erroneous measurements. It is particularly observable when there is a net in the surroundings and therefore SLAM can not estimate a robust reference point. One potential solution to make up for the missing data is to improve the SLAM algorithm [174] or change the lighting conditions which also impact considerably. However, major improvements could be achieved using a



a) SLAM trajectory output

b) VICON output

Figure 7.4: Robot trajectory using SLAM output and real position using VICON. SLAM output shows a limited number of poses calculated using ORB-SLAM2 while VICON has dense measurement.

better filtering approach or even emerging deep learning [175].

- Due to major computation complexity and other factors such as communication delay there could be a time delay. Modelling such a time delay would be very interesting and challenging. Once modelled it could be incorporated within the AS-EKF algorithm in order to offer a potential solution towards robust robot pose prediction.

- Although we have focused on the time delay issue in the telepresence system, many other factors that contribute to positional error require to be factored in any proposed algorithm. Therefore, it would make better sense to propose a joint delay compensation and positional error correction filtering algorithm by modelling the contributing factors.

- Finally, it is worth investigating other sensors (including ultrasound sensors or GPS) which can provide robot pose information. However, this will depend on the application-specific scenarios, *e.g.,* , at present GPS based systems are not effective in indoor scenarios.

Bibliography

[1] K. M. Tsui, M. Desai, H. A. Yanco, and C. Uhlik, "Exploring use cases for telepresence robots," in *Proceedings of the 6th international conference on Human-robot interaction*. ACM, 2011, pp. 11–18.

[2] I. Double Robotics, "Double 3," 2022, accessed: 2022-06-04. [Online]. Available: https://www.doublerobotics.com/double3.html

[3] R. Held and N. Durlach, "Telepresence, time delay and adaptation," *Pictorial communication in virtual and real environments*, pp. 232–246, 1991.

[4] "Interplanetary Internet," https://marspedia.org/Interplanetary_Internet#:~:text=on%20its%20return.-,Latency,minutes%20in%20bi-directional%20communications, accessed: 2023-01-01.

[5] "Network Latency - Common Causes and Best Solutions," https://www.ir.com/guides/what-is-network-latency, accessed: 2023-01-01.

[6] J. Y. C. Chen, E. C. Haas, and M. J. Barnes, "Human performance issues and user interface design for teleoperated robots," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1231–1245, 2007.

[7] J. S. Tittle, A. Roesler, and D. D. Woods, "The remote perception problem," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 46, no. 3, pp. 260–264, 2002.

[8] H. Dybvik, M. Løland, A. Gerstenberg, K. B. Slåttsveen, and M. Steinert, "A low-cost predictive display for teleoperation: Investigating effects on human performance and workload," *International Journal of Human-Computer Studies*, vol. 145, p. 102536, 2021.

[9] M. S. P. de Melo, J. G. da Silva Neto, P. J. L. da Silva, J. M. X. N. Teixeira, and V. Teichrieb, “Analysis and comparison of robotics 3d simulators,” in *2019 21st Symposium on Virtual and Augmented Reality (SVR)*. IEEE, 2019, pp. 242–251.

[10] M. Hollier, A. Rimmell, and D. Burraston, “Spatial audio technology for telepresence,” *BT Technology Journal*, vol. 15, no. 4, pp. 33–41, 1997.

[11] A. Valiton and Z. Li, “Perception-action coupling in usage of telepresence cameras,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3846–3852.

[12] A. Casals, “Assisted teleoperation through the merging of real and virtual images,” in *Robotics Research. The Eleventh International Symposium*, ser. Springer Tracts in Advanced Robotics, P. Dario and R. Chatila, Eds. Springer Berlin Heidelberg, 2005, vol. 15, pp. 135–144.

[13] P. F. Hokayem and M. W. Spong, “Bilateral teleoperation: An historical survey,” *Automatica*, vol. 42, no. 12, pp. 2035–2057, 2006.

[14] A. Kristoffersson, S. Coradeschi, and A. Loutfi, “A review of mobile robotic telepresence,” *Advances in Human-Computer Interaction*, vol. 2013, pp. 1–17, 2013.

[15] L. Almeida, P. Menezes, and J. Dias, “Telepresence social robotics towards co-presence: A review,” *Applied Sciences*, vol. 12, no. 11, p. 5557, 2022.

[16] S. Lichiardopol, “A survey on teleoperation,” *Dept. Mech. Eng., Dynamics Control Group, Technische Universiteit Eindhoven, Eindhoven, Dept., Mech. Eng., Dyn. Control Group, The Netherlands, Tech. Rep. DCT2007*, vol. 155, 2007.

[17] I. Spectrum, “Kobra,” May 2018, accessed: 2022-06-06. [Online]. Available: https://robots.ieee.org/robots/kobra/?gallery=photo1

[18] Army Technology, “710 Kobra multi-mission robot,” Nov 2015, accessed: 2022-02-12. [Online]. Available: https://www.army-technology.com/projects/irobot-710-kobra-multi-mission-robot/

[19] A. Witze, “Space rovers in record race,” *Nature*, vol. 498, pp. 284–285, 2013.

[20] J. A. E. Agency, “About Engineering Test Satellite VII "KIKU-7" (ETS-VII),” https://global.jaxa.jp/projects/sat/ets7/, 1997, [Online; accessed 31-December-2021].

[21] M. Garcia, “Mobile servicing system,” Oct 2018, accessed: 2022-02-12. [Online]. Available: https://www.nasa.gov/mission_pages/station/structure/elements/mobile-servicing-system.html

[22] M. exploration program, “Mars Curiosity Rover,” https://mars.nasa.gov/msl/home/, 2011, [Online; accessed 30-December-2021].

[23] “The Solar Powered Under Water Robot,” https://thefutureofthings.com/5342-the-solar-powered-under-water-robot/, accessed: 2022-06-09.

[24] Oceaneering, “Rov systems,” 2017, accessed: 2022-03-05. [Online]. Available: http://www.oceaneering.com/rovs/rov-systems/millennium-plus-rov/

[25] U. of South Florida, “Tavros: Submersible, autonomous seatweeting robots!” 2012, accessed: 2022-03-03. [Online]. Available: https://web.archive.org/web/20131107220816/http://www.marine.usf.edu/systems/

[26] U. S. A. Force, “Mq-1b predator,” 1994, accessed: 2022-03-05. [Online]. Available: https://www.af.mil/About-Us/Fact-Sheets/Display/Article/104469/mq-1b-predator/

[27] GlobalSecurity.org, “Gladiator tactical unmanned ground vehicle,” 2022, accessed: 2022-06-04. [Online]. Available: https://www.globalsecurity.org/military/systems/ground/gladiator.htm

[28] R. Sherman, “An/slq-48 - mine neutralization vehicle,” 1999, accessed: 2022-06-04. [Online]. Available: https://man.fas.org/dod-101/sys/ship/weaps/an-slq-48.htm

[29] “Dragon Runner Small & Compact Robot,” https://www.qinetiq.com/en-us/what-we-do/services-and-products/dragon-runner-small-and-compact-robot, accessed: 2022-06-09.

[30] “iRobot 510 PackBot Multi-Mission Robot,” https://www.army-technology.com/projects/irobot-510-packbot-multi-mission-robot/, accessed: 2022-06-09.

[31] “da Vinci Surgical System,” https://www.davincisurgery.com/da-vinci-systems/about-da-vinci-systems, accessed: 2022-06-09.

[32] “Cooper Gray Robotics launches remote controlled, electric skid steer,” https://www.oemoffhighway.com/drivetrains/news/12133515/cooper-gray-robotics-launches-remote-controlled-electric-skid-steer, accessed: 2022-06-09.

[33] “Loccioni a Londra con Felix,” https://www.automazionenews.it/loccioni-a-londra-con-felix/, accessed: 2022-06-09.

[34] P. Debenest, M. Guarnieri, K. Takita, E. F. Fukushima, S. Hirose, K. Tamura, A. Kimura, H. Kubokawa, N. Iwama, and F. Shiga, “Expliner-robot for inspection of transmission lines,” in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 3978–3984.

[35] T. P. Sattar, S. Chen, B. Bridge, and J. Shang, “Robair: mobile robotic system to inspect aircraft wings and fuselage,” *Robotics and Factories of the Future*, vol. 2, pp. 795–802, 2003.

[36] “Introducing the NEW EX5600-7,” https://www.hitachicm.eu/machinery/excavators/large-excavators/ex5600-7/, accessed: 2022-06-09.

[37] “Mining Trucks,” https://www.cat.com/en_GB/products/new/equipment/off-highway-trucks/mining-trucks.html, accessed: 2022-06-09.

[38] “What is a Telepresence Robot and what can they do?” https://telepresencerobots.com/what-telepresence-robot-and-what-can-they-do, [Online; accessed 30-December-2021].

[39] “Giraff,” http://www.giraff.org/?lang=en, accessed: 2022-06-09.

[40] J. Yeung and D. Fels, “A remote telepresence system for high school classrooms,” in *Electrical and Computer Engineering, 2005. Canadian Conference on*, May 2005, pp. 1465–1468.

[41] S. C. Herring, “Robot-mediated communication,” *Emerging trends in the social and behavioral sciences: An interdisciplinary, searchable, and linkable resource*, pp. 1–16, 2015.

[42] “Double Robotics’ new telepresence robot brings more speed, stability, and sight,” https://www.theverge.com/2016/1/6/10720812/double-robotics-new-robot-price-specs-ces-2016, accessed: 2022-06-09.

[43] “Your alter ego on wheels,” https://www.economist.com/technology-quarterly/2013/03/09/your-alter-ego-on-wheels, accessed: 2022-06-09.

[44] “Remote presence robots help elderly people live in their own homes for longer and improve patient care in hospitals,” https://ifr.org/news/improving-health/, accessed: 2022-06-09.

[45] “Q&A: Managing a Telepresence Robot in the Classroom,” https://schoolconstructionnews.com/2016/12/21/qa-managing-telepresence-robot-classroom/, accessed: 2022-06-09.

[46] O. Robotics, “ROS - Robot Operating System,” https://www.ros.org/, 2007, [Online; accessed 09-January-2022].

[47] "ROS Kinetic Kame," http://wiki.ros.org/kinetic, accessed: 2022-06-06.

[48] "ROS Beam," https://github.com/xlz/rosbeam, accessed: 2022-06-06.

[49] V. motion systems, "What is motion capture?" https://www.vicon.com/about-us/what-is-motion-capture/, [Online; accessed 31-December-2021].

[50] G. Welch, G. Bishop *et al.*, "An introduction to the kalman filter," 1995.

[51] D. Chen, X. Fu, W. Ding, H. Li, N. Xi, and Y. Wang, "Shifted gamma distribution and long-range prediction of round trip timedelay for internet-based teleoperation," in *2008 IEEE International Conference on Robotics and Biomimetics*, 2009, pp. 1261–1266.

[52] Kim, S., Lee, J. Y., and Sung, D. K., "A shifted gamma distribution model for long-range dependent internet traffic," *IEEE Communications Letters*, vol. 7, no. 3, pp. 124–126, 2003.

[53] J. Hua, Y. Cui, Y. Yang, and H. Li, "Analysis and prediction of jitter of internet one-way time-delay for teleoperation systems," in *11th IEEE International Conference on Industrial Informatics (INDIN)*, 2013, pp. 612–617.

[54] T. T. Andersen, H. B. Amor, N. A. Andersen, and O. Ravn, "Measuring and modelling delays in robot manipulators for temporally precise control using machine learning," in *IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 2015, pp. 168–175.

[55] R. Oboe and P. Fiorini, "Issues on internet-based teleoperation," *IFAC Proceedings Volumes*, vol. 30, no. 20, pp. 591–597, 1997.

[56] A. Afzal, D. S. Katz, C. L. Goues, and C. S. Timperley, "A study on the challenges of using robotics simulators for testing," *arXiv preprint arXiv:2004.07368*, 2020.

[57] "Gazebo," https://gazebosim.org/home, accessed: 2022-06-06.

[58] L. Nogueira, “Comparative analysis between gazebo and v-rep robotic simulators,” *Seminario Interno de Cognicao Artificial-SICA*, vol. 2014, no. 5, 2014.

[59] “RViz,” http://wiki.ros.org/rviz, accessed: 2022-06-06.

[60] K. Kawabata, T. Sekine, and T. Suzuki, “Mobile robot teleoperation system utilizing a virtual world,” *Advanced Robotics*, vol. 15, no. 1, pp. 1–16, 2001.

[61] T. Sekine, T. Suzuki, K. Kawabata, T. Fujii, H. Asama, K. Sato, and I. Endo, “Mobile robot teleoperation system utilizing virtual world,” in *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No. 99CH36289)*, vol. 3. IEEE, 1999, pp. 1727–1732.

[62] T. Kot, J. Babjak, and P. Novák, “Virtual operator station for teleoperated mobile robots,” in *Modelling and Simulation for Autonomous Systems*. Springer, 2015, pp. 144–153.

[63] T. Kot and P. Novák, “Application of virtual reality in teleoperation of the military mobile robotic system taros,” *International journal of advanced robotic systems*, vol. 15, no. 1, 2018.

[64] T. Sanguino, J. M. A. Marquez, T. Carlson, and J. del R M., “Interaction and evaluation of an augmented virtuality assistance system for teleoperated robots,” in *Robotic and Sensors Environments (ROSE), 2012 IEEE International Symposium*. IEEE, 2012, pp. 19–24.

[65] T. M. Sanguino, J. A. Márquez, T. Carlson, and J. d. R. Millán, “Improving skills and perception in robot navigation by an augmented virtuality assistance system,” *Journal of Intelligent & Robotic Systems*, vol. 76, no. 2, pp. 255–266, 2014.

[66] T. Jin, J. Lee, and H. Hashimoto, “Internet-based obstacle avoidance of mobile robot using a force-reflection,” in *Intelligent Robots and Systems, 2004.(IROS*

2004). Proceedings. 2004 IEEE/RSJ International Conference on, vol. 4. IEEE, 2004, pp. 3418–3423.

[67] K. Kominami, T. Takubo, K. Ohara, Y. Mae, and T. Arai, "Optimization of obstacle avoidance using reinforcement learning," in *2012 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2012, pp. 67–72.

[68] J. Salmeron-Garcia, P. Inigo-Blasco, F. Diaz-del Rio, and D. Cagigas-Muniz, "A tradeoff analysis of a cloud-based robot navigation assistant using stereo image processing," *Automation Science and Engineering, IEEE Transactions on*, vol. 12, no. 2, pp. 444–454, April 2015.

[69] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Transactions on automation science and engineering*, vol. 12, no. 2, pp. 398–409, 2015.

[70] J. Wan, S. Tang, H. Yan, D. Li, S. Wang, and A. V. Vasilakos, "Cloud robotics: Current status and open issues," *IEEE Access*, vol. 4, pp. 2797–2807, 2016.

[71] M. Colledanchise, D. V. Dimarogonas, and P. Ogren, "Robot navigation under uncertainties using event based sampling," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, 2014, pp. 1438–1445.

[72] C. Santos, M. Martínez-Rey, F. Espinosa, A. Gardel, and E. Santiso, "Event-based sensing and control for remote robot guidance: An experimental case," *Sensors*, vol. 17, no. 9, p. 2034, 2017.

[73] F. Janabi-Sharifi and I. Hassanzadeh, "Experimental analysis of mobile-robot teleoperation via shared impedance control," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 2, pp. 591–606, 2010.

[74] S. Islam, P. Liu, and A. El Saddik, "Nonlinear control for teleoperation systems with time varying delay," *Nonlinear Dynamics*, vol. 76, no. 2, pp. 931–954, 2014.

[75] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer Science & Business Media, 2008.

[76] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, "Modelling, planning and control," *Advanced Textbooks in Control and Signal Processing. Springer,*, 2009.

[77] C. Armbrust, M. Proetzsch, B. Schäfer, and K. Berns, "A behaviour-based integration of fully autonomous, semi-autonomous, and tele-operated control modes for an off-road robot," in *Proceedings of the 2nd IFAC Symposium on Telematics Applications, Timisoara, Romania*, 2010.

[78] C. Martínez and F. Jiménez, "Implementation of a potential field-based decision-making algorithm on autonomous vehicles for driving in complex environments," *Sensors*, vol. 19, no. 15, p. 3318, 2019.

[79] J. B. Park, J. H. Lee, and B. H. Lee, "Rollover-free navigation for a mobile agent in an unstructured environment," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 36, no. 4, pp. 835–848, 2006.

[80] J. G. Wildenbeest, D. A. Abbink, C. J. Heemskerk, F. C. Van Der Helm, and H. Boessenkool, "The impact of haptic feedback quality on the performance of teleoperated assembly tasks," *IEEE Transactions on Haptics*, vol. 6, no. 2, pp. 242–252, 2012.

[81] J. Du, C. Mouser, and W. Sheng, "Design and evaluation of a teleoperated robotic 3-d mapping system using an rgb-d sensor," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, no. 99, pp. 1–7, 2015.

[82] K. Huang, D. Chitrakar, F. Rydén, and H. J. Chizeck, "Evaluation of haptic guidance virtual fixtures and 3d visualization methods in telemanipulation—a user study," *Intelligent Service Robotics*, vol. 12, no. 4, pp. 289–301, 2019.

[83] S. David, T. Giles, J. S. Ian, and R. David, "Simple expert systems to improve an ultrasonic sensor system for a tele operated mobile robot," *Sensor Review*, vol. 31, no. 3, pp. 246–260, 2011.

[84] D. Sanders, D. Ndzi, S. Chester, and M. Malik, “Adjustment of tele-operator learning when provided with different levels of sensor support while driving mobile robots,” in *Proceedings of SAI Intelligent Systems Conference*. Springer, 2016, pp. 548–558.

[85] A. Mora, D. G. Axelson, M. Chacin, K. Nagatani, and K. Yoshida, “Assisted tele-operated navigation system based on 3d mapping,” in *Systems Conference, 2007 1st Annual IEEE*. IEEE, 2007, pp. 1–6.

[86] H. Hu, C. Quintero, H. Sun, and M. Jagersand, “On-line reconstruction based predictive display in unknown environment,” in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4446–4451.

[87] J.-H. Lee, J.-B. Park, and B.-H. Lee, “Turnover prevention of a mobile robot on uneven terrain using the concept of stability space,” *Robotica*, vol. 27, no. 5, pp. 641–652, 2009.

[88] J. B. Park, “Multiple mobile robot path planning for rollover prevention and collision avoidance,” in *2011 11th International Conference on Control, Automation and Systems*. IEEE, 2011, pp. 1732–1734.

[89] S. Kim, C. Roh, K. Sung-Chul, and M. Park, “A hybrid autonomous / teleoperated strategy for reliable mobile robot outdoor navigation,” *SICE-ICASE International Joint Conference*, no. c, pp. 3120–3125, 2006.

[90] S. Alers, D. Bloembergen, D. Claes, J. Fossel, D. Hennes, and K. Tuyls, “Telepresence robots as a research platform for ai,” in *2013 AAAI Spring Symposium Series*. Citeseer, 2013.

[91] M. Kuderer, H. Kretzschmar, and W. Burgard, “Teaching mobile robots to cooperatively navigate in populated environments,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 3138–3143.

[92] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 335–356, 2015.

[93] C. Cao, P. Trautman, and S. Iba, "Dynamic channel: A planning framework for crowd navigation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5551–5557.

[94] M. Ollis, W. Huang, and M. Happold, "A Bayesian approach to imitation learning for robot navigation," *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 709–714, 2007.

[95] A. Kleiner and C. Dornhege, "Operator-assistive mapping in harsh environments," *Proc. of the IEEE Int Workshop on Safety Security and*, pp. 1–6, 2006.

[96] N. Kubota, Y. Toda, and B. H. L., "Multifeatured visualization and navigation in tele-operation of mobile robots," in *Robotic Intelligence In Informationally Structured Space (RiiSS), 2011 IEEE Workshop on*, April 2011, pp. 85–92.

[97] J. Borenstein, R. Miller, and A. Borrell, "Teloptrak: Heuristics-enhanced indoor location tracking for tele-operated robots," *Journal of Navigation*, vol. 65, no. 02, pp. 265–279, 2012.

[98] H. Sun, Y. Zhang, Z. Wu, and J. Xue, "The detecting robot based on sopc," in *2014 IEEE International Conference on Mechatronics and Automation*, 2014, pp. 1935–1939.

[99] A. Hace and M. Franc, "Fpga implementation of sliding-mode-control algorithm for scaled bilateral teleoperation," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1291–1300, 2013.

[100] D. Portugal, M. E. Andrada, A. G. Araújo, M. S. Couceiro, and J. F. Ferreira, *ROS Integration of an Instrumented Bobcat T190 for the SEMFIRE Project*. Springer International Publishing, 2021, pp. 87–119.

[101] A. Hace and M. Franc, *FPGA-Based Haptic Teleoperation*. Springer International Publishing, 2014, pp. 145–159.

[102] K. W. Chung, L. T. Lam, and Y. Xu, "A natural assisted navigation motion for telepresence robots," in *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on*, Dec 2014, pp. 2268–2273.

[103] Y. Sato, K. Hashimoto, and Y. Shibata, "A new remote camera work system for teleconference using a combination of omni-directional and network controlled cameras," in *Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on*, March 2008, pp. 502–508.

[104] R. J. Anderson and M. W. Spong, "Bilateral control of teleoperators with time delay," *IEEE Transactions on Automatic Control*, vol. 34, no. 5, pp. 494–501, 1989.

[105] I. Vittorias, J. Kammerl, S. Hirche, and E. Steinbach, "Perceptual coding of haptic data in time-delayed teleoperation," in *World Haptics 2009 - Third Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 2009, pp. 208–213.

[106] J. Funda, T. S. Lindsay, and R. P. Paul, "Teleprogramming: Toward delay-invariant remote manipulation," *Presence: Teleoperators and Virtual Environments*, vol. 1, no. 1, pp. 29–44, 1992.

[107] M. Hernando and E. Gambao, "A robot teleprogramming architecture," in *Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, vol. 2, 2003, pp. 1113–1118.

[108] K. Brady and T. . Tarn, "Internet-based teleoperation," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 1, 2001, pp. 644–649.

[109] J. Bolarinwa, A. Smith, A. Aijaz, A. Stanoev, M. Sooriyabandara, and M. Giuliani, "Haptic teleoperation goes wireless: Evaluation and benchmarking of

a high-performance low-power wireless control technology," *arXiv preprint arXiv:2210.07212*, 2022.

[110] G. Dogangil, B. Davies, and F. Rodriguez y Baena, "A review of medical robotics for minimally invasive soft tissue surgery," *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, vol. 224, no. 5, pp. 653–679, 2010.

[111] K. Natori, T. Tsuji, K. Ohnishi, A. Hace, and K. Jezernik, "Time-delay compensation by communication disturbance observer for bilateral teleoperation under time-varying delay," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 3, pp. 1050–1062, 2010.

[112] A. Bejczy, W. Kim, and S. Venema, "The phantom robot: predictive displays for teleoperation with time delay," in *Proceedings., IEEE International Conference on Robotics and Automation*, vol. 1, 1990, pp. 546–551.

[113] J. Kikuchi, K. Takeo, and K. Kosuge, "Teleoperation system via computer network for dynamic environment," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 4, 1998, pp. 3534–3539.

[114] E. Slawiñski, V. Mut, L. Salinas, and S. García, "Teleoperation of a mobile robot with time-varying delay and force feedback," *Robotica*, vol. 30, no. 1, p. 67–77, 2012.

[115] E. Slawiñski, V. Moya, D. Santiago, and V. Mut, "Force and position–velocity coordination for delayed bilateral teleoperation of a mobile robot," *Robotica*, vol. 37, no. 10, p. 1768–1784, 2019.

[116] "Beam," https://suitabletech.com/support/welcome, accessed: 2022-06-09.

[117] R. Summan, S. Pierce, C. Macleod, G. Dobie, T. Gears, W. Lester, P. Pritchett, and P. Smyth, "Spatial calibration of large volume photogrammetry based metrology systems," *Measurement*, vol. 68, pp. 189 – 200, 2015.

[118] "VICON Technical Information," https://www.vicon.com/hardware/cameras/vantage/, accessed: 2023-01-26.

[119] E. Ivanjko, M. Vasak, and I. Petrovic, "Kalman filter theory based mobile robot pose tracking using occupancy grid maps," in *International Conference on Control and Automation (ICCA2005)*, vol. 2, 2005, pp. 869–874.

[120] J. Borenstein and L. Feng, "UMBmark: A benchmark test for measuring odometry errors in mobile robots," in *SPIE Mobile Robots X*, vol. 2591, 1995, pp. 113–125.

[121] "STAIR: STanford Artificial Intelligence Robot," http://stair.stanford.edu/index.php, accessed: 2023-01-07.

[122] "Personal Robotics Program," http://personalrobotics.stanford.edu/, accessed: 2023-01-07.

[123] trojrobert, "Hands-On Introduction to Robot Operating System(ROS)," https://trojrobert.github.io/hands-on-introdution-to-robot-operating-system(ros)/, 2020, [Online; accessed 04-January-2022].

[124] P. Merriaux, Y. Dupuis, R. Boutteau, P. Vasseur, and X. Savatier, "A study of vicon system positioning performance," *Sensors*, vol. 17, no. 7, p. 1591, Jul 2017.

[125] "Beam Technical Information," https://suitabletech.com/images/stories/Documentation/files/12-2017-Beam-Dimensions.pdf, accessed: 2023-01-26.

[126] M. Choi, J. Choi, and W. Chung, "State estimation with delayed measurements incorporating time-delay uncertainty," *IET Control Theory and Applications*, vol. 6, pp. 2351–2361(10), Oct 2012.

[127] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[128] M. Bak, T. D. Larsen, M. Norgaard, N. A. Andersen, N. Poulsen, and O. Ravn, "Location estimation using delayed measurements," in *5th International Workshop on Advanced Motion Control, 1998. AMC '98-Coimbra, 1998*, Jun 1998, pp. 180–185.

[129] M. Shin, M. Park, D. Oh, B. Kim, and J. Lee, "Clock synchronization for one-way delay measurement: A survey," in *International Conference on Advanced Communication and Networking*. Springer, 2011, pp. 1–10.

[130] "IP Latency Statistics," https://www.verizon.com/business/terms/latency/, accessed: 2023-02-14.

[131] Y. Bar-Shalom, "Update with out-of-sequence measurements in tracking: exact solution," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 3, pp. 769–777, Jul 2002.

[132] Y. Bar-Shalom, H. Chen, and M. Mallick, "One-step solution for the multi-step out-of-sequence-measurement problem in tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 1, pp. 27–37, Jan 2004.

[133] T. D. Larsen, N. A. Andersen, O. Ravn, and N. K. Poulsen, "Incorporation of time delayed measurements in a discrete-time kalman filter," in *Proceedings of the 37th IEEE Conference on Decision and Control*, vol. 4, Dec 1998, pp. 3972–3977.

[134] S. Challa, R. J. Evans, and X. Wang, "A bayesian solution and its approximations to out-of-sequence measurement problems," *Information Fusion*, vol. 4, no. 3, pp. 185 – 199, 2003.

[135] R. Van Der Merwe, E. A. Wan, and S. Julier, "Sigma-point kalman filters for nonlinear estimation and sensor-fusion: Applications to integrated naviga-

tion," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2004, pp. 16–19.

[136] S. J. Julier and J. K. Uhlmann, "Fusion of time delayed measurements with uncertain time delays," in *Proceedings of the 2005, American Control Conference.*, vol. 6, June 2005, pp. 4028–4033.

[137] J. Hu, Z. Wang, F. E. Alsaadi, and T. Hayat, "Event-based filtering for time-varying nonlinear systems subject to multiple missing measurements with uncertain missing probabilities," *Information Fusion*, vol. 38, pp. 74–83, 2017.

[138] L. Zou, Z. Wang, Q.-L. Han, and D. Zhou, "Recursive filtering for time-varying systems with random access protocol," *IEEE Transactions on Automatic Control*, vol. 64, no. 2, pp. 720–727, 2018.

[139] B. Das, G. Dobie, and S. G. Pierce, "State estimation of delays in telepresence robot navigation using bayesian approaches," in *19th Annual Conference, Towards Autonomous Robotic Systems Conference (TAROS)*, July 2018.

[140] ——, "AS-EKF: a delay aware state estimation technique for telepresence robot navigation," in *IEEE International Conference on Robotic Computing (IRC)*, Feb 2019.

[141] B. Das and G. Dobie, "Delay compensated state estimation for telepresence robot navigation," *Robotics and Autonomous Systems*, vol. 146, p. 103890, 2021.

[142] A. H. Jazwinski, *Stochastic processes and filtering theory*. Courier Corporation, 2007.

[143] H. Ahmad and T. Namerikawa, "Extended kalman filter-based mobile robot localization with intermittent measurements," *Systems Science & Control Engineering: An Open Access Journal*, vol. 1, no. 1, pp. 113–126, 2013.

[144] J. Jin, L. Petrich, S. He, M. Dehghan, and M. Jagersand, “Long range teleoperation for fine manipulation tasks under time-delay network conditions,” *arXiv preprint arXiv:1903.09189*, 2019.

[145] A. Matheson, B. Donmez, F. Rehmatullah, P. Jasiobedzki, H.-K. Ng, V. Panwar, and M. Li, “The effects of predictive displays on performance in driving tasks with multi-second latency: Aiding tele-operation of lunar rovers,” *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 57, no. 1, pp. 21–25, 2013.

[146] A. Rachmielowski, N. Birkbeck, and M. Jägersand, “Performance evaluation of monocular predictive display,” in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 5309–5314.

[147] T. Burkert, J. Leupold, and G. Passig, “A photorealistic predictive display,” *Presence*, vol. 13, no. 1, pp. 22–43, 2004.

[148] D. Cobzas, M. Jagersand, and H. Zhang, “A panoramic model for remote robot environment mapping and predictive display,” *International Journal of Robotics and Automation*, vol. 20, no. 1, pp. 25–33, 2005.

[149] K. Yerex, D. Cobzas, and M. Jagersand, “Predictive display models for telemanipulation from uncalibrated camera-capture of scene geometry and appearance,” in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 2. IEEE, 2003, pp. 2812–2817.

[150] D. Lovi, N. Birkbeck, A. H. Herdocia, A. Rachmielowski, M. Jägersand, and D. Cobzaş, “Predictive display for mobile manipulators in unknown environments using online vision-based monocular modeling and localization,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2010, pp. 5792–5798.

[151] G. Klein and D. Murray, “Parallel tracking and mapping for small ar workspaces,” in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, Nov 2007, pp. 225–234.

[152] D. Lovi, N. Birkbeck, D. Cobzas, and M. Jagersand, “Incremental free-space carving for real-time 3d reconstruction,” in *Fifth international symposium on 3D data processing visualization and transmission (3DPVT)*, 2010.

[153] A. Kelly, N. Chan, H. Herman, D. Huber, R. Meyers, P. Rander, R. Warner, J. Ziglar, and E. Capstick, “Real-time photorealistic virtualized reality interface for remote mobile robot control,” *The International Journal of Robotics Research*, vol. 30, no. 3, pp. 384–404, 2011.

[154] K. Schmid, T. Tomic, F. Ruess, H. Hirschmüller, and M. Suppa, “Stereo vision based indoor/outdoor navigation for flying robots,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 3955–3962.

[155] H. Hu, C. Perez, H.-X. Sun, and M. Jagersand, “Performance of predictive display teleoperation under different delays with different degree of freedoms,” in *2016 International Conference on Information System and Artificial Intelligence (ISAI)*. IEEE, 2016, pp. 380–384.

[156] J. Orlosky, K. Theofilis, K. Kiyokawa, and Y. Nagai, “Effects of throughput delay on perception of robot teleoperation and head control precision in remote monitoring tasks,” *PRESENCE: Virtual and Augmented Reality*, vol. 27, no. 2, pp. 226–241, 2020.

[157] R. K. Megalingam, C. R. Teja, S. Sreekanth, and A. Raj, “Ros based autonomous indoor navigation simulation using slam algorithm,” *International Journal of Pure and Applied Mathematics*, vol. 118, no. 7, pp. 199–205, 2018.

[158] “Open Dynamics Engine.” https://www.ode.org/, accessed: 2022-06-09.

[159] “Bullet Real-Time Physics Simulation.” https://pybullet.org/wordpress/, accessed: 2022-06-09.

[160] “Simbody: Multibody Physics API.” https://simtk.org/projects/simbody, accessed: 2022-06-09.

[161] “Dynamic Animation and Robotics Toolkit.” https://dartsim.github.io/, accessed: 2022-06-09.

[162] “Open-source graphics rendering engines,” https://www.ogre3d.org/, accessed: 2022-06-09.

[163] K. Takaya, T. Asai, V. Kroumov, and F. Smarandache, “Simulation environment for mobile robots testing using ros and gazebo,” in *20th International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2016, pp. 96–101.

[164] M. Quigley, B. Gerkey, and W. D. Smart, *Programming Robots with ROS: a practical introduction to the Robot Operating System*. " O’Reilly Media, Inc.", 2015.

[165] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, “Monoslam: Real-time single camera slam,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, June 2007.

[166] H. Strasdat, J. M. M. Montiel, and A. J. Davison, “Visual slam: Why filter?” *Image Vision Comput.*, vol. 30, pp. 65–77, 2012.

[167] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, “Orb-slam: A versatile and accurate monocular slam system,” *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct 2015.

[168] J. Engel, T. Schöps, and D. Cremers, “Lsd-slam: Large-scale direct monocular slam,” in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 834–849.

[169] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, “Kinectfusion: Real-time

dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, Oct 2011, pp. 127–136.

[170] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[171] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 2100–2106.

[172] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, Oct 2017.

[173] "Asus Xtion 2," https://www.asus.com/ch-en/Networking-IoT-Servers/Smart-Home/Security-Camera/Xtion-2/, accessed: 2022-06-06.

[174] S. Milz, G. Arbeiter, C. Witt, B. Abdallah, and S. Yogamani, "Visual slam for automated driving: Exploring the applications of deep learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 247–257.

[175] A. Staroverov, D. A. Yudin, I. Belkin, V. Adeshkin, Y. K. Solomentsev, and A. I. Panov, "Real-time object navigation with deep neural networks and hierarchical reinforcement learning," *IEEE Access*, vol. 8, pp. 195 608–195 621, 2020.