Mixed Reality Simulation for Mobile Robots

Ian Yen-Hung Chen

July 2011

Supervisors: Assoc. Prof. Bruce MacDonald
Dr. Burkhard Wünsche

Department of Electrical & Computer Engineering
The University of Auckland
New Zealand

A thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering

Abstract

In recent years robots have dramatically improved in functionality, but as a result their designs have also become more complicated. The increase in the complexity of the tasks and the design of the robot result in a challenging and time-consuming robot development process. Sophisticated simulation models, expensive hardware setup, and careful human supervisions are necessary to provide a safe and close-to-real world testbed that helps to draw insights to the actual operation. However, the considerable cost and time for creating well-designed tests and for meeting safety requirements present a barrier to rapid development of complex robot systems.

This thesis analyses the robot development process and identifies areas for improvement. Based on the requirements analysis, it proposes that, by applying the concept of Mixed Reality (MR) to robot simulation, it is possible to create a realistic, synthetic environment that enables robot developers to experiment freely and safely. The new simulation approach, named MR simulation, offers the flexibility of creating simulation environments composed of real and virtual objects that seamlessly interact with one another. This enables robot developers to mix virtual objects into the real experimental setup for cost and safety reasons.

A generic MR framework is proposed that provides the basis for creating MR simulations. The most novel element of the framework is a behaviour based interaction scheme that formalises interactions between real and virtual objects. To validate the framework, a general purpose MR robot simulator is implemented, and it is accompanied by MR interfaces designed to provide users intuitive views of MR simulations. Notably, markerless Augmented Reality (AR) techniques are integrated for real time visualisation.

Through three case study evaluations, the use of MR simulations has been demonstrated to produce results that reliably represent the real world. The thesis provides an insight to how MR simulations can help to minimise resource requirements in an industrial application, and enable efficient testing of a prototype aerial robot system. Findings from an initial user study indicate positive acceptance to the technology, while an observational user study identifies a strong contribution of MR simulation to the later stages of robot development.

Acknowledgements

First of all, I would like to thank my supervisors Assoc. Prof. Bruce MacDonald and Dr. Burkhard Wünsche for their guidance and support throughout my PhD. Thanks for the weekly meetings and the feedback on the papers, presentations, and the thesis. With their help, I have learnt a great deal about research throughout the years, slowly developing my communication and writing skills which were essential to help promote and defend my work among other experts in the research community. These skills have also helped me to network in conferences and to engage in collaborations with other research groups. I would also like to thank them for assisting me in applying scholarships, awards, and other research fellowships. They were all essential financial support to help me get through my PhD. Also, without the travel funds, it would never have been possible for me to visit so many different places in the world and learn about their research, culture, and environment (and also great food). The research visits and conference travels have broadened my vision and enabled me to look at my research from wider perspectives. I thank them for providing me such an enjoyable research environment to work in. I really enjoyed working and exchanging ideas with groups of talented people in the labs, and also having these valuable opportunities to play with (and break) different robots.

I would like to thank the people in the Robotics Group and the Computer Graphics group. They have shared with me many ideas that helped to shape this research.

I would also like to acknowledge the University of Auckland for providing the essential financial support over the past few years. I also thank the National Institute of Advanced Industrial Science and Technology for funding and hosting my three month visit in Tsukuba, Japan.

I would like to thank Doris for her support and encouragement which kept me going through stressful times. Finally I thank my parents for their patience, love, and continual support.

Contents

1

Introduction

Simulation is a method of experimentation by imitation. By imitating the behaviour of a real system through modelling with assumptions and approximations, a sample representation is created which allows us to study its operating characteristics. Simulation enables one to test the system with alternative conditions and observe the effect of those changes in a safe environment. The benefit includes the ability for users to explore possibilities, identify constraints, and diagnose problems of the system under development. Virtual simulations in particular allow users to speed up, slow down, or pause phenomena so that they can thoroughly analyse the series of events that occurred. Today, simulations are widely used for engineering design, research, education and training, and entertainment.

Robots are complex dynamic systems that benefit from simulation. Nevertheless, it is important to note that the development of robots cannot be treated in the same way as the development of computer software systems. Robots are mobile and autonomous, and perform tasks that require them to operate in the real dynamic world interacting with other physical entities. Conventional offline simulation alone is insufficient for creating robust robot systems, and it is only one of many ways of experimenting with robot systems.

The remainder of this chapter gives an overview of the areas that this thesis examines for improving the robot development process. Mixed Reality (MR) is then introduced; a technology that will be applied to enhance simulations and visualisations in robot development.

1.1 From Simulation to the Real World

Mobile robots are increasingly entering the real and complex world of humans in ways that necessitate a high degree of interaction and cooperation between human and robot. The development cycle of a robot system requires experiments to be conducted under many different environments and validated using a multitude of scenarios and inputs in order to gather reliable results before deploying the robot in actual operation. The transition from simulation to the real world can be a long and costly process, especially for the development of complex and expensive robot systems.

Simulation is an important technology for debugging and prototyping in the early stages of robot development and offers robot developers valuable insights to problems that are likely to occur in the real world. Computer simulation provides robot developers a safe and virtual environment for experimentation without depending on the physical robot. A long desired goal of virtual robot simulation tools is to facilitate the development of robotic software that can be ported to the real robot with minimal or no modifications. However, the real world contains noise and variations that cannot be fully replicated in simulation with existing technology. As virtual simulation tools grow in fidelity, the demand for computational resources increases. It is a challenge for a standard desktop computer to incorporate all sources of variations from the real world for realistic modelling of robot sensory input and motion characteristics, which may require simulation of lighting, noise, fluid and thermal dynamics, as well as physics of soil, sand, and grass as encountered in nature. The reliability of the simulation outcome is affected by the quality of the model representing the real system, which often relies on simplifications and assumptions of the real world. Results from simulation are therefore not a definite proof, but an indication of the possible outcomes in the real experiment.

To validate simulation results, experiments are carried out in the real world. Experiments are often conducted in controlled environments, such as a laboratory, in order to minimise any undesired effects. For example, the assumption of consistent illumination in an experiment to evaluate vision tracking algorithms requires controlled lighting conditions. It is also useful to create physical testbeds to mimic the target environment where the robot will eventually be deployed, but the process can be time-consuming and costly. Many robot applications require field tests to be undertaken in natural environments where robot developers have minimal control of the surroundings. In particular, field tests of high risk robot applications, such as aerial or underwater robot tasks, may require substantial human resources, equipment and technical support to ensure safety.

It can be seen that there still remain a number of key issues, mainly concerning cost and safety, in the robot development process. There are high risks and uncertainties involved in bringing results from simulation to reality, and the problem has not yet been

effectively solved by existing approaches. This thesis proposes a new development method, namely MR simulation, that offers a cost-effective solution to the experimentation of robot systems while facilitating smooth and reliable transfer of simulation results to reality.

1.1.1 Visualisation in Robot Development

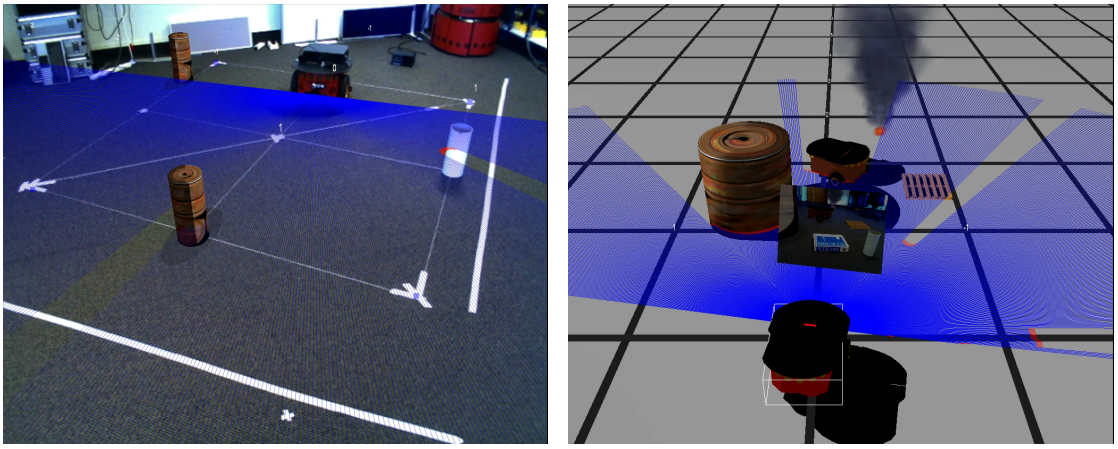
A robot is typically installed with multiple heterogeneous sensors and actuators, each produces large volumes of data. Visualisation helps to transform these raw data into graphical representations that can be easily understood by the human. For example, the shape of the robot's laser scan can be visualised over a 2D plane, which describes the range readings clearer and faster than printing each individual range value of the laser rays. Visualisation has a number of advantages in robot development. Specifically, visualisation can shorten the time needed for interpreting data, isolate problems of the underlying robot platform and software system during debugging, improve understanding of robot's view of the world, and aid learning the behaviour of unfamiliar robot sensors and actuators.

This research designs visual interfaces to facilitate understanding of the correlation between the robot data and the physical environment, and to help monitor the safety of the experiment. During robot development, there are limitations to real world views available to humans, who cannot sense, reason, or act like mobile robots. Additional textual, graphical and virtual displays are commonly used to help humans understand robots. However, the human may find it difficult to relate the additional visual information to the real world, e.g. displaying information in a separate window requires the human to mentally correlate its geometric association with the physical environment, which increases the cognitive load on the user. The use of MR is examined to help provide users a more intuitive view of the robot data and the experimentation environment, while improving the users' awareness of the components in the experiment (see Figure 1.1 for examples).

1.2 Mixed Reality for Robotics

MR merges the real world and the virtual world to create a single coherent environment. In other words, MR spans from the physical world to the digital world, forming synthetic environments or visualisations involving a mixture of real and virtual elements. MR encompasses Augmented Reality (AR) and Augmented Virtuality (AV), both of which are commonly used as visualisation techniques to help enhance human perception in completing a task.

AR is a technology that overlays virtual objects onto the real world. A common way to create AR systems is to render computer generated graphics over live video imagery in



(a) (b)

Figure 1.1: (a) An AR visualisation example that shows the robot's laser data graphically rendered over a view of the real world, helping to illustrate what the robot is sensing. (b) An AV visualisation example that displays the real world experiment from a virtual perspective, providing an extended view of the physical environment that can not be seen by the robot's onboard camera.

real time, and the resulting synthetic images are displayed to users using display devices such as computer monitors or Head Mounted Displays (HMD). AR is commonly used to enhance the view of the user by overlaying additional information about the surrounding real world environment. For example, a virtual arrow is drawn on top of the real world image to provide directional information.

In contrast, AV augments a virtual environment with real world information. Several computer graphics and simulation systems are created based on a similar concept. For example, AV may refer to a virtual avatar replicating the motion of a human in the real world. In this case, the virtual world is augmented by inputs from the real world. More recently, AV systems are created by streaming live video imagery inside a virtual environment, a technique that is now increasingly being applied to the areas of multimedia and entertainment.

There are various applications of MR in the field of robotics, primarily used as visualisation aids that help humans understand robots. This thesis builds on the work of MR visualisation for robot development. Example applications of AR and AV visualisations can be seen in Figure 1.1. AR can help to understand and debug robot data by presenting visualisations in geometric registration with the real world, while AV can help to increase awareness of components by providing an extended view of the experimentation environment.

Before proceeding further, it is important to point out that there exist different interpretations of the term MR, and it is commonly found that the term has been used interchangeably with AR and AV in the literature. To provide the readers a clearer un-

derstanding of how the term MR is used this thesis, a definition is provided as follows: "Mixed reality is a single coherent dimension where real and virtual objects co-exist and seamlessly interact in real time". MR is treated more as a concept than a technology. AR and AV are methods for creating MR.

This thesis proposes that MR is more than just a visualisation technique. The concept of merging of the real and virtual can be applied to the development of robot systems to create a new development environment that bridges the gap between virtual simulation and real world tests.

1.3 Application Study

The field of robotics is large and diverse which makes it difficult to derive a solution suited to all different robots and applications. Instead, a more practical approach is to focus the study on a specific area and generalise the requirements from the study in order to design a solution that is also applicable to other areas. This thesis narrows the study to the area of aerial robotics, which the University of Auckland Robotics Group has a growing interest in. In particular, its application in the field of agriculture is of great interest due to the importance of the industry in New Zealand.

The examination into the area of aerial robotics primarily serves as a case study for the design of the new development environment. Later in the thesis, it is shown that the requirements derived from the study are used to develop a framework that is not intended for only this specific area, but for various robot applications. In addition, an inherent advantage for choosing aerial robotics is that this area requires a more sophisticated development process with far stricter requirements compared to many ground robot applications because of the more serious consequences of failure. As the result, the analysis of the requirements is used to design a development environment that is capable of supporting the development of both small low-risk robot projects as well as expensive robot systems.

1.4 Scope of Research

This thesis analyses the requirements for creating a safe and realistic environment for the development and experimentation of robot systems. A review of the area of aerial robots in agriculture will provide the basis for deriving the requirements. It is also important that the requirements can be generalised in order to create a development environment suitable for a wide range of robot platforms, environments, and tasks.

This thesis then explores the application of MR for enhancing robot simulations. It analyses the common interactions between the robot, the environment, and the human

required in robot tasks. A simulation framework can then be built for creating MR simulations that capture these interactions. An implementation of the framework is also necessary to demonstrate the feasibility of the new approach.

A high fidelity robot simulator is of little use if the users could not extract meaningful information from the simulation. This thesis reviews the application of AR and AV visualisations in robotics and designs visual interfaces based on proven existing approaches to help users of MR simulation to better understand the underlying robot system being developed and its interaction with the surrounding environment.

This thesis includes quantitative and qualitative evaluations of the implemented MR simulation system. The accuracy of MR simulation will be quantitatively compared with results from existing development methods. User studies will be carried out to collect quantitative performance data of participants in using MR simulations. Qualitative measures will also be taken in the user studies to identify user acceptance to MR simulation and to evaluate the designed visual interfaces.

1.5 Contributions

The contributions of this thesis are split into main and secondary contributions.

The main contributions are:

- the application of MR to robot simulation, proposing a novel generic framework for constructing MR environments and facilitating interaction between real and virtual objects.
- the implementation of an MR robot simulator based on the framework. The simulator is open source and has been designed to be extensible and capable of integrating with different robotic software frameworks. This makes it useful immediately to robot developers from different communities and aids future research in MR for robotics.
- a thorough evaluation of MR simulations in different application areas for different robot tasks, showing improvements and limitations in comparison to virtual robot simulation and real world experiments.

The secondary contributions are:

- an analysis of the robot development process and the development environment, with findings suggesting a number of key areas for improvements
- an analysis of requirements for applying AR in robotics, resulting in two AR tracking configurations proposed. The strengths and weaknesses of each approach are also discussed.

- a set of guidelines for helping users to design MR simulations that suit their needs.

1.6 Publications

The work described in this thesis resulted in a number of relevant publications by the author, as listed below.

- I. Y.-H. Chen, B. MacDonald, and B. Wünsche, "Markerless augmented reality for robotic helicopter applications", in *Proceedings of the 2nd Workshop on Robot Vision*, Auckland, New Zealand, February 18–20 2008.

- I. Y.-H. Chen, B. MacDonald, and B. Wünsche, "Markerless augmented reality for robots in unprepared environments", in *Proceedings of the Australasian Conference on Robotics and Automation*, Canberra, Australia, December 3–5 2008.

- I. Y.-H. Chen, B. MacDonald, and B. Wünsche, "Mixed reality simulation for mobile robots", in *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 12-17 2009, pp. 232–237.

- I. Chen, B. MacDonald, B. Wünsche, G. Biggs, and T. Kotoku, "A simulation environment for OpenRTM-aist", in *Proceedings of the IEEE International Symposium on System Integration*, Tokyo, Japan, November 29 2009, pp. 113–117.

- I. Y.-H. Chen, B. A. MacDonald, and B. C.Wünsche, "Designing a mixed reality framework for enriching interactions in robot simulation", in *Proceedings of the International Conference on Computer Graphics Theory and Applications*, Angers, France, May 17–21 2010, pp. 331–338.

- I. Y.-H. Chen, B. MacDonald, B. Wünsche, G. Biggs, and T. Kotoku, "Analysing mixed reality simulation for industrial applications: A case study in the development of a robotic screw remover system", in *Proceedings of the Second International Conference on Simulation, Modeling and Programming for Autonomous Robots*, Darmstadt, Germany, November 15–18 2010, pp. 350–361.

1.7 Thesis Layout

The thesis is organised as follows:

Chapter 2 presents a literature review of related work.

Chapter 3 gives an analysis of requirements for designing an improved robot development environment and describes how MR can help.

Chapter 4 presents a generic framework for creating MR systems and describes its application to robot simulation.

Chapter 5 details the implementation of the MR robot simulator based on the framework and reveals the design of its visual interfaces.

Chapter 6 then describes the main visualisation technology, AR, that has been implemented and used to create the visual interfaces of the MR robot simulator.

Chapter 7 presents an evaluation of the MR robot simulator using a series of case studies.

Chapter 8 draws conclusions from the research and discusses potential future work.

2

Literature Review

This chapter presents a literature review that examines existing work in a number of related areas. It helps to provide an understanding of existing work, and reveal problems that remain unresolved. Figure 2.1 illustrates the areas that are related to this research.

Dark blue boxes represent areas that are closely related and will be examined in detail. Light blue boxes are areas that are considered less relevant but understanding of the topic is still necessary, and so the coverage is given at an overview level. White boxes are areas that are also important in the robot development process, but have been considered out of the scope of this research.

It can be seen from Figure 2.1 that an area of robot development examined is simulation. This is the primary area that this thesis focuses on for improvements. The review further divides simulation to virtual simulation and hybrid simulation, both are valuable methods for the implementation and testing of robot systems, and they have been used to save cost and time during robot development. Existing work on robot simulation is presented in Section 2.1.

There are two parts to visualisation that this literature review considers. The first part is an overview on visualisation design concepts which are important for creating effective presentations of data produced from robot experiments. The second part focuses specifically on AR visualisation (a subset of MR in the diagram). The real time visualisation technique has the potential to improve the human's perception of visual information in context with the real world, and this is particularly important for helping robot de-

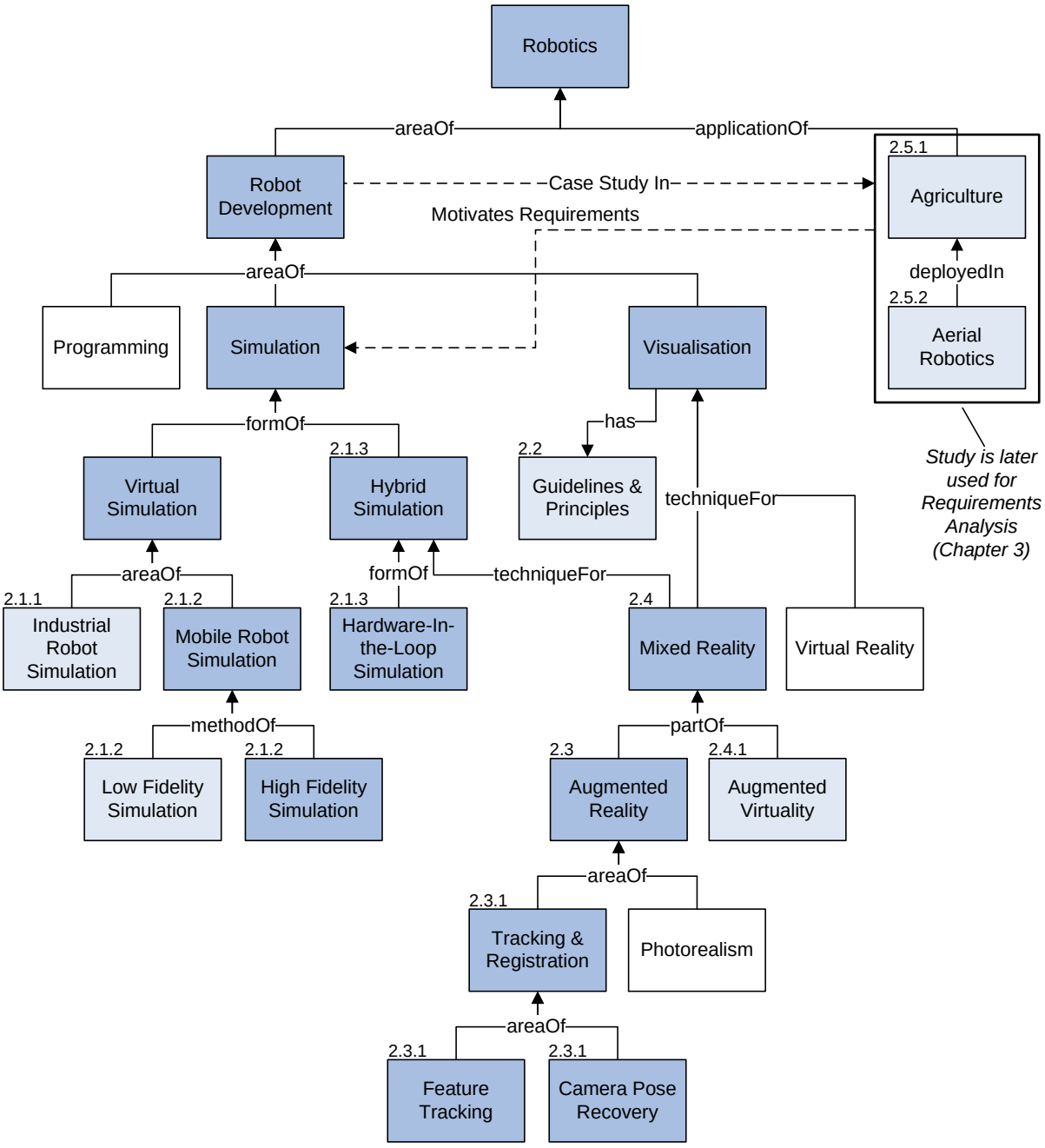


Figure 2.1: Dark and light blue boxes represent areas covered in this literature review. The number above each box represents the corresponding section in this chapter which the review is covered in.

velopers understand robot data in real world experiments. While the MR spectrum (see Section 2.4) covers Virtual Reality (VR), the field of VR visualisation is broad and diverse with substantial ongoing research, so the literature review will not include a section on VR visualisation. However more information about VR visualisation for robotics and descriptions of sample systems can be found in papers [57, 111, 188]. It has been decided to narrow down the scope to focus on a review of well-established concepts, which will be

covered in the first part of the visualisation review.

Section 2.2 and Section 2.3 present the first and second part respectively.

The application of MR in robotics has recently received increasing attention. This chapter reviews the literature of MR to understand its concepts, and explore how it contributes to the various areas of robotics. Section 2.4 provides a definition of MR, and describes the other constituting technology, AV. Existing applications of MR in robotics will be described in Section 2.4.3.

Finally, it is also important to understand how the robot development methods and tools are used in practice. Section 2.5 looks into the field of agriculture and identifies the various tasks performed by robots, with a particular focus on Unmanned Aerial Vehicles (UAV). Existing work on the development methods for UAV systems are examined in detail, which will later be used as a case study to derive requirements for designing an improved robot development environment in Chapter 3.

2.1 Robot Simulation

During robot development, developers need to create, debug, evaluate, and understand robots, from the working of the system as a whole to the execution of a component such as a sensor or robot arm, then down to individual code segments. A range of robot simulators have been designed for these purposes. This section presents existing work on virtual simulations and hybrid simulations in robotics. It first reviews features of industrial virtual robot simulators and a number of widely used virtual simulators for mobile robots. Hybrid simulation is then introduced, including applications where existing hybrid simulators have commonly been used in.

2.1.1 Industrial Robot Simulation

Robots were initially deployed for automation in the manufacturing industry. These robots operate in a constrained environment and repetitively perform tasks typically composed of a fixed sequence of pre-programmed actions. Industrial robots are generally in the form of manipulators with limited or no locomotion capability. Applications include spray painting [33, 70], welding [146, 25], and vehicle assembly [279, 202]. Virtual environments designed for the development of industrial robots are primarily tailored to simulation of these tasks. Simulations ensure the production of high quality products, and minimise production downtimes.

Several commercial software packages are available for simulation of industrial robot operations in manufacturing environments. These industrial robot simulation tools commonly include a built-in Integrated Development Environment (IDE) with features to sup-

port design and evaluation of robot work cells, trajectory planning, motion optimisation, task allocation, and offline programming and debugging. Some simulators are released by the robot vendors and are designed to be robot specific, e.g. ABB Robot Studio [1]. There are also many simulation tools designed to support a wider range of industrial robots, such as COSIMIR Robotics [110], Workspace 5 [198], and RobotWorks [222]. In comparison to more general purpose robot simulation tools (see Section 2.1.2), a common requirement in industrial robot simulators is the need for support of simulations of robot programs created in their native languages.

2.1.2 Mobile Robot Simulation

Outside the manufacturing industry, robots are expected to operate in a more complex and unpredictable environment, interacting with humans, other robots, and the surrounding environment. These robots come in various shapes and forms, including wheeled robots, legged robots, robot manipulators, aerial robots, underwater robots, and crawling robots. As the result, a broader range of robot simulators have been developed.

Low Fidelity Virtual Simulation

Low fidelity simulation tools such as 2D robot simulators are computationally more efficient and thus enable a larger population of robots to be simulated [114]. The nature of the robots tasks being simulated is mainly limited to the navigation of mobile robots on a 2D plane. The robots and the environment are represented by simple geometric primitives such as circles, rectangles, and lines. Some 2D robot simulators offer simplified 2D physics, and detect collisions by simply checking for intersections between geometric primitives, or by using ray tracing techniques.

Stage, developed as part of the Player project [20], is an example of a popular open source 2D robot simulator widely used in the research community worldwide. It provides simulation of mobile robots in a 2D bitmapped indoor environment, and recent development has enabled simulations to run in a 2.5D world. Player client programs, such as control or sensor algorithms, are able to exchange messages with virtual devices in Stage the same way as they would with real devices.

Other 2D robot simulators offering similar functionalities include Carmen's simulator [6] and the Rossum's Playhouse Mobile Robot Simulator [11]. However, compared to Stage they offer fewer sensor models in simulation. The development of these robot simulators have also become less active (see Appendix B for a list of robot simulators evaluated in this research).

High Fidelity Virtual Simulation

3D robot simulators are suitable for high fidelity simulation of smaller populations of robots. In general, they are computationally more expensive than 2D simulators but offer a higher accuracy and enable simulations of a wider range of robots and robot tasks. 3D robot simulators often include dynamics modelling for simulating physical interactions such as collisions and deformations.

Many high fidelity robot simulators are developed by integrating existing graphics rendering engines, physics engines, or game engines. These readily available technologies facilitate rapid creation of simulation systems. They allow robot developers to concentrate on building robot and sensor models without worrying about low level graphics programming or heavy mathematical implementations. A number of widely used 3D robot simulators are described in the subsequent paragraphs.

The Microsoft Robotics Studio simulator [177] is a 3D robot simulator that offers high fidelity simulation of general robot tasks. The simulator is built on a service-oriented architecture, and each entity in the simulation provides a number of operations in the form of a service for exchanging messages with other components. Both the simulator and the underlying Ageia PhysX [18] physics engine are closed source.

USARSim [61] is a 3D mobile robot simulator, initially designed for the simulation of urban search and rescue operations, and is now commonly used as a research simulation platform for a wide range of robot tasks, including robot soccer in the RoboCup [261] community. The simulator is built on the commercial game engine, Unreal Engine 2 [103]. Robot programs exchange messages with simulation entities in the engine through the provided Gamebot interface.

OpenRAVE [92] is an open source planning and simulation tool. The plug-in driven architecture supports the integration of custom functionalities, such as planning, control, or sensing modules, which are loaded at run time. The simulator integrates the open source Open Dynamics Engine (ODE) [245] and Bullet Physics Library [4] for dynamics simulation and collision checking. A notable utility included the simulator is the analytical inverse kinematics solver, ikfast, that can automatically generate 6D/3D inverse kinematics solutions given the position of the end-effector of a robot link. Robot programs interact with the simulation through high level scripts in scripting environments such as Octave or Matlab. The simulator is designed to be cross-platform, and many of the components are reusable. The focus of the simulator has been placed on, but is not limited to, robot manipulation and humanoid robot tasks.

The OpenHRP3 project [9] has a dynamics simulator that simulates humanoid robotics and is used for the development of humanoid software applications. The simulator uses Virtual Reality Modeling Language (VRML) for modelling, and features algorithms for dynamics simulation, contact force calculation, collision detection, and walking pattern

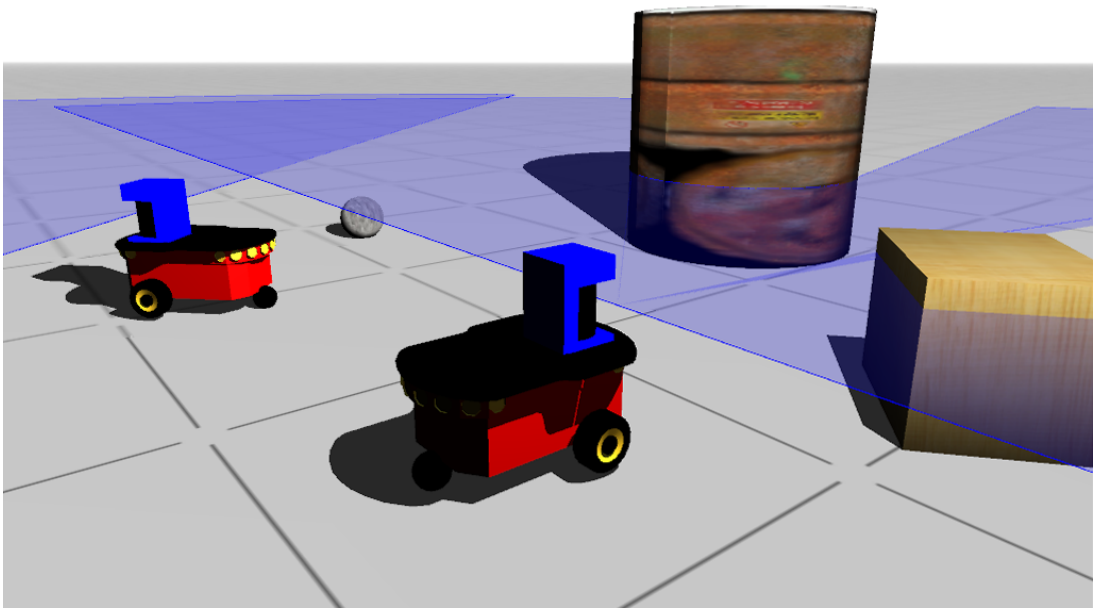


Figure 2.2: Gazebo - a 3D multi-robot simulator from Player/Stage [20].

generation. The simulator is tightly integrated with the OpenHRP framework.

Gazebo [20] is an open source 3D multi robot simulator from the Player Project, see Fig 2.2. It integrates open source libraries to provide high quality graphics rendering and rigid body physics. Robot models can be created from various sensors and actuators available in the distribution, and the simulator also supports loading of plug-in models developed by users. Gazebo is a standalone program, and designed to be compatible with Player. Interfacing with other robotic software frameworks is possible.

High fidelity commercial robot simulators are also available, such as Webots [84] and the anyKode Marilou Robotics Studio [30], both of which are widely used in the research community.

Webots [84] is a 3D robot simulation tool from Cyberbotics Ltd. The tool itself is an IDE that supports programming of robots and visualisation of the simulation in the same environment. The IDE has a built-in editor to let users create 3D worlds and robot models, and design motion sequences for articulated robots. The simulator uses VRML for 3D graphical modelling and ODE for physics simulation.

The anyKode Marilou Robotics Studio [30] is a 3D robot simulation tool that also provides an IDE with an editor for graphical modelling of robots and the environment, and a simulation engine that supports rigid body physics simulation. The simulation tool includes the open source Marilou Open Devices Access (MODA) software development kit to support creating centralised and decentralised applications involving multiple robots and worlds over the network.

Evaluation of Robot Simulators

While there are different virtual robot simulators available, there tend to be very few evaluation efforts. USARSim [61] is a robot simulator that presents a more formal quantitative assessment of its simulation accuracy. Comparative evaluations validated USARSim's simulated vision and wireless communication systems with real world setups [62]. Each validation involved a series of tests performed on simulated and real world components under different conditions. For example, edge tracking, template matching, contour extraction, and optical character recognition algorithms were applied to simulated images and real images under different lighting conditions, and the discrepancies between the results were identified. Moreover, efforts have also been put into validating the simulator's use for Human-Robot Interaction (HRI) tasks, showing results from user studies that involved participants teleoperating a robot in simulated and real world environments [273]. However, the analysis performed on the results was minimal, revealing only a comparison between user task completion times. No formal statistical tests were used to indicate significance in the findings.

In the area of evolutionary robotics, Jakobi [138, 137] argues that the stochastic nature of the evolved robot system makes objective evaluation of robot simulation difficult since it is unclear what the final (or optimal) robot behaviour would be like. The work investigated the influence of noise in robot simulation, and the evaluation involved a panel of judges subjectively giving scores on the correspondence between the real robot behaviour and those produced from simulations under different noise levels. Two experiments, obstacle avoidance and light seeking, were conducted but no metrics were used to define what constitutes a high simulated-to-real behaviour correspondence.

The review so far has seen that statistical, objective evaluation methods are not extensively used in robotics for evaluating robot simulation tools. However, this is in contrast to studies in the general field of simulation modelling which recommend collecting quantitative, objective data for validating simulation models using methods such as regression analysis. Kleijnen [147] presents a survey of methods for validation of simulation models, from data collection to the use of statistical tests, such as t-tests, for comparing between simulated and real world responses and testing whether they are positively correlated. Sargent [229] also presents a general review of methods for validating the operational behaviour of simulation models. Similar to Kleijnen, the primary objective validation method involves the use of hypothesis tests and confidence intervals for comparing between the means of simulated and real world outputs.

2.1.3 Hybrid Simulation

Conducting tests in the real physical world can be costly and risky given the uncertainty of deploying the developed robot system directly from virtual simulation. Hybrid simulation combines simulated and physical components in simulation to offer developers a cost-effective alternative to real world tests. In principle, hybrid simulation is based on the Hardware-In-the-Loop (HIL) simulation paradigm.

HIL simulation is a form of real time/online simulation. HIL is often used for testing of real time embedded systems when they cannot be safely and thoroughly tested in their operating environments. An HIL simulation includes the actual hardware components in the testing process and requires other parts of the system to be simulated. Typical tests involve feeding synthetically generated sensor data and actuator commands as inputs to the embedded system under test. HIL simulation is often used for developments with tight schedules, providing an efficient solution to thorough testing and integration of complex and expensive systems. HIL simulation has been commonly applied in the automotive industry and the development military defense systems, and now it has found its way to robotics.

HIL simulation is commonly applied to simulation of high risk robot applications. An example is in the area of underwater robotics. Kuroda *et al.* [158] describe an early example hybrid simulation that simulates Autonomous Underwater Vehicle (AUV) operations in a synthetic world. HIL simulation is applied to test a simple obstacle avoidance scenario involving a real robot navigating in a pool filled with virtual cylindrical obstacles. Built on the same idea is the AUV simulator, Neptune [220]. The work explicitly provides a classification of real time simulators, proposing a difference between conventional HIL simulation and the more advanced hybrid simulation. The classification considers that conventional HIL simulation feeds low level simulated inputs to the real hardware and similarly the outputs are generated in the simulated world, whereas hybrid simulation focuses on creating simulations where both simulated and real robots/sensors can be used. This classification has also been extended by other HIL simulation work in the area of underwater robotics, such as in [86]. Interestingly, these HIL simulation tools are inspired by the idea of AR.

HIL simulation is particularly important for the experimentation of UAV systems as it provides a valuable solution to the coupled testing of software and hardware components on ground. Johnson and Mishra [139] describe a number of configurations for real time testing of UAV systems. A simple HIL simulation tests the onboard flight avionics system alone without the physical UAV platform. GPS, inertial, or radar data are passed to the onboard computer that executes high level software algorithms. Recorded videos from previous test flights can also be used to test onboard image processing algorithms. The resulting actuator/control outputs from the algorithms are then forwarded to a virtual

UAV simulator that visualises the flight motion on a computer screen. To test the onboard algorithms using data from the actual onboard sensors, one solution is to place the UAV on the back of a truck that is driven outdoors. This allows the system to be tested with dynamic data in real time. Nevertheless, these approaches do not involve testing the actual UAV in flight.

The area of space robotics also benefits from HIL simulation because testing in the robot's operating environment, i.e. space, is too costly. An example of an HIL simulator for space robot operations is developed by the Canadian Space Agency [89, 22]. The HIL simulator emulates a space robot manipulator's interaction with the environment by replicating the dynamic behaviour of the simulated space robot model on a ground robot manipulator. To study collisions with objects in space, Takahashi *et al.* [256] develop a hybrid simulator that incorporates a simulated model for micro-gravity and uses a nine Degree-Of-Freedom (DOF) motion platform for demonstrating the resulting motions from collisions between a robot manipulator and floating objects in space. The use of a hybrid simulator in this case avoids the cost of building a full hardware simulation environment such as a water tank.

2.1.4 Summary

The literature reveals that there has been extensive work in creating virtual robot simulation tools that aid robot development. There is a trend indicating simulation tools for research are becoming more general and applicable to different robot tasks, platforms, and environments. However, there appears to be little support for these simulation tools to provide real time simulations and to interoperate with real hardware components (see Section 2.6 for a discussion on this issue). On the other hand, hybrid simulation integrates physical components into simulation for experimentation of robot systems, although its use has mainly been targeted at the development of expensive robot systems in high risk operations. A general purpose hybrid simulation tool for robot development is not yet available.

2.2 Visualisation Concepts

The previous section focused more on the technologies for the robot simulation, introducing various mobile robot simulators as tools for development. This section will study the human's interaction with the robot during development, focusing on the use of visualisations as an interface for interaction.

Robots today are installed with a variety of sensors, and each individual sensor may produce a different type of data over time. These raw sensory data can also be processed

internally by robot programs into abstract data types such as an algorithm state. Understanding of these data is necessary for developers to debug and evaluate their systems in both simulations and real world tests. This section reviews guidelines and principles for transforming the large volumes of data typically produced by robots into effective visual representations that can be intuitively understood by the human.

2.2.1 Human Perception of Visual Information

Robot data is not always useful in its raw form, or visible from direct observation of the environment. So a common idea is to create a visualisation of the data that provides a human observer with an intuitive, albeit non-real image of the data.

Although it may seem desirable to visualise as much as possible and maximise the amount of information that can be delivered to the users at once, doing so has a number of negative effects. It results in clutter and conceals important information. Moreover, overloading the human visual system can increase the effort required to interpret the data presented. It is crucial to understand the human capabilities and limitations, and design systems that maximise the human and robot performance.

After all, robot simulators are interactive tools designed to assist human users. Visualisation studies in the field of Human Computer Interaction (HCI) begin by understanding the capabilities of the human perceptual system. Human visual perception is concerned with the physical reception of the stimulus, and processing and interpretation of that stimulus. The way humans perceive size, depth, brightness, and colour is important to the design of effective visual interfaces [94].

Perception of size and depth are highly dependent on existing cues in the environment. As objects appear in the scene, the space each object occupies within the field of view of the human eye provides cues for its size. Visual cues such as partial occlusions and shadows help to determine the relative positions and distances of objects [204]. Furthermore, natural scenes are rich in textures, and these texture cues help humans to perceive the structure and the relative size of objects in the scene. Familiar objects in the scene, such as trees and buildings, also strengthen spatial perception because humans already have a general expectation of their sizes [231]. Perception of depth can also be enhanced by monocular cues such motion parallax and perspective transformations of the visual image, and binocular cues such as the disparity between stereo views [66, 223].

Colour is one of the most important factors that influences human visual perception. The human eye is able to distinguish between colours because the receptor cells are sensitive towards light of different wavelengths. The perception of colour plays an important role in discriminating among surfaces which helps humans to distinguish between objects. Brightness on the other hand is associated with luminance, which is the amount of light emitted by an object. In comparison to the use of colours, studies found that brightness

yields better results in helping humans to distinguish a target from its background, and scenes with limited brightness contrast lead to a lack of sensitivity towards motion and depth [221].

2.2.2 Visual Representation

Scientific Visualisation [123] and Information Visualisation [69] are two fields that study methods for transforming raw data into effective visual presentations. Scientific visualisation creates graphical representations of high dimensional, physical data generally collected from scientific experiments. Information visualisation reinforces understanding of data which do not have natural physical representations, such as text, hierarchies, and statistical data, and these data are generally non-numeric, non-spatial, and abstract [169, 218]. Both fields provide guidelines, principles, and proven existing visualisation techniques that work within the limitations of the human visual perceptive capabilities.

Collett [80] proposes that because robot applications are far too varied, there are no strict design rules that can be applied to robot visualisation. Moreover, the work argues that it is important to identify the common data types in order to promote the reuse of data visualisation techniques. Collett examined the Player framework [20] and produced a characterisation of robot data based on the Player data types, showing that robot data can be spatial, sequential, abstract, or stochastic. Therefore, it is advantageous to exploit techniques from different visualisation domains for visualising robot data.

For example, different visual representation methods have been used to aid robot manipulation tasks. Glyphs of varying sizes, deforming effects [159, 246], and colour codes [91] are used to indicate proximity and magnitude of contact force, while a virtual animated character [96] can be used to represent the different states of the robot throughout the execution of the task. Studies in information visualisation also encourage the use of real world based metaphors as they provide visual representations in forms that humans are familiar with, which help to reduce the cognitive load on the users. For example, real world navigational devices such as compasses [182] and radars [58] have commonly been used to aid navigation in unknown environments.

Other than finding suitable visual representations for data, additional visual aids of different colours, sizes, shapes, and forms are commonly introduced in the design of a visual interface in existing work to help reinforce human perception. These aids do not correspond to any existing data and serve as supplementary cues to guide human interaction. Navigation or teleoperation in large scale virtual environments are areas which benefit from the use of additional visual aids. Visual aids such as directional arrows, virtual waypoints and artificial landmarks are commonly used to convey spatial information in unfamiliar environments [264, 268, 134, 182].

2.2.3 Summary

The review in visualisation concepts shows that the important design principle is to create visualisations that facilitate communication of information by working within human visual perceptive capabilities. Concepts from the two main visualisation areas, scientific visualisation and information visualisation, are both applicable to robotics. For visualisation in robot development (and also robotics in general), a common approach is to first understand the type of data that need to be visualised then apply proven existing techniques. Section 2.4.3 will show how the visualisation concepts reviewed above can be used in AR for real time visualisation of robot and task relevant data in robotics. But first, a review of AR is given in the next section.

2.3 Augmented Reality Tracking

Over the past decades, VR has become more and more widely known and it has made its way outside the research laboratories into games, movies, museums, theme parks, and many more application domains. VR is typically associated with virtual or synthetic worlds composed entirely of computer generated graphics, sometimes referred to as virtual environments. In contrast, AR refers to a real world augmented with virtual objects, or real world images enhanced by the addition of computer generated graphics.

AR can be found in many application domains, such as entertainment [262], archaeology [269], medicine [225], military training [163], and more [37, 35]. A common use of AR is to reveal information that is hidden to users. For example, a building can be superimposed with 3D models illustrating its infrastructure. Similarly, surgeons can be guided with X-ray vision that provides an internal view of the patient during surgery training. In many cases, AR is used as visual cues to assist users in accomplishing a task. For example, annotations of instructions can be overlaid onto the view of technicians during repair and maintenance tasks. AR has also been applied to various areas of robotics as will be shown later in Section 2.4.3

This section reviews in detail methods for creating effective AR visualisation. It examines existing tracking and registration techniques necessary for the development of AR systems, as well as methods for improving AR realism.

2.3.1 Tracking and Registration

One of the main challenges of today's AR applications lies in accurate registration of virtual information into the real world. As the camera moves around in the environment, the virtual objects introduced into the scene need to be accurately aligned with the real world to give the users the illusion that these virtual objects are part of the physical

environment. A small error in the registration can lead to ineffective AR results because small misalignments are easily perceptible to the human eye.

AR registration requires the camera's external pose parameters to be tracked in order to overlay virtual objects onto the view of the camera. Visual tracking methods track the camera movement by analysing features detected in the input video stream. Given the correspondence between the 2D information acquired from the camera images and the 3D information of the physical environment is known, the camera pose can be extracted.

Existing AR systems track a) fiducials or b) natural features in the scene to recover the camera pose.

Fiducial Tracking

Fiducial based AR systems operate by placing artificial markers or landmarks of known geometric or colour properties in the scene. These fiducials can be easily recognised and extracted from camera images.

Early AR systems typically employed active fiducial tracking techniques. Azuma and Bishop [36] mount optoelectronic sensors on HMDs that track infrared Light Emitting Diode (LED) beacons mounted in ceiling panels. Tracking is made easy because the use of optical filters eliminates other sources of illumination, and the only features visible are these LED fiducials. With known geometric arrangement of the LED beacons, the position and the orientation of the user's head can be calculated. A similar approach is taken by Bajura and Neumann [39] who attach red LEDs on objects of interest. The LEDs emit bright illuminations that can be detected by applying brightness thresholds to the captured image.

In comparison, passive fiducial tracking rely on completely passive vision sensors and fiducials. One of the early AR systems that use passive fiducial tracking is proposed by Hoff and Nguyen [130] who place concentric contrasting circles, arranged in a distinctive geometric pattern, on a physical object to enable estimation of the relative pose between the object and the camera. These circles are formed from a high contrast of black and white colours, and the locations of their centroids can be identified by performing simple thresholding and filtering operations on the image.

Perhaps, the most well-known passive fiducial tracking system of all is the ARToolKit library, based on the work by Kato and Billinghurst [143] (see Figure 2.3). It offers a set of planar square markers, each has a unique pattern with thick black borders. After thresholding the input camera image and finding regions in the image that can be bounded by four line segments, template matching is performed on these regions to detect the identity of the marker. Once the marker is found in the camera image, its vertices and direction information are used to compute the position and orientation of the camera, which enable virtual objects to be accurately placed over the marker. ARToolKit has

Figure 2.3: Sample ARToolKit markers from the open source library [143].

inspired other marker based AR libraries, including ARTag [106] and ARToolKitPlus [271]; the later is optimised to operate on mobile handheld devices.

While marker based tracking methods are a fast, low cost solution to many AR applications, they have a number of drawbacks. Partial occlusion of the marker or direct exposure to strong lighting conditions would cause tracking to produce erroneous registration results. Moreover, the camera is constrained to look at regions where the markers are in view, hence limiting the user's working space [38, 187].

Natural Feature Tracking

Natural feature tracking is a form of markerless tracking. Tracking natural features in the scene avoids environment modifications and has the potential to scale AR systems to operate in large, outdoor environments. However, natural feature tracking remains challenging due to noise and changing lighting conditions often encountered in uncontrolled environments.

A number of AR systems rely on point detection algorithms to extract features from the scene. Interest points commonly positioned in image regions with rich information content, e.g. local intensity maxima, are extracted using point detectors, such as the Harris corner detector [125], and tracked over subsequent input images using feature matching algorithms.

The Kanade-Lucas-Tomasi (KLT) tracker [263, 237] is an example of a point tracking algorithm that has been used to track natural features for recovering the camera pose in many outdoor AR applications [280, 141, 196, 45, 284]. In principle, the KLT feature selection and tracking processes are closely dependent on each other. The KLT tracker defines the quality of a feature by how well the feature can be tracked. The feature selection process is based on the Harris corner detector, and accepts a window of pixels as a good feature if it is above a certain noise threshold and well-conditioned. The KLT tracker then applies a dissimilarity measure to quantify the transformation between image frames based on two models of motion. A simpler pure translation model is used for approximating small inter-frame motion of the camera. Larger motions are modelled by affine motions which compute the deformation matrices in the transformations between

the image frames. The dissimilarity measure provides an indication of how well the feature is tracked and helps to filter out feature windows containing depth discontinuity.

To strengthen the reliability of feature tracking, region based feature descriptors are built around interest points to enable wide-baseline matching. Feature descriptors need to be highly distinctive in order to minimise the chance of incorrect matches. An example is the Scale-Invariant Feature Transform (SIFT) descriptor proposed by Lowe [166]. Lowe's implementation extracts Difference of Gaussians (DoG) keypoints from images and builds SIFT feature descriptors based on the surrounding image region. These descriptors are invariant to scale, translation, rotation and partially to illumination changes. Feature matching is performed by finding the nearest neighbour using the Best-Bin-First (BBF) search approximation algorithm [46]. The introduction of SIFT has received wide attention from many researchers, and variants of the SIFT algorithm have emerged, such as SURF [44], PCA-SIFT [144], and GLOH [178]. In comparison to other feature descriptors, SIFT and its variants are claimed to yield superior results in terms of feature distinctiveness and matching accuracy but lack real time performance [184, 178, 151]. SIFT has also been used in many AR applications [243, 274, 275]. The downfall of the use of feature descriptors normally lies in expensive computational requirements during feature extraction, which presents a barrier for these AR systems to perform at interactive frame rates.

To address performance degradations due to the high computational cost necessary for building and matching feature descriptors at runtime, Özuysal *et al.* propose a Semi-Naive Baysian classification framework, named Ferns [199], for planar region detection. The algorithm offsets the computation load of building the descriptor to an offline training phase in which a planar object classifier is trained. During the online detection phase, the planar object classifier determines the best match of each input interest point using simple binary image tests. The classification process is fast, however, a separate classifier is required for detecting each planar object, which can lead to large memory requirements. Calonder *et al.* [60] later propose to complement Ferns with compact features which describe interest points online then add them to the training set. These compact features are memory efficient and they improve the Ferns classifier to recognise a larger database of planar features.

Simultaneous Localisation And Mapping (SLAM)

A solution for recovering the camera pose is to use SLAM. Vision based SLAM or visual SLAM tracks the pose of a camera while building a map of the environment. Localisation of a camera in visual SLAM typically begins with an initial estimate of camera position, and uses a motion model to predict its movement. As the camera continues to move and make observations of the scene by analysing features in the camera images, the camera

pose estimates are iteratively refined in statistical filter frameworks such as the Extended Kalman Filter (EKF).

Pioneer work on visual SLAM has been done by Davison *et al.* [87], who proposed an EKF based Monocular SLAM (MonoSLAM) system that is capable of tracking the pose of a single camera and building a map in real time. The MonoSLAM system reduces the computational complexity of visual SLAM problem by using an active vision scheme and a constant velocity model of the camera. The active vision scheme uses a measurement model to estimate the projection of features in the image frame so that the amount of image processing required in feature matching can be reduced by searching for features only in these predicted regions. However, despite efforts in improving tracking performance, tracking failures are inevitable. Williams *et al.* [276] enhanced the MonoSLAM system with a relocalisation strategy to rapidly recover from failures. The algorithm trains optimised Ferns classifiers in a background thread for each feature in the map, and when the SLAM system becomes lost, these classifiers operate on the next input image to detect previously seen features and apply the three-point-pose algorithm [107] to relocalise the camera.

Pupilli and Calway [212] propose to use a particle filter for visual SLAM as it maintains multiple hypotheses of the camera pose and provides improved robustness towards erratic motions. The particle filter framework is later combined with an auxiliary Unscented Kalman Filter (UKF) for a more efficient mapping process [213]. Alternatively, Chekhlov *et al.* [68] improve the robustness of visual SLAM systems by using SIFT-like descriptors in their UKF SLAM system for more robust feature matching. AR visualisation is shown to resume after recovery from failures due to erratic motions of the camera and visual occlusions.

Structure from Motion (SfM)

SfM is an alternative approach to SLAM for localising a camera. In SfM of a single camera, the camera first takes multiple views of the same scene as it moves in the environment. Based on epipolar constraints, the image feature correspondences between these image frames provide information to extract the camera pose associated to each view. The camera pose estimates are then used to predict the 3D coordinates of the image features and reconstruct a sparse 3D representation of the scene. Like SLAM, estimates of the 3D positions of features and the relative motion of the camera need to be iteratively refined. Recent SfM based AR systems investigate ways to refine these estimates by applying bundle adjustment, a computationally demanding batch refinement technique that minimises the reprojection error between the observed and the predicted locations of the image features [126].

A number of SfM based AR systems explore the use of planes in the scene for recovering

the pose of the camera. Simon *et al.* [239] propose a markerless AR system based on the assumption that the operating environment contains one or more planar surfaces. The camera is tracked by calculating the frame-to-frame planar homography which is then used to recover the relative camera pose. Similarly, Lourakis *et al.* [165] recover the camera pose using homographies from image triplets, and the pose estimates are continuously refined using local bundle adjustment. Yuan *et al.* [284] embed in a reconstructed projective space four planar control points which are used to recover the camera pose based on pose estimation techniques commonly found in marker based AR systems.

Recently, Klein and Murray [148] proposed a Parallel Tracking and Mapping (PTAM) system that achieves accurate and robust AR results while maintaining a relatively high frame rate. The system separates tracking and mapping into two threads running in parallel to achieve real time performance. Thousands of FAST keypoints [226] are tracked in every image frame, and only a number of selected keyframes are used for map building. Bundle adjustment is applied both locally with a subset of most recent keyframes and globally with all keyframes. Klein and Murray also improve the tracking performance of the PTAM system for fast moving cameras using a combination of point and edge features [149]. The PTAM system has also been modified to work in larger environments by maintaining several independent sub-maps [63].

An analysis by Strasdat *et al.* [249] found that, with large processing budgets available, SfM techniques with bundle adjustments offer superior accuracy compared to filter based visual SLAM systems while maintaining real time performance.

Visual Servoing

In the robotics community, visual servoing is the control of robot motion based on feedback from vision sensors. Typically, the vision sensor is attached to the robot and guides the robot's movement based on a closed loop control system. The work of Espiau *et al.* [104] has served as the basis for many existing visual servoing systems.

A new form of visual servoing, known as virtual visual servoing, considers the case where the camera is attached to a virtual robot. Given a known initial camera pose, visual servoing approaches the visual tracking problem by determining the translational and rotational movements necessary to move the camera in order to minimise the forward projection error of the image features. Virtual visual servoing is first applied to AR by Sundareswaran and Behringer [254] who place circular concentric circles (fiducials) on a computer case in the scene to drive the motion of the camera while visualising the components inside the case. Behringer *et al.* [45] extend the visual servoing approach to use edges of a Computer Aided Design (CAD) model and scale the system to operate outdoors.

A virtual visual servoing framework for real time markerless AR is proposed by Marc-

hand and Chaumette [171]. The framework discusses the mathematical foundations of virtual visual servoing for tracking geometric features including points, circles, cylinders, and straight lines. This has later led to both model-based and model-free virtual visual servoing approaches for creating AR systems [81]. The framework integrates the M-estimator for refining camera pose estimates, which produces robust AR in case of noise and partial occlusions.

Hybrid Vision-Inertial and GPS Tracking

A hybrid approach involving two passive sensing technologies, inertial and vision sensors, is commonly used to track the head pose of the user wearing an HMD. Inertial sensors are computationally efficient but are susceptible to drift accumulation. Vision systems can improve tracking accuracy by measuring and minimising residual error but lack robustness to rapid motions and incur heavy computational cost. Each technology has its shortcomings and with a hybrid approach, one technology compensates the weakness of another.

A prediction-correction technique is proposed by You *et al.* [282] in their hybrid tracking system for AR registration. Angular motion estimates are predicted using inertial data from gyroscopes and a compass sensor, and the resulting accumulated drift error is corrected by vision at regular time intervals. The system is used to track the viewpoint of the user so that virtual objects can be inserted with the correct orientation. A similar approach is taken by Satoh *et al.* [230] who use vision to compensate accumulated drift error of an inertial sensor, but they simplify the feature matching process in order to create real time wearable AR systems. The major limitation of the above systems is that they are only capable of tracking the camera rotation and not the translational movements. Later approaches [281, 219, 217] recover the full six DOF camera pose by fusing the two sensor outputs in EKF frameworks

In outdoor settings, AR systems have been found to use position data from GPS devices for tracking and registration. An early example is Columbia's Mobile Augmented Reality System (MARS) [131] that uses a real-time-kinematic GPS to correct errors of pose estimates produced by inertial/magnetometer orientation sensors. The wearable Tinmith-Metro system [206] combines GPS with inertial and marker based tracking for creating outdoor AR applications. As opposed to using GPS for error correction, in an application called ARQuake [262] the system uses GPS position data for registration of distant objects when the confidence of ARToolKit [143] tracking results falls below a certain threshold.

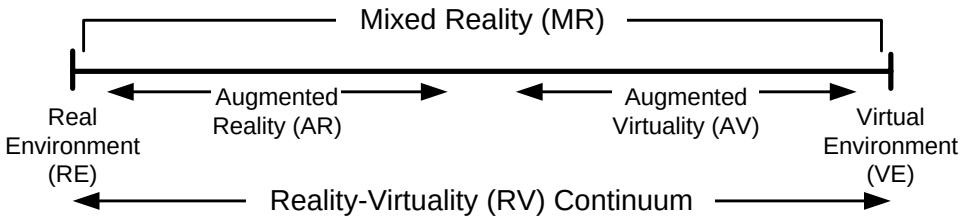


Figure 2.4: Reality-Virtuality (RV) Continuum (Adapted from [179]).

2.3.2 Summary

Recent work on AR focuses on improving the accuracy and robustness of markerless tracking, while reducing the overall computational and memory demands. The main shortcoming of point feature tracking is that points are easily lost due to sudden camera motions and occlusions. While highly distinctive feature descriptors can be built around interest points for increasing the robustness of the feature matching process during tracking, expensive computational requirements have an impact on system performance in terms of speed, thus reducing the frame rate and degrading AR quality.

Another focus of recent AR systems is on enabling tracking in large, unknown environments. The problem has recently been tackled using filter based visual SLAM systems, e.g. MonoSLAM [88], or SfM techniques with bundle adjustments, e.g. PTAM [148]. These methods enable the system to extend its working area by incrementally building a map as the camera moves towards previously unknown scenes. However, creating real time markerless AR in large dynamic environments still remains a challenge and is actively being addressed by researchers from the areas of computer vision and robotics.

Note that AR can be considered as a subset of MR as will be described in the next Section. In addition, Section 2.4.3 shows how AR visualisations are applied to various areas of robotics, demonstrating unique contributions.

2.4 Mixed Reality

MR is a relatively new term compared to AR. MR can be described by a Reality-Virtuality continuum proposed by Milgram *et al.* [180, 179], see Figure 2.4. The Real Environment and the Virtual Environment sit on the two opposite extrema of the continuum. MR is a term proposed to encompass all mixtures between the two extrema of the Reality-Virtuality continuum. Within the continuum, there lie AR and AV. As reviewed earlier, AR augments the real world with virtual information. In contrast, AV augments a virtual environment with real world information.

It is proposed that MR is a more encompassing term for applications where distinguishing between AR and AV is unnecessary or difficult. For example, a system may

appear to have a real world background augmented with virtual information (AR), but in fact the underlying scene is a virtual environment but predominantly covered by a live video overlay of the real world (AV). It can often be found that the primary background, real or virtual, of an environment varies over time. For example, a system can lead an observer to walk from a virtual dinning room filled with real world furniture through a door into a real world bedroom with virtual furniture. In this case it illustrates a traversal from AV to AR within the Reality-Virtuality continuum.

The definition of MR is still vague as there exist several interpretations. It must be noted that the term MR has been used interchangeably with AR and AV in the literature. Section 1.2 provided the definition of MR used in this thesis.

MR has been applied in many areas, primarily in entertainment, multimedia, and education and training. However, only a small number of MR systems have found their place outside the research laboratories. Successful technology transfers of MR systems have been demonstrated for educational purposes in museums, aquariums, and schools [247, 133].

Section 2.3 provided a review on AR and how it is used for visualisation. The remainder of this section gives an overview of AV, and presents existing software tools for developing MR systems. Finally, the application of MR in various areas of robotics will be examined.

2.4.1 Augmented Virtuality

AV is a new term proposed together with MR by Milgram *et al.* in their work on classifying MR displays [180]. Nevertheless, an analogous technology has always been in use in the area of computer graphics. Typical examples can be seen in computer games where virtual geometric models are mapped with photo textures taken from the real world to enhance realism. The simple case of texture mapping with real world images also falls into the classification of AV according to Milgram *et al.*'s taxonomy, however, this technique has been more commonly referred to as VR.

A similar technique for achieving AV is through the use of image-based rendering. Image-based rendering is used to create an AV world based on a database of real world images with no prior geometric measurements. The technique has been applied to reconstruct large scale synthetic environments from high quality camera images, and arbitrary viewpoints can be generated for users to observe and navigate in these environments [258].

A more advanced variation of AV can be seen in recent MR applications that superimpose real human actors from live video streams into a virtual environment [64, 189]. This is achieved by using techniques such as chroma keying to extract silhouettes from video images captured by cameras surrounding the human actor situated in a separate physical location, then inserting the silhouettes into the virtual environment in real time.

The resulting system appears as if the real human were interacting with virtual objects through speech and gestures. This idea resembles television weather forecasts where a weather reporter appears to be standing in front of a computer generated weather map, but in fact, it is only a blue background.

2.4.2 MR Development Tools

As there exist different interpretations of MR, development tools with different underlying technologies have been found in the literature for creating MR systems. In addition to tracking and registration functionalities common in these tools, it is found that the focus of recent tools have shifted towards content creation and promoting richer interactions with the end-user.

A number of generic AR frameworks originally designed for the development of AR systems have been used for creating MR applications. These frameworks include Studierstube [233], DWARF [43], and Tinmith-evo5 [207], which provide rich programming interfaces that allow developers from different application domains to create a wide range of AR/MR systems.

AMIRE (Authoring MIxed REality) [2], is an open source software that focuses on providing non-programmers an intuitive graphical interface for building their own MR systems. AMIRE has been used to enable users to create step-by-step instructions for assembling furniture [285]. Users first associate ARToolKit [143] markers with real world objects before performing sequences of actions that will be recorded by the authoring tool.

MR Platform [266] is a commercial package that has been used to create a number of MR applications in entertainment. The package consists of a software development kit that provides tracking, rendering, and calibration utilities primarily for the development of AR systems. In addition, it requires a specialised See-Through HMD, named ST-HMD, for interacting with the developed system.

Another commercial development tool is the Unifye SDK produced by *metaio* [8]. The tool comes complete with different video capture, tracking and rendering technologies for creating AR content that can be presented on different devices/displays such as mobile phones, computer monitors, HMDs, and projectors. Practical applications of *metaio*'s AR technology can be seen in industries such as automobile manufacturing [209, 203].

Other than AR based MR systems, MXR Toolkit [16] is an open source software package that is capable of creating a variety of MR systems from both AR and AV technologies. The toolkit provides utilities for tracking, rendering, calibration, and background subtraction which enable virtual animated objects to be overlaid onto a real image (AR) and live 3D imagery of humans to be inserted into a virtual environment (AV). MXR Toolkit also supports creating tangible interactions with any synthetic objects inserted into the environment through the use of handheld instruments.

It can be seen that the majority of the development tools described above are designed for creating AR systems (with the exception of the MXR Toolkit), and tools for creating AV systems are not yet widely available. In addition, these tools are designed for creating visualisations that provide minimal support for facilitating interactions between real and virtual components.

2.4.3 Mixed Reality in Robotics

MR has recently been applied to robotics, primarily for visualisations. In particular, the various AR techniques presented in Section 2.3 have received growing interest due to their ability to visualise complex robot and task relevant data in geometric registration with the physical environment. On the other hand, AV has more commonly been applied for visualisation in robot teleoperation tasks.

The remainder of this section shows how MR visualisations, as well as the concept of MR, can be applied to assist remote operators in navigational tasks, improve social interaction between the human and the robot, and help in debugging and simulation of robot systems.

Teleoperation

AR has long been used to aid teleoperation and monitoring of robots, e.g. [181, 54, 251], by enhancing the remote operator's view of the robot environment with virtual information or visual cues to increase situation awareness. A recent example is presented by Sugimoto *et al.* [251] who use an AR interface to help operators understand the robot's geometric location and spatial relationship with the environment. The AR interface is able to provide a synthetic exocentric view of the robot vehicle by superimposing a graphical robot model over images captured by the onboard camera.

As opposed to using AR for assisting robot teleoperation, Chen *et al.* [71] use in their AV system a predominantly virtual environment to visualise live robot sensory readings. By using prior measurement data of the building where the robot is to be deployed, a virtual replica of the environment is constructed in an offline phase. During the online experiment, as the robot navigates in the same building, its real time laser sensor scans are visualised and mapped onto the virtual environment to indicate newly appeared objects.

Amstutz and Fagg [26] create an AV environment to visualise spatial information of multiple robots to remote users over the network. The environment is dynamically constructed based on distributed sensory readings, and the robots' locations are updated and reflected in real time in the virtual environment.

Nielsen *et al.* [190] propose an AV based MR environment that visualises multiple sets of data in a single integrated interface for increasing the remote operator's situation

awareness during robot teleoperation. The system overlays the robot's onboard camera data, and map information onto their corresponding geometric locations within a virtual environment. As the operator moves the real robot in the real world, the virtual environment is also updated accordingly. This improves the operators understanding of the spatial relationship between scenery shown in the live video imagery and its surrounding environment.

There are also immersive virtual environments for robot teleoperation which are very similar to existing AV environments. VirCA [13] is an collaborative virtual environment that enables users to telemanipulate robots and other sensor devices over the network. The virtual environment is synchronised with the real world and reflects the state of the remote robot and its physical environment. Video imagery of the remote environment is also immersed in the virtual environment. The virtual environment can be visualised through HMDs or projective displays.

Kelly *et al.* [145] propose a photorealistic virtualised reality interface for remote mobile robot control. Their system achieves dynamic, real time reconstruction of high quality, photorealistic models of the robot environment by exploiting the use of highly dense lidar data and an efficient data compression technique. The 3D video view of the scene is updated in real time as the remote operator drives the robot vehicle using a steering wheel to explore an unknown environment.

More examples of using virtual interfaces for robot teleoperation can be found in papers [129, 109, 105]

Human Robot Interaction

AR is widely used to help convey robot state information and improve HRI. As one of the ultimate goals in the field of social robotics is to deploy robots to work alongside of humans, it is important for robots to be capable of communicating with humans. AR is applied in this area to address the interaction problem by overlaying virtual information in the view of the user to help the user understand what the robot is thinking and doing. For example, virtual directional arrows are overlaid on top of robots to indicate the current robot heading [85].

Young *et al.* [283] use AR to improve HRI by visualising robot information inside bubblegrams to users wearing HMDs. The approximate location of the robot in the video image is first detected, 2D virtual bubblegrams are then superimposed near the robot.

Dragone *et al.* [96] present a system that describes robot states using virtual characters. In this work, animated virtual characters are placed on top of the robots to describe every action that the robot takes. The virtual character is able to express the robot states and convey them to users with more natural interaction methods such as emotions and gestures.

The HRI problem has also been addressed by providing robots with a realistic human-like appearance. U-Tsu-Shi-O-Mi [238] is an AR system that allows users wearing HMDs to look at the robot as if the robot were a human. This is achieved by overlaying a virtual human avatar onto a humanoid robot and keeping the motions of the robot and the avatar synchronised. The avatar is able to generate speech with synchronised lip movements and physically interact with the human.

An extensive survey on social robot interaction using MR can be found in [132]

Robot Programming

AR has also been applied to Programming by Demonstration (PbD). PbD in the context of robot programming is a growing research area that aims to enable arbitrary users to teach robots various tasks. PbD consists of two phases, demonstration and playback. AR has primarily been used during the demonstration phase to aid teaching of tasks by the user.

Pettersen *et al.* [205] use AR to improve the process of teaching industrial robots painting tasks by providing instant real time feedback of the painting results to the user.

Aleotti and Caselli [23] use a combination of AR and AV techniques in their PbD system for teaching robot manipulation tasks. AR markers are placed in the scene to construct a virtual model of the workspace and the objects to be grasped. Users wearing AR gloves are then able to demonstrate typical robot grasps, and the movements are visualised in real time in the constructed virtual environment.

Brageul *et al.* [53] present an AR control interface that enables users to intuitively teach robots navigational tasks by directly pointing to locations on a real world view of the environment rendered inside the AR interface (see Figure 2.5).

Physical Interaction

In addition to using purely visualisation techniques, simulation of physical interactions such as collisions between real and virtual objects are useful in a number of robot applications.

Interactions between physical robots and virtual objects can be seen in an educational robotics framework proposed by Anderson and Baltes [28]. The framework creates MR games such as robot Pac Man and robot soccer for teaching concepts in robotics to students. The setup consists of an overhead camera that tracks robots and other objects on a physical table-like platform. Given that the pose of all objects are continuously tracked in real time, the system simulates relatively simple 2D collisions between physical robots and a virtual ball. The interactions created are limited to a small operating area bounded by the table-like platform.

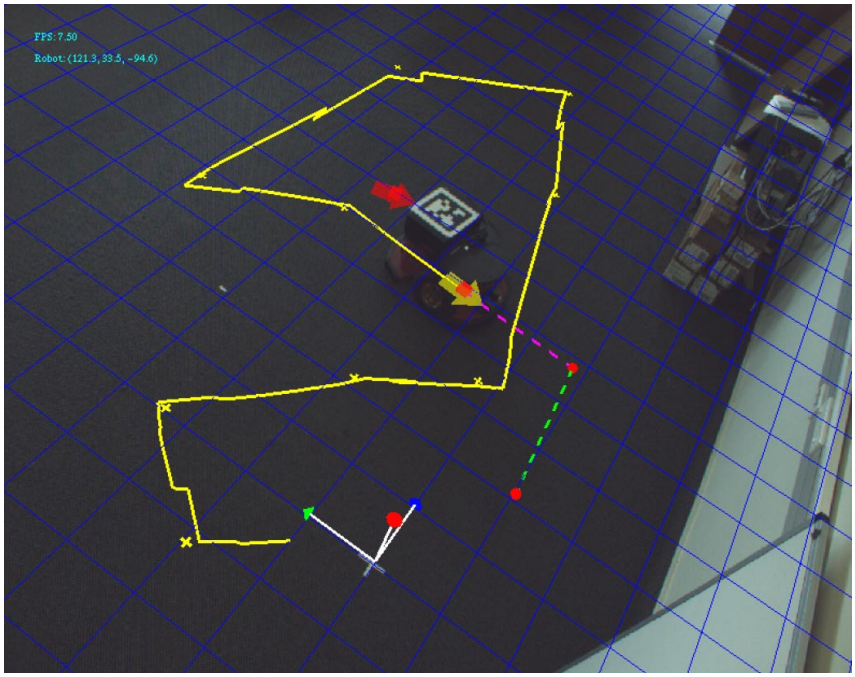


Figure 2.5: An AR control interface for programming robots by demonstration. The yellow crosses are the waypoints specified by the user during the demonstration phase. The yellow line and arrow describe the actual path taken by the robot, while the red arrow represents the current odometry reading [53].

A similar technology is used in the Mixed Reality Sub-League (formerly called Physical Visualization Sub-League) in RoboCup [261], which involves teams of physical thumb-size robots (Eco-be systems) engaging in soccer matches on a virtual simulated soccer field.

Barakonyi *et al.* [42] introduce animated agents into the real world that can interact with a LEGO Mindstorms robot for teaching the task of robot assembly. A touch interaction is achieved by detecting collisions between the bounding volumes of the virtual agent and the virtual representation of the physical LEGO robot. It is also shown that animations of a virtual repairman mounting a virtual wheel onto the physical LEGO robot can be created. The animated interaction is, however, predefined, and no physics simulation is used.

Robot Development

AR visualisation has been used in the robot development process to help developers debug and understand complex robot data. Collett and MacDonald [79] propose an intelligent debugging space that uses AR visualisation for overlaying robot data in context with the real world (see Figure 2.6). The robot developer is then able to understand the robot's view of the world and compares it with the ground truth of the real world image. The system uses a fixed overhead camera to track ARToolKitPlus markers attached to ground robots in the operating environment for overlaying the virtual robot information.

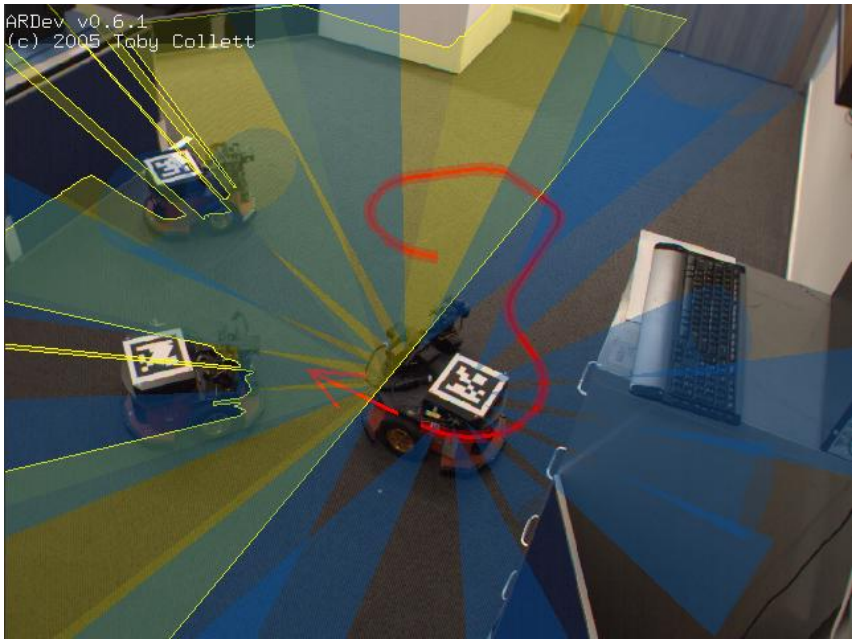


Figure 2.6: AR is used to render laser (yellow) and sonar (blue) sensory data in context with the real world. The red arrow represents the robot's path [80].

A similar technique is used by Kozlov *et al.* [157] to help robot developers debug SLAM algorithms.

Ong *et al.* [195] create an AR environment for immersive programming of robots. A developer wearing an HMD is able to see the results of its algorithm replicated on a virtual robot in the real world. The visualisation takes place over markers placed in the physical environment. It is proposed that this method provides developers the flexibility of choosing where the development takes place, from visualising miniature virtual robots in small scale mock-up environments to seeing full scale virtual robots working in the actual operating environment. However, the virtual objects introduced into the scene do not interact with the physical environment.

MR has also been applied to robot simulations to create hybrid or MR environments for safe and cost-effective testing and evaluation of robot systems. These experiments operate in a manner similar to the HIL simulation paradigm (see Section 2.1.3).

Stilman *et al.* [248] propose a hybrid experimental environment that simulates virtual components in a physical laboratory environment for decoupled testing of individual subsystems of a humanoid robot. The experimental setup uses a motion capture system that consists of an array of cameras for tracking objects labelled with 3D retro-reflective markers. This allows the geometry of the scene to be reconstructed, providing ground truth data of the objects' positions in the environment. With a representation of the real world available, virtual elements can be introduced to test planning and control algorithms. Kobayashi *et al.* [153] and Nishiwaki *et al.* [192] later named this an "MR environment", and apply AR visualisations techniques for extensive debugging of motion

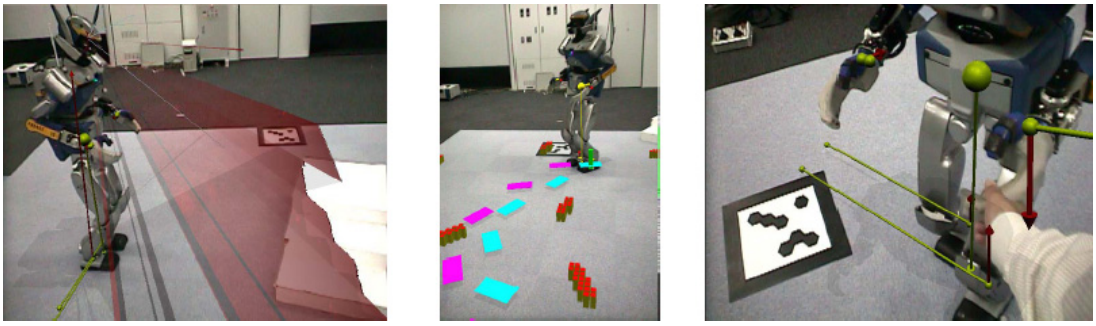


Figure 2.7: An MR environment that visualises the robot's sensory perception, planning, and motion control data over real world views [154].

planning and vision based recognition algorithms (see Figure 2.7). The system is however limited to operating in a carefully controlled and modified environment, which requires expensive hardware equipment, i.e. multiple cameras and position tracking devices.

Davis *et al.* [86] propose an Augmented Reality Framework (ARF) for simulating AUV operations. The work builds on previous work on HIL simulations for underwater robots, and extends the concept to provide a taxonomy of HIL simulation methods based on the Reality-Virtuality continuum. The simulator relieves the high expense of conducting experiments in hazardous underwater environments by allowing experiments to be conducted using real robots, simulated sensors, and simulated environmental objects. It incorporates a distributed communication framework that can load real and virtual modules interchangeably to enable HIL based testing, e.g. a virtual sensor mounted on a real robot platform. However, the ARF does not support interaction between virtual sensors and real world objects in the environment.

Göktoğan and Sukkarieh [116] integrate AR technology with a real time multi-UAV simulator for experimenting cooperative control of a team of UAVs in target detection tasks. The mission requires multiple sensors and UAVs which cannot be supplied with available resources, preventing further testing of these operations. The AR framework overcomes this limitation by augmenting the real UAVs' sensors with data from the simulated world. This allows the real UAVs to communicate with virtual sensors on real UAVs as well as completely virtual UAVs during the mission. Due to application-specific requirements, the proposed system does not consider simulation of physical interactions, or interactions between non-sensor components.

Evaluation of MR Systems in Robotics

While there appears to be an increasing interest in using AR for HRI [118, 132], thorough evaluations of such systems in robotics are not prevalent. This is not a surprise as surveys in the general field of AR also indicate that user evaluations of AR systems are underutilised [255, 101].

Green *et al.* [119] present an evaluation of their AR system for human-robot collaboration. The experiment investigated the use of the proposed multimodal AR interface for users to interact with a robot and guide it in a navigation task. Quantitative measures were taken to collect performance data that included time to completion, impending collisions, and the number of collision under three conditions. Results from statistical analysis indicate that the AR system allowed the users to carry out the task more accurately compared to direct teleoperation through an egocentric camera. Results from subjective questioning support the finding, suggesting that the AR system helped the users maintain spatial awareness.

Maida *et al.* [170] carried out a user evaluation of their AR interface designed to guide robot operators in an alignment operation. The experiment investigated the user under the assistance of AR overlays, and compared the results with those obtained from relying on the robot's onboard video alone. Quantitative analysis found that the use of AR overlays helped to improve the robot operators' performance, showing reduced position and orientation alignment errors and task completion times. Subjective results obtained from a questionnaire also indicate that the AR interface was helpful.

Evaluation of an AV system for HRI can be seen in the work of Nielsen *et al.* [190]. Their 3D AV interface for ground robot teleoperation is compared with a traditional 2D virtual interface through a series of user studies with different task complexity and information presentation methods. Quantitative measures were taken to collect data such as the task completion time, number of collisions, average robot velocity, and proximity to obstacles. Results from statistical tests indicate that the AV interface improved user performance and awareness in these robot control operations. Similar results were obtained in an evaluation of the AV interface for UAV teleoperation, showing an improved spatial awareness between the UAV, its video data, and the terrain data [97].

In the area of robot development, Collett and MacDonald [79] conducted an ethnographic study to investigate the use of AR visualisation for robot debugging tasks. Compared to the above work, the study is more of an observational one without the use of formal metrics. The study recruited robot software engineers to program a mobile robot system. The primary method taken to collect data from the experiment was by asking the participants to take notes of bugs and critical events they encountered during the development process. Based on the collected comments, the work discusses effects of the AR system that could influence usability; these include network latency, occlusions of AR markers, visual representation of data, and physical layout of the user workstation and the AR display.

2.4.4 Summary

The term MR has been used interchangeably with AR and AV in many existing works. Similarly, many MR development tools are based on AR technology. It is noted that MR has been used mainly as a visualisation technique in robotics, and its applications predominantly take the form of AR visualisation. Nevertheless, there are signs of interest in the robotics community to use AR/MR for more than visualisation. It was seen that the concept of AR can be applied to robot development, e.g. as in the case of some hybrid simulation systems. Similar to HIL simulation, AR based simulation systems have been designed for specific applications such as underwater and aerial robotics. These frameworks and implemented systems have yet to support seamless interactions between arbitrary real and virtual components.

2.5 Unmanned Aerial Vehicles in Agriculture

As mentioned in Chapter 1 robot applications in the field of agriculture are of particular interest and this section presents reviews and examines methods of robot development in this field. The goal is to recognise how robots can provide automation of various tasks in agriculture, and identify the specific tasks suitable for UAVs. This serves as a case study in a challenging area of robotic system development from which both generic and application specific requirements can be derived. Once the tasks have been identified, the different areas in aerial robotics are reviewed to understand the research and developments that are undergone for creating autonomous UAVs capable of completing these tasks. Tools for simulation of UAV systems are then examined.

The review in this section is intended to be general and not an exhaustive coverage of all existing work. It will later be analysed in Chapter 3 to derive requirements for improving the robot development process.

2.5.1 Robots in Agriculture

The agriculture sector makes a significant contribution to New Zealand's economy and accounts for more than half of the country's export earnings. However, agriculture is a labour intensive industry and farmers are sometimes required to operate in adverse agriculture environments or perform tedious and health threatening tasks such as spraying of chemicals, and cleaning of unsanitary environments. Labour is also expensive. There are trends that indicate a decline in employment and the lack the young generation farmers joining this workforce [17, 277]. This motivates the need for automation in agriculture. Robots, in particular, can serve as helpful assistants in the field of agriculture, reducing human labour while improving productivity and enabling effective utilisation of resources.

Farmers can then spend their time wisely to farm intelligently and produce high quality products which meet the high expectations of consumers.

Agricultural Automation

Benefits of automation in agriculture can be seen in the dairy farm industry. An Automatic Milking System (AMS) is an example of an automated application that significantly reduces the expensive labour requirements of farmers while increasing milk productivity. Studies reveal that not only have AMSs increased labour flexibility, they also improved the farmers' social life [90]. During the milking process, the AMS is capable of monitoring the cow's physiological information which provides indication of its heath and welfare. Information such as productivity, fertility, and health status are also automatically recorded into a database which can later be analysed by the farmer [83]. On the other hand, there are still improvements that need to be made to the AMS. In comparison to a conventional milking parlour, the concern is the negative effect on milk quality. A higher milking frequency increases free fatty acid and a longer period of milk storage in the milking equipment enhances bacterial growth [152].

Agricultural robots such as driverless tractors, transplantors, and harvesters can be found performing tasks including automatic planting, harvesting, nursery of vegetable and fruits, weeding, tilling, and spraying of chemicals [136, 127]. However, the cost and maintenance required to invest in these robotic platforms are generally high, and therefore, farmers are more reluctant to embrace this new technology. This is also because various agricultural tasks are season-dependent, and it is not cost-effective to purchase expensive equipment which operate only a few hours in a year [241]. For example, there are only a number of months in a year in which seeds can be planted by an automatic planter. In response to this issue, some agricultural robots have evolved to become multipurpose. For example, a grape production robot with multiple manipulators and vision sensors is designed to perform harvesting, spraying, bagging, and berry thinning operations [183]; similarly, a multi-operation robot working in a strawberry field is capable of harvesting, spraying, and grading agricultural products [31].

Vision based Agricultural Robots

Robots are often required to navigate their way over traversable ground while avoiding harm to the surrounding agricultural objects. Vision based navigation can offer a richer perception compared to noisy inputs obtained from time-of-flight sensors in agricultural environments. For example, Ortiz and Olivares [197] automate the navigation of robots by segmenting the camera images to identify a path between two rows of plants. This allows robots to traverse through the agricultural fields for collection of plantation data.

Similarly, Ollis and Stentz [194] build an automated harvester that follows the boundary between cut and uncut hays. Camera images are processed to identify and track the crop line that separates cut and uncut regions while detecting obstacles in the robot's path.

Vision is also used in recognition tasks in agriculture. For example, by using cameras mounted on the end-effectors of robot manipulators, the robot is able to locate and recognise fruits based on reflectance, shape, and size information from the input images. This guides the movement of the manipulators to perform picking operations. The technique has been used to harvest tomatoes [257, 117], apples [56, 288], oranges [216], grapes [183], and watermelons [228].

On the other hand, Andersen *et al.* [27] use vision guided robots for pig house cleaning in order to maintain high quality livestock health and food products. The camera images are first processed to inspect for dirt and manure in the environment then a high pressure cleaning tool is applied to wash these areas.

In an indoor controlled environment where there are limited or no navigation requirements, the use of vision can concentrate on analysing agricultural objects. For example, objective grading is performed on fruits based on colour, shape, and size. Vision can also be used for inspection of crops for defects, germs, or other viral activities to ensure product quality [242, 156, 161]

UAVs in Agriculture

Other than ground robots, UAVs have been demonstrated to be an ideal candidate for outdoor remote sensing and monitoring in agriculture. While other alternatives exist such as satellite imaging and ground-based sensing, UAVs offer a more economic and flexible option. Images captured from satellites have spatial resolutions which are too low to distinguish between features such as crops from soil. Satellite imaging is also susceptible to cloud cover and poor weather conditions. Ground-based robots or vehicles have a lower mobility and move far slower than aerial platforms, and are sometimes unsuitable for traversing between narrow rows of crops or muddy soil [252].

In 1987, Yamaha [14] introduced the world's first industrial unmanned helicopter, R-50, with a 20kg payload for dusting of rice paddies in Japan. The remotely controlled unmanned helicopter was able to spray pesticides and specialised nutrients over large scale paddy fields and at areas that are otherwise difficult to access by a human farmer. The unmanned helicopter was shown to be a cheaper and faster alternative to manned helicopters in carrying out such tasks while also providing high quality productions of food crops.

Other than the spraying of crops, UAVs have been widely used for remote sensing in agriculture. One application is the mapping of vegetation. The maps generated are useful for farmers or domain experts who can then thoroughly analyse the information

offline. For example, Sugiura *et al.* [252] mount an imaging sensor on an unmanned helicopter for capturing images which are used to generate Geographic Information System (GIS) maps for the analysis of crop status. Maps can also be used to detect invasive vegetation in order to plan site-specific herbicide applications. The NASA solar-powered Pathfinder-Plus UAV [128] has been used to capture high resolution colour images for mapping invasive weed outbreaks and for revealing irrigation and fertilisation anomalies in coffee plantations. Bryson *et al.* [55] also used images captured by a low flying UAV to reconstruct large scale terrain maps for invasive plant detection and biomass mapping.

Another application in agriculture is the monitoring of animals. Video imagery collected by UAVs has been used in the management of wildlife population and species. Jones IV *et al.* [140] deploy a small fixed wing UAV to collect video imagery for surveying midsize vertebrates and habitats in Florida. Similarly, Grierson *et al.* [120] also made use of aerial imagery from a fixed wing UAV for surveying the population of kangaroos in Australia. High resolution thermal images are also taken to detect kangaroos that take cover from heat under trees.

2.5.2 UAV Robotics

It was seen that UAVs are ideal for automation in agriculture due to their high mobility. They are also an inexpensive alternative to manned aerial vehicles, replacing the presence of humans in hazardous or repetitive tasks.

The birth of UAVs dates back to 1916 when an automatic control system was developed for the Curtiss Flying Boat. Substantial development of UAVs began during the World Wars [82]. The war sparked the call for pilotless aircrafts for aerial bombing operations. Since then, UAVs have been developed for the delivery of nuclear weapons and reconnaissance in hostile areas [253]. As time progresses, the focus in UAV development has branched and contributed to important civilian applications such as search and rescue, surveillance, border patrol, aerial imaging, communication relay, and remote sensing.

Flight Control

One of the main research areas in aerial robotics is flight guidance and control. There are many researchers who design and assemble their own onboard avionics systems and vibration isolation systems, e.g. [51, 59, 113], and the first step requires understanding of the flight dynamics and constraints of these custom built UAVs which influence their maneuverability in air. The findings are used to develop a flight control system that allows for flight stabilisation and position control, and provides smooth navigation that adapts to different flight conditions. This basic level of autonomy in navigation enables other developers to concentrate on high level navigational algorithms such as path planning and

obstacle avoidance.

Mission control has commonly been achieved with human operators monitoring the status of UAVs and manually altering the mission plans as required. Today, efforts are put into introducing onboard intelligence that minimises human intervention in mission planning and execution. Given a set of goals, the UAV is required to dynamically plan, make decisions, and take appropriate actions using limited resources and knowledge of the environment. Artificial Intelligence (AI) presents a viable solution to various UAV control and planning problems. AI techniques, such as simple rule based systems, fuzzy logic, neural networks, and evolutionary algorithms, can be used to govern the behaviour of UAVs by mimicking human expertise in reasoning and decision making. AI has been applied in low level flight control [250, 100, 102], path planning [232, 167], optimization of energy use in missions [124], and multiple UAV management [211, 244].

Ground Communication

In remote sensing, surveillance, and reconnaissance applications, a good communication strategy helps to make effective decisions. Long ago, UAVs were sent for surveillance and reconnaissance operations and the information could only be obtained after the return of the vehicle [253]. This was inefficient and motivated new UAV telecommunication strategies.

Satellite relay for communication between UAVs and the ground control stations was originally used in military operations for issuing of commands, and transferring of data that the UAV collected at an extended range. The downfall is the apparent transmission delay. On the other hand, the UAV itself can also act as a relay node for wireless communication over mountainous or hostile regions. The method involves deploying a group of UAVs as mobile airborne stations that route data to and from mobile ground users who can not be reached by ground point-to-point terrestrial links [208, 34, 121, 95, 287]. It presents an alternative option to satellite and terrestrial communication systems with the potential to offer more adaptable, movable on-demand, and high data rate communication.

Mulitple UAV Coordination

There is a growing research interest in coordinating multiple UAVs. Advances in computing and wireless network technology have made it possible for groups of UAVs to collaboratively carry out tasks and accomplish a global mission [227]. Research topics in cooperative control of multiple UAVs include formation flights, collision avoidance, and task allocation strategies.

When a group of UAVs cooperate on a task, the coordination of their motions results in flight formations. It is required that the formation reconfigures safely in response

to changes in environmental status, mission objectives, or the number of UAVs. Control strategies are developed by considering the dynamics models of UAVs, relative constraints between the vehicles, and time restrictions [286].

Collision avoidance includes avoiding inter-UAV collisions as well as collisions with objects in the environment. Research on path planning of formation flight addresses this by incorporating safe cooperative navigation algorithms which take into account the relative position measurements of UAVs in the group. Constraints such as the minimum allowable distance between UAVs, the maximum velocity or acceleration, and bounds on turning radii are set to enforce flight safety and aid in the selection of optimal paths [265]. Flocking formation of a group of UAVs is an example which ensures collision-free, tight clustered movement along a common heading direction [259, 260].

A typical cooperative UAV mission has an overall task that can be broken down into smaller sub-tasks. Task allocation strategies are used to assign these tasks to different teams of UAVs. Mapping of tasks to UAVs needs to consider the nature of the tasks which can have different priorities and levels of risk, and the heterogeneous nature of the UAV platforms which can have different sensing and maneuver capabilities. Planning and execution of tasks should be flexible. A solution is to allow for re-planning due to changes in the environment or unforeseen situations. Alternatively, robust planning algorithms are designed to take uncertainty into account to avoid the effect of churning; a problem caused by continuous re-planning at each time step due to noise in the latest perceived data [24].

UAV-UAV Communication

Another important factor for successful collaborations between UAVs is a reliable communication framework. UAV-to-UAV communication is concerned with sharing of resources such as sensor and map information, UAV states, and mission status. There are two main information management schemes for a swarm of cooperative UAVs: Centralised and Decentralised/Localised information management [142, 265, 67]. Centralised information management essentially uses a central database to store all data collected by UAVs. It provides a global view of the state of the world which all UAVs must access to acquire the most recent information, hence, a reliable communication link between the UAV and the central database is essential. In contrast, decentralised/localised information management relies on the UAVs to each carry a partial view of the world. In this case, information is communicated between neighbouring or relevant vehicles. The decentralised/localised strategy is more suited to missions involving multiple teams of UAVs, each assigned with a different objective, and thus, requires different information.

An issue in UAV-to-UAV communication is the problem of maintaining a high performance communication link due to the relative high mobility of both ends in the commu-

nication channel, which results in a larger Doppler Spread [278, 78].

2.5.3 UAV Simulation

Various research areas in aerial robotics were identified, however, the complex dynamics of UAV systems and the high risk tasks required of UAVs also present a challenging and time-consuming development process. Performing real world tests can be difficult. Test flights are expensive operations, requiring a substantial amount of human effort, technical support, and hardware equipment. Local air safety and flight regulations further impose restrictions on flight areas. Simulations can provide a safe and easily accessible method for testing of UAV systems.

In comparison to ground robot simulations, solutions for UAV simulation are predominantly application or platform specific. This is mainly because different UAVs exhibit different flight characteristics which are difficult to simulate by a general model. In many cases, the development of UAV simulators is considered an integral part of the development cycle, i.e. develop a new simulator or modify the existing simulator to adapt to the new application. The approach may be suitable for rapid development of small projects, but for use in a long term research, it leads to duplication of efforts. Overall, there exist a fewer number of general purpose UAV simulators widely used around the world compared to ground robot simulators. Section 2.1.3 and Section 2.4.3 described hybrid and AR based simulations for UAV development, and this section intends to identify other types of simulation tools that are more commonly used in the research community.

Flight Dynamics Simulation

Simulation of UAV dynamics requires heavy mathematical modelling and is generally computationally expensive. Matlab/Simulink is a numerical simulation tool that has commonly been used for such tasks, and enables researchers to develop and evaluate UAV control algorithms. MultiUAV [215, 214] is an example of a UAV simulator based on Matlab/Simulink developed by the U.S. Air Force Research Laboratory. The simulator provides six DOF vehicle dynamics models, target/threat models, and inter-vehicle communication models that allow developers from different application domains to evaluate algorithms on cooperative control, trajectory planning, and multi-UAV communication. MultiUAV has later been successfully extended for simulation of military operations in a battlefield environment [191].

There are also generic UAV Flight Dynamics Model (FDM) libraries available that can be integrated for the development of flight simulation tools. AeroSim Blockset [12] is a freely available Matlab/Simulink block library that provides non-linear six DOF aircraft dynamics models, linear aerodynamics models, propulsion models, and atmosphere models

such as background wind and turbulence. JSBSim [7] is a similar FDM library, but in comparison, it is an independent, open source programming library designed to be easily integrated with any software programs. Notably, JSBSim is the original flight dynamics library for the popular flight simulator, FlightGear [108]

Commercial UAV simulators, such as Simdrone [122], and the CAE UAV simulator [5], offer high fidelity simulation. These integrated products provide a rich database of FDMs, realistic environment models, and ground control station modules. However, they are mainly tailored to training operations involving a human operator.

Simulation with Flight Simulators

Extending existing general purpose flight simulators presents an efficient and rapid approach to the development of UAV simulators, and saves developers time and effort from creating their own simulation tools from the ground up.

Flight simulators such as the commercially available Microsoft Flight Simulator [176] and X-Plane [174], and open source FlightGear [108] are computer based flight simulation tools widely used in the research community, e.g. [112, 65, 210]. These flight simulators provide a variety of features for customising aircrafts models, environment models, and interfacing with other software tools. As the result, UAV simulation can be created by tuning the flight dynamics parameters of a custom aircraft to exhibit characteristics similar to those of UAVs.

Many flight simulators also support interfacing with external control modules. The flight simulator is used to provide high quality visualisation of the resulting flight behaviour of an aircraft controlled by external inputs. For example, X-Plane is used to provide real time visualisation of the UAV flight behaviour based on outputs from an HIL simulation [173].

Due to the high quality visualisation and computationally expensive modelling of flight dynamics incorporated in flight simulators, simulations are generally limited to one or two UAVs per computer. The other limitation of building UAV simulations from flight simulators is that they do not support simulation of robot sensors, such as a thermal imaging sensor or a laser rangefinder, that could potentially be carried onboard the UAV.

2.5.4 Summary

Robots tasks in agriculture were found to be diverse, and robots are often required to operate in harsh, dynamic environments interacting with agricultural objects that can be delicate. On the other hand, UAV tasks in agriculture focus on spraying, remote sensing, and monitoring operations, and there are less direct physical interactions with objects in the environment.

Research areas in aerial robotics are largely related to improving the navigational capabilities of UAVs and the communication strategies for accomplishing their missions. It was also found that the high mobility and instability of the platform present greater risks during UAV operations, and thus result in the need for a more thorough development process. There are existing work which focus on delivering simulations with highly accurate flight dynamics but the problem is the proportionally high computational demand.

2.6 Discussions

This chapter provided a literature review that focuses on several areas related to this research, including robot simulation, visualisation, MR, and aerial robotics. The review provided a number of key findings as will be discussed below:

There has been considerable work in the area of virtual robot simulation. Many existing tools have been designed to be general purpose, providing support for dynamics, kinematics simulation, as well as various sensor and environmental models. However, there are limitations when using existing virtual robot simulation tools to provide real time simulations for robot systems that need to meet strict timing requirements[1]. Expensive computations from the physics engine can be a limiting factor. Consequently, it may lead to difficulty in integrating real hardware components in simulations. A missing global clock for controlling the execution cycle of all components participating in the simulation can be another limiting factor for simulating time-critical robot systems.

In contrast to virtual robot simulation, hybrid simulation tools are less widely distributed as each tool is customised to its own robot task, platform, and system architecture. Hybrid simulation tools are more commonly used for the development of expensive robot systems in high risk operations. While both virtual and hybrid simulation approaches rely on the use of numeric simulation models, there are no signs indicating the sharing of technologies between the two fields. An interesting note is that more recently, the idea of applying AR/MR to aid testing of robot systems has been proposed, primarily by augmenting the robot's perception to reflect virtual inputs in a similar way to HIL simulations. Again, the proposed frameworks are primarily application-specific.

Little effort has gone into validating virtual and hybrid robot simulation tools. Quantitative assessment of a simulator's accuracy is important to determine whether the tool can be used reliably and produce results that are portable to the real world. The evaluation metrics used in validating USARSim [61] and its comparative analysis of results with real world settings provide a good approach to evaluating robot simulation tools. On the other hand, AR/MR based hybrid simulation tools could benefit from the evaluation

[1]Timing issues in creating real time simulation were frequently discussed, for example, in Gazebo and OpenRAVE forums. The author was also involved in one of these discussions

methods reviewed in Secion 2.4.3. The ethnographic study presented in [79] could help to identify less-trivial events in robot development tasks. However, the study was a minimal one in terms of the number of participants and the metrics used. Other more formal methods for evaluating of MR systems in HRI should be employed to collect stronger empirical evidence of any potential benefits of an AR/MR based hybrid simulation tool. For example, user studies with objective measures, e.g. task completion time and accuracy, could help to identify whether the system improves user performance, while the use of subjective questionnaires could gain an insight into user acceptance of any new concepts and technologies derived from this research.

The review of visualisation provided an understanding of the human factors involved in designing visualisations, and revealed a wealth of proven existing techniques which can be readily applied to aid understanding of robot data. AR in particular is a technique that helps to create intuitive views of complex data in real time. Evaluations of AR for HRI in Section 2.4.3 have found that AR visualisations improve user performance, e.g. increase accuracy in alignment [170] and navigation [119] tasks. The downfall of AR visualisation is that it has primarily been achieved in carefully controlled environments and/or with expensive hardware setups.

It was found that while UAVs are ideal candidates for a number of tasks in agriculture, the development process is risky, long, and costly. Considerable effort has been put into devising a safe and reliable method for transferring results from simulation to reality in order to minimise the chance of failure in real world tests. In addition to HIL simulations, high fidelity but computationally expensive dynamics simulations are crucial to provide accurate models of the UAV behaviour in the real world. The next chapter provides a more thorough analysis, focusing on the UAV development cycle in detail to identify requirements for building a development environment that can improve the robot development process.

In summary, the literature review has so far identified gaps in the following areas that invite further research:

- Real time robot simulation and the ease of integrating real components in simulation tools
- Reuse of HIL simulation for a broader range of robot applications
- Formal evaluation of robot simulators
- Application of AR to less controlled, dynamic robot environments

3

Design of an Environment for Robot Development

Chapter 2 provided a review in the area of aerial robotics and identified various development methods for UAV systems. In comparison to general mobile robot simulation tools, the development of UAVs relies on customised tools and equipment that are rarely used in other areas of robotics. But how do the user and functional requirements really differ? This chapter continues the case study of UAV robotics in agriculture and conducts an analysis on the robot development cycle of UAVs to understand how robot developers experiment with UAV systems, the environments that these experiments take place in, and the strengths and weaknesses of each method. The findings from the analysis can help to identify specific user, technical, and safety requirements for designing an improved robot development environment that provides a more generalised and reusable solution to experimenting with both simple and complex robot systems.

Section 3.1 begins the analysis with the case study in the development cycle of UAVs then focuses on identifying issues in the development process. Section 3.2 presents the requirements for designing an improved robot development environment. A number of features are necessary to provide support for building the new robot development environment, as will be described in Section 3.3. Finally, the role of MR in robot development is introduced in Section 3.4.

3.1 Analysing the Robot Development Cycle: Case Study in the Development of UAVs

To identify the requirements for this work, it has been specifically chosen to focus in the area of aerial robotics. The literature review described methods for the development of UAV systems. This section begins by showing how the methods relate to each stage of robot development. It identifies key issues that have been addressed by existing development environments, then presents outstanding issues.

3.1.1 Understanding the UAV Development Cycle

First, the UAV development cycle is examined in detail. A typical robot development cycle can be divided into seven stages as described by Seward and Garman [236]. As this thesis specifically focuses on the design of a development environment, the first three stages presented in [236] on problem definition, knowledge acquisition, and requirements definition have been condensed into one stage, resulting in five stages presented below. In each stage, a brief description is first given of the robot development process in general, followed by a description of issues and development methods specific to UAVs in agriculture. Note that this UAV development cycle assumes the final robot system has a certain level of autonomy and is to be deployed in an application, as opposed to a development cycle that focuses solely on the mechanical system without the integration of high level software components. This analysis has been put together by studying materials found in the literature provided in Section 2.5.

Stage 1 - Requirements Analysis: Like the development of a computer software system, a typical development cycle of a robot system begins with problem definition and requirements analysis [236]. A project plan is prepared indicating estimates of time, budget, and resources required.

A similar process is taken in the development of UAV systems. From the user requirements, the project determines the tasks suitable for UAVs and specifies mission objectives. It was identified that the nature of aerial robot tasks differs from that of ground robots; UAVs have a higher mobility and are deployed for tasks that take place in large, outdoor environments. Functional requirements of the UAV system may also differ from ground robots. For example, UAVs are often required to move out of the user's line-of-sight during the mission, and thus a higher level of autonomy or remote monitoring interfaces may be necessary. It is also necessary to identify specialised instruments required for specific agricultural tasks such as remote sensing, and determine whether these equipments are of appropriate weight and size to be carried on UAVs. Safety requirements are also im-

portant for UAV operations. Fail-safe procedures are necessary and should be carefully designed for any untrained users, such as farmers.

Stage 2 - System Design: The requirements analysis is then used to determine the functional specifications that drive the design of the system architecture. Certain development environments are available for formal specification and verification of software systems.

Based on the requirements specified in the previous stage, the design of an UAV determines the appropriate aircraft model, size, payload, power, onboard sensors, and communication systems needed. The design may also need to consider local flight regulations. For example, the Civil Aviation Authority of New Zealand [15] provides rules governing the noise and emission levels of model aircrafts, which may place a restriction on the type of power source that the UAV can use. The software system design should take into account the available processing power of the onboard avionics system, and may need to offload expensive processings to ground stations. No specific development environments are common in this stage of the UAV development cycle.

Stage 3 - Low Level Autonomy Development (Prototyping): A prototype of the robot system is to be developed. Development and testing may be first performed using a virtual simulation tool. It is also useful to create software and hardware models. In general, the prototype may simply use off-the-shelf components to demonstrate a proof of concept. The prototype will also help to identify refinements to the requirements.

The development of a UAV system in this stage focuses on providing the UAV with a basic level of autonomy. Assuming that the basic body of the UAV airframe has been acquired or built, prototyping begins by developing an autopilot system with low level control of the UAV, ranging from attitude control, altitude hold, to position control. Low level control requires a model of the dynamic system to be developed and its parameters to be identified. This can be carried out by using model identification tools and working with data collected from manual test flights, e.g. [162], or by adopting and tuning FDMs from publicly available flight simulators, e.g. [112]. As the result, a closed-loop controller can be designed to provide flight stability. Offline simulations are commonly used for evaluating the performance of the controller. Online simulation methods may be used to identify issues that may arise from interactions between the components in real time by testing the control system on a desktop computer with simulated hardware, or on the actual hardware with simulated data (HIL) [186]. Simple test flights in the real world may be necessary before proceeding to the development of advanced UAV control algorithms. For certain UAV models, such as smaller aircrafts with Vertical Take Off and Landing (VTOL)

capabilities (helicopters) it may be possible to perform testing of flight stabilisation in indoor, controlled environments [50]. Testing of larger UAV platforms or control systems relying on GPS data, may need to take place in open clearances in outdoor settings under weather-permitting conditions. Implementation of other application-specific algorithms, such as vision based sensing and mapping, may also begin in this stage, and they can be tested in virtual simulations.

Stage 4 - High Level Autonomy Development: The findings in the previous stage help to refine the design of the robot system. The robot will be enhanced with core functionalities as specified in the requirements. Implementation and testing of robot components may take place in parallel, primarily in controlled environments to minimise the overall cost and risk involved.

The development of a UAV system at this stage focuses on the implementation of high level navigational algorithms, making use of the basic autonomy provided in the previous stage. Various custom UAV simulators are used for the development and testing of these components. HIL simulations are extensively used in this stage to test the onboard avionics system in response to simulated data or data collected from previous test flights. However, HIL simulations often do not involve deploying the physical UAV platform in flight because the risk is generally higher when testing high level algorithms [139]. Testing and evaluation of application-specific algorithms may take place in physical mock-up environments with representative physical models, e.g. placing sample crops for evaluating image processing or sensing applications [197]. Once the UAV system has been tested extensively on ground, further testing may need to be conducted with the actual aircraft in flight in carefully controlled open fields. If significant deficiencies are observed in the system, the development falls back to further debugging and testing of individual components on ground.

Stage 5 - Final Testing and Tuning: The robot will have all the functionalities as required. Thorough testing needs to be performed in conditions close to those of the deployment environment. The development of the robot system needs to be finalised with tuning and optimisation.

More field tests will be conducted for testing and evaluation of the UAV system. If possible, the UAV should be tested in the environment it is intended to be deployed. This is important for evaluating application-specific algorithms. For example, animal and vegetation monitoring operations require the UAV to autonomously navigate in grasslands where these agricultural objects exist, e.g. [128]. Difficulties in carrying out experiments in such environments include limited access-to-field, expensive transportation arrangements (for the development team and the UAV platform), and safety assurance of the experiment. Field tests must strictly adhere to local flight regulations.

3.1.2 Key Issues and Existing Approach

The overview of the UAV development cycle provided an understanding of how various development methods fit into the overall process. It shows that UAVs are an example of a complex and expensive robot system which requires a more thorough and longer development process. Looking at the development cycle, there are two apparent issues that can be raised.

Safety: It was seen that safety is a prime issue in the UAV development cycle as it involves testing of high risk robot operations. During the development, humans are involved in supervising, operating, and observing the UAVs, and faulty or unexpected behaviours can pose several threats to the humans. Failures of UAVs can consequently result in collateral damage to property and loss of life of persons on the ground or in proximate aircraft [15].

Cost: Testing and evaluation of UAV systems use various methods and take place in different environments throughout the development cycle. Considerable time and efforts are often invested in building simulations and setting up controlled environments to provide a safe testbed for experimentation. Extensive testing may also require designing a wide range of scenarios representing typical and unusual situations in order to discover different behaviours of the robot which have not been foreseen. However, in the development of UAVs, and also other expensive robot systems, the amount of testing can often be limited by resource, time, and budget constraints, which in turn affects the quality of the system developed.

Another important issue is the difficulty in creating realistic test environments that can reliably represent the real world. The real world contains complex variations that are too difficult or expensive to simulate or replicate in mock-up environments.

To produce reliable results and also to address issues related to cost and safety, it can be seen that existing work carries out incremental developments. The robot development cycle is shown to be an iterative process. Particularly in stage 3 and stage 4, there is a clear trend showing that, as the development progresses, the experiments take place in environments with properties closer and closer to those of the real world, or specifically, the deployment environment. By making incremental developments, it reduces uncertainties when porting the results from one development stage to another and minimises the chance of failures in real world tests. The literature review identified that this has been achieved either by increasing the fidelity of simulation through accurate dynamics modelling (see Section 2.5.3), or by including more real world components into the experiment as in the case of HIL simulations [139, 186] (see Section 2.1.3). The amount by which the level of reality increases is closely dependent of the overall risk involved during the transfer of

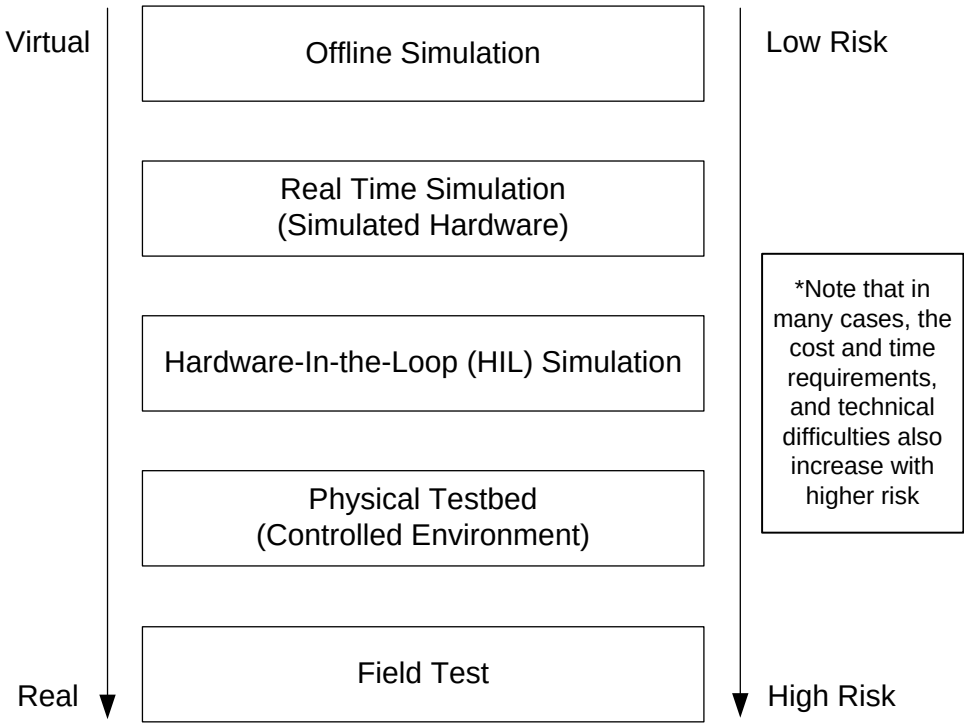


Figure 3.1: Development methods in the robot development cycle.

results in each step. In the development of UAVs, finer steps are taken due to the higher risks involved. Figure 3.1 shows the development environments used throughout the UAV development cycle, illustrating a transition from simulation to real world tests.

The development environments can be categorised into three groups: Virtual Environment, Hybrid Environment, and Real Environment. Each has its corresponding advantages and disadvantages as summarised below. This analysis has been put together by studying the different robot simulation methods and tools described in Section 2.1 and Section 2.5.3. In addition, other sources that helped to formulate the list below include [137, 267, 41] for the Virtual Environment category, and [160, 135, 172] for the Hybrid Environment category.

- Virtual Environment (Virtual Simulation, and Real Time Simulation)
  - Advantages
    - Simulate unavailable/expensive/dangerous components
    - Complete control over all objects, variables, and configurations, i.e. enable repeatable experiments with identical or different parameters
    - Provide safe environment for experimentation of failure cases.
    - Allow time manipulation, e.g. speed up and slow down simulation, record/-playback events.

  - Disadvantages
    * Coarse models produce unreliable results
    * High fidelity models are computationally expensive
    * Developing models takes time

- Hybrid Environment (HIL Simulation, Hybrid Simulation)
  - Advantages
    * Generate repeatable tests involving real components
    * Enable implementation and testing of components in parallel
    * Substitute physical components with simulated ones for cost and safety benefits
  - Disadvantages
    * Initial development of simulator can be expensive
    * Incompatible simulated and real components need redesigns to interfaces
    * Inconsistent execution rates between simulated and real components may impact the behaviour of time-critical systems

- Real Environment (Physical Testbeds in Controlled Environments, and Field Test)
  - Advantages
    * Experiment in real operating conditions for increased reliability of results
    * Provide testing with real world variations to discover unforeseen behaviours of the system
    * Create test scenarios that are difficult or expensive to model by a computer
  - Disadvantages
    * The real world is unpredictable and operations can potentially be dangerous
    * Demand resources such as fuel, battery, spare hardware parts, etc.
    * Take time and effort to setup and run experiment while ensuring safety
    * Experiments may not be repeatable

These advantages are perhaps well-known to many robot developers, and have been well-utilised in practice to address issues related to safety and cost, while providing a way of transitioning towards a more physical world for testing. They also implicitly reflect the needs of the robot developers and describe the features that a robot development environment must support.

3.1.3 Outstanding Issues

While different development methods have been proposed in existing work to address the issues described in Section 3.1.2, there are also limitations of each method. The disadvantages of each category of methods propose areas that require improvements. Virtual simulations are a benefit for prototyping both UAV control algorithms and application-specific algorithms as they offer the safest environment for development but they cannot completely emulate the real world causing inconsistencies in the results produced. HIL or hybrid simulations have been used to facilitate safe and cost-effective experiments, but due to issues related to the initial cost and effort required in preparing the simulation environment, they have not been commonly used other than for the development of expensive robot systems. Testing and evaluation of UAV systems in the real world help to validate simulation results and identify places that need tuning and optimisation to increase the performance and robustness of the system, but the process can be dangerous, costly, time-consuming, and resource demanding.

Above all, the disadvantages only represent a fraction of the problems, as they only consider the limitations of each development environment individually. While different methods are used to create environments of different reality (i.e. the three categories of environment) and facilitate transfer of results to the real world, this does not guarantee a smooth transition process. Problems are that:

- different development environments may rely on the use of different tools, and new users may need a considerable amount of time to learn how to use each tool,
- moving to a different development environment requires time and effort to re-setup experiments and equipments specific to that environment,
- additional tweaks and modifications to robot programs are often required before porting to another development environment,
- results from one development environment may not always be transferable to another due to the differences between the tools used.

The issues raised in this section need to be addressed for an improved robot development process.

3.2 Enhancing Robot Development Environments

The findings in the previous sections are the result of an analysis in the development cycle of UAVs. Outstanding issues were identified which suggested areas that could be improved are: the reliability of simulation results, the reusability of HIL simulations, and

the safety of real world tests. At the same time, a smoother transition process from simulations to real world tests is necessary.

This section proposes the following three requirements for implementing the suggested improvements.

1. A Common Simulation Environment

2. A Flexible Scenario Design

3. Context Awareness

First, it was seen in Figure 3.1 that different development methods were used to bridge the gap between simulations and real world tests to ensure a more reliable transfer results to the actual robot operation. However, this requires an indefinite amount of user time and effort to port results from one environment to another. Thus, this is motivation for creation of a method that can potentially enable developers to experiment in a common environment throughout different stages of the robot development process.

The second requirement is derived to address a similar issue as the first, targeting at facilitating incremental development. In addition to a common simulation environment, simulating different levels of reality requires a highly flexible and modular approach [172, 86]. It was seen in Section 2.1.3 that HIL simulations are able to increase the flexibility of experimentations by allowing unavailable, dangerous, or expensive components to be substituted with simulated counterparts. For example, testing an agricultural monitoring operation in an environment with real animals is far too dangerous in many stages of development, and the developers could choose to replace these components with safe alternatives that can be better controlled, such as simulated counterparts or physical surrogates. Using an HIL based simulation method can potentially facilitate incremental development, provided that its own limitations are addressed.

The last requirement is derived specifically from the need for addressing safety issues encountered in the experimentation of high risk robot operations such as UAV field tests. In the field of HRI, safety is closely related to human-robot awareness which is defined to be the human's awareness of the robot's identity, status, activities, locations, and surroundings [98]. Case studies in the robot search and rescue domain have shown that awareness violations lead to more critical incidents that cause damage to the robot, the victim, or the environment [98, 234]. Thus, to improve safety of robot experimentation, this research proposes the need for increasing the developer's awareness of the robot and other components in the experiment.

Although these requirements are the result of the analysis from the case study in UAVs in agriculture, it is important to point out that the robot development environments in UAV development share many of the common objectives with those in the general robotics

domain. The goal is to provide a development environment that can accurately, cost-effectively, and safely emulate reality for experimentation - the very same goal that applies to the robot development process in many other application areas. It is hypothesized that many other areas in robotics have similar needs, and thus the requirements in this section have been purposely designed to be generic so that it could be generalised to other applications.

3.2.1 Requirement A: Common Simulation Environment

To provide a reliable transfer of results and a smooth transition process, it was proposed that a development environment capable of enabling experimentation in a common environment across different development stages is needed. To do so, it is necessary to provide the robot developers with the ability to control various properties of the experimentation environment. These are properties that determine the degree to which the environment represents the target environment in the real world. Depending on the progress and the requirements, the robot developers could vary the complexity of the simulation models to obtain more accurate results at the expense of additional computational cost.

Attributes that should be given control of include physical and functional properties of objects in the experimentation environment, and they could be varied to affect the outcome of the experiment. Physical characteristics such as shape, colour, and size of objects are important in certain robot tasks that rely on identifying these distinctive features in the scene. For example, in virtual simulations, graphical models with fine details of geometric properties and photorealistic texture mapping can be created to increase the realism of the objects' appearance for tasks such as object detection and recognition. Equally important is the functional behaviour of an object which, in some robot applications, plays a more dominant role than its appearance. For example, simulations of underwater robot tasks often do not require realistic rendering of water, but they may demand accurate simulations of hydrodynamics.

The real world contains unpredictable variations, and experiments have found that using an appropriate level of noise in simulations can help to produce robot behaviours that are close to the real world [138]. To provide a common simulation environment, there is a need to be able to vary the degree to which the environment emulates the dynamic conditions of the real world. Effects of real world variations may not be desired early in the development since the focus would be on concept validation. However, as the development of the robot system begins to mature, an environment with a certain degree of variations is necessary to evaluate the robustness of the system being developed.

3.2.2 Requirement B: Flexible Scenario Design

HIL simulations are a benefit to the robot development process in terms of cost and safety, providing a flexible approach to robot experimentation. However, they resort to using customised solutions which could be difficult to be reused in other applications. It is pointed out in [172] that a generic and modular robotic HIL simulation architecture can lead to a more reconfigurable and reusable simulation platform, with a key feature being standardised interfaces for communication between software and hardware components. To this end, this research proposes that an essential requirement for improving the reusability of HIL simulations is a standardised environment where simulated components can be seamlessly integrated with the existing experimental setup.

A simulated component can be a part of a robot system. For example, developers may wish to experiment with several alternative ranging sensors for navigation before deciding on the optimal design. In this case, using simulated sensors can be a faster and cheaper way of evaluating concepts than acquiring all corresponding real devices. The requirement is that simulated robot components must be able to interact with the rest of the system so that the robot could correctly function as a whole. The integration should be achieved with minimal or no changes to both real and simulated components of the robot system.

Existing work on HIL simulation such as [86, 256] has indicated the importance of simulating the robot's interaction with the environment. Thus, the standardised environment also includes the robot environment into the loop of the experimental design. The robot environment often plays a major role on the influence of robot behaviour in common robot operations. The robot needs to interact with the environment to obtain an understanding of the world in order to make decisions to complete its task. Thus, objects that are simulated do not necessarily need to be components of a robot system, they can be any object in the environment. For example, wind, lighting, or a physical obstacle can be simulated. It is important that the real robot perceives and interacts with the simulated objects as if they exist in the same coherent space, and likewise, these simulated objects must also be able to interact with the dynamic environment they reside in.

3.2.3 Requirement C: Context Awareness

Not all threats to the safety of an experiment can be foreseen and unexpected events can occur. A safe development environment should support means of improving the the developer's awareness of the robot, its surrounding environment, and potentially dangerous situations. For example, experimentation involving UAVs operating in out-of-sight locations requires providing awareness of the UAV's identity, status, spatial relationship with

other objects, operational threats, and weather. [99]. Common threats include physical contacts, proximity intrusion, human error, abnormal robot behaviour.

Robot, environment, and task relevant data need to be highlighted and communicated to the developers, potentially with the use of visualisations, so that they understand and are constantly aware of the states of the components in the experiment. This also helps them to react immediately to unusual events in order to prevent any disastrous incidents from occurring.

3.3 Features of a Robot Development Environment

Now that the requirements have been listed on what an environment needs to support for enhancing the development process, this section focuses on features for building the enhanced development environment. The first four features are considered to be essential, and each of them is mapped to one or more requirements from the previous section to illustrate which ones they address. The last feature is placed at a lower priority but desirable to have for a better robot programming process.

Generic Interaction Framework: (*Necessary for Requirements A and B)* A formal framework is necessary to provide the basis of creating systems from simulated and real world components. The framework needs to be generic, providing standardised definition of object properties, data interfaces, environment structure, and describing how their interactions should proceed. The framework should be an extension to the traditional HIL simulation paradigm, while facilitating seamless integration between real and simulated components.

The framework may include formal methods for specification, design, and verification of systems consisting of simulated and real world objects. For example, ontologies can be used to classify objects of different reality while illustrating their relationships, and descriptive and specification languages also help to describe behaviours and elicit rules for interactions between real and simulated objects.

To facilitate seamless interaction, the framework may need to provide synchronisation mechanisms that ensure the consistent execution cycle of real and simulated components.

Standardised Communication Interface (*Necessary for Requirements A and B*): In addition to the high level framework for facilitating interaction between components from different dimensions, a communication mechanism is required at the data level. It is necessary to provide standard interfaces or protocols for exchanging messages so that components can communicate with one another regardless whether they are real or simulated.

A standardised component interface is necessary to allow similar components that implement identical interfaces to be easily interchanged. For example, the developed software should be able to operate on simulated robot platforms as well as the actual robot devices.

Network transparency is also a required feature as it helps in managing distributed resources, e.g. robot systems often involve computationally expensive components that are relocated to separate computing units.

Fortunately, many of the features are now common practice in robotic software frameworks such as Player [20].

Simulation Platform (*Necessary for Requirements A, B and C*): A virtual simulation platform is necessary for providing developers models of the robot's sensors, kinematics, dynamics, environment, and the robot's interaction with the environment. The simulation platform should offer a rich database of resources that users could immediately use to create complex scenarios involving different robots and environments. In addition, it should allow users to create and customise their own models.

The simulation platform should be modifiable and extensible, allowing developers to enhance or add new features to create a tool that suits their needs. For example, extending the user interfaces to use different input devices, integrating new visualisation techniques, or adding support for simulation of robot software created from different robot development tools.

The robotic software developed using the simulation platform must also be portable to operate on real robot hardware devices.

The simulation platform should also support interfacing of external data such as sensor inputs necessary for running the simulation. For example, in order have context awareness, it may be necessary provide sensor inputs into the robot development environment through the simulation platform.

Real Time, Intuitive Visualisation (*Necessary for Requirement C*): Visual interfaces with real time visualisation are necessary to provide users instantaneous feedback as the robot operates in the real world so that the users are constantly aware of the statuses of the components in the experiment.

Visualisation techniques for displaying additional, synthetic information in context with the robot environment are necessary in order to help users understand the experiment in an intuitive manner. Robot sensory data and task relevant information

should be transformed into appropriate representations and placed in their corresponding geometric locations in the physical environment. An example to achieve this is through the use of AR visualisations.

Moreover, the use of a real time graphics rendering library is necessary for creating interactive visualisations at high frame rates. The quality of visualisation should also be configurable, e.g. different levels of details or resolutions, depending on the requirements and the available computational resources.

Robotics-Specific Integrated Development Environment: The nature of robot platform, task, and environment differ from general computer applications. Using a robotics-specific IDE makes the robot programming process easier.

The IDE should support writing of formal specifications with syntax checking. A compiler can automatically inform users on the feasibility of the experimental setup to be created based on user specifications of available resources, object properties (e.g. real or simulated), entity relationships, and constraints, and generate warnings on missing or undefined components before implementation takes place.

The IDE should also provide an application-specific language for programming robots. Support for dimensioned data can be built into the language [49]. Spatial, sequential, and abstract data should have its own data type, which can then be mapped to pre-customised visual representations for visualisation within the IDE.

The IDE should provide debugging capabilities tailored to the development of real time systems. Unlike the development of general computer applications, the execution of robot programs cannot be paused, using breakpoints for example, to wait for the developer to debug and analyse its data.

As opposed to the standard monitor, mouse and keyboard input/output devices on desktop computers, robots use sensors and actuators to interact with the environment. Robot programming at the task level can potentially be made easier with a programming environment and a set of tools for describing interactions specific to robotic systems [168], for example, with the use of a robot visual programming language.

3.4 Mixed Reality Simulation

The analysis in this chapter resulted in requirements for the design of a new robot development environment that aims to improve the accuracy, flexibility, and safety of experiments.

This thesis proposes that building a simulation environment based on MR concepts and interfaces can provide a solution to fulfill the three requirements. The transition from

virtual simulations to real world tests has a close resemblance to the Reality-Virtuality continuum proposed by Milgram *et al.* [180, 179] (see Figure 2.4). Virtual simulations can be considered to take place in the virtual environment situated on one end of the continuum, while real world tests are conducted in the real environment situated on the other end of the continuum. By using an appropriate combination of simulated (virtual) and physical (real) components, robot developers are able to design a simulation environment with a good balance of realism and safety as required. The new development method will be referred to as *MR Simulation*.

MR simulation shares many advantages with existing HIL and hybrid simulation approaches. In fact, it stems from the HIL simulation paradigm. In comparison, MR simulation stresses a higher degree of coherency between the existence of real and virtual object, so that in simulations, any virtual entities are able to seamlessly and physically interact with real entities, and vise versa, as if they exist in the same dimension.

MR simulations also support incremental development in one common, coherent environment, i.e. we can develop an algorithm for a robot and incrementally experiment and test the different robot designs in the same experimentation environment, using simulated components as necessary, before the actual hardware/software components such as sensors or an arm are available.

Compared to existing solutions, the essential features described in Section 3.3 help to shape the design of a generic simulation environment that can be applied to the development and testing of a broader range of robot systems and applications. The proposed solution is presented throughout the remainder of this thesis. Chapter 4 presents the first essential feature for achieving MR simulations – a generic MR framework. The framework is inspired by the Reality-Virtuality continuum. It creates MR environments composed of real and virtual objects and facilitates interactions between them.

An implementation of the MR framework, referred to as an MR robot simulator, is demonstrated in Chapter 5. It is shown that the MR robot simulator integrates the second essential feature, standardised communication interfaces, in its design to support communication between heterogeneous components for increased reusability and interoperability of the simulation tool. It also integrates the third essential feature, a highly modifiable and extensible general purpose robot simulator, to provide rich simulation models.

The other advantage of applying MR to robot development is its unique visualisation technique that has been demonstrated to benefit various areas of robotics. Visual presentations of MR simulations are treated as a separate problem from the construction of the simulation environment. Real time MR visualisations that accompany MR simulations form the final essential feature implemented in this research. MR interfaces are designed to provide robot developers intuitive views for understanding the MR simulation environment, and at the same time, increase their situation awareness of the components in

the experiment. The design of the MR interfaces is presented in Chapter 5. The primary technique employed in this work to provide intuitive visualisations is AR, and details can be found in Chapter 6.

4

Mixed Reality Framework

As part of the overall solution to improve the robot development process, Section 3.3 determined that a generic, high level framework is an essential feature for fulfilling requirements concerning the need for a common and flexible simulation environment. To provide this feature, this thesis presents a conceptual framework, referred to as an MR framework [76], that is inspired by concepts found in the literature of MR. This chapter presents the MR framework and demonstrates its application to robot simulations.

The MR framework provides the basis for creating all MR simulations described in this thesis. The core MR framework is designed to be generic, and not limited to modelling of robot simulations. It formalises MR systems in a generic manner that is useful to MR system researchers and designers, and will enable clearer comparisons and more standardised implementations of MR systems. Section 4.1 presents the core MR framework. It first describes concepts for building an MR environment then proposes a novel method for facilitating interactions in the MR environment. Section 4.2 illustrates how the framework is used to create MR simulations for robotics. It is demonstrated that the interaction method enables various entities (real or virtual, a robot or an object in the environment) to physically participate in simulation, thus allowing complex test scenarios to be created for evaluating robot systems. Lastly, Section 4.3 proposes guidelines for designing MR simulations.

4.1 Generic MR Framework

The MR framework constructs an MR environment that facilitates interactions between real and virtual entities. Existing work on forming and combining representations of real and virtual objects in the MR domain serve as the basis for modelling entities in the MR environment [42, 193]. Moreover, the framework can be considered as an extension of general concepts in the field of computer graphics and object-oriented design for modelling a world composed of objects from different dimensions of reality.

4.1.1 MR Entity

The first step in constructing an MR environment is to create appropriate representations of objects that constitute the environment. Recall that an MR environment consists of a mixture of real and virtual objects. Virtual objects are digitised and completely modelled in the computer. In contrast, we may not always have prior knowledge of all real objects in the physical environment. The completeness and accuracy of the model of the real world depends on the amount of information available, either measured *a priori* or collected during operation of the MR system. Note that to achieve real-virtual interactions, representations of real objects in the virtual world are necessary so that their interaction process can be modelled.

To create a model capable of representing an object that exists within the Reality-Virtuality continuum, an abstract *MR entity* is introduced.

In the MR world, an entity can be physical, digital or anywhere in between the two extremes. The MR entity is modelled with an attribute called *level of physical virtualisation* which describes the degree to which an entity's physical characteristics (primarily physical features that are measurable such as shape, colour, mass, and temperature) are virtualised (or digitised) with respect to an object in the real world, e.g. a virtual object that is completely modelled in a computer has the maximum level of physical virtualisation. An entity with an intermediate level of physical virtualisation is possible, as will be described in Section 4.1.2.

In addition, an entity may not be, or does not always fully represent the functional character of the intended object in the actual operation. For example, a photograph or a 3D model of a cow can be used to represent a real cow but it does not have the capability to move and act like a living animal. Consequently it is necessary to model the MR entity with another attribute called *level of functional virtualisation* which describes the degree to which an entity's functional characteristics are simulated with respect to the intended object. Note that it can be difficult to determine the level of functional virtualisation in practice. It may be first necessary to devise a list of functional requirements expected of the entity then decide on the level of functional virtualisation based on the extent to

which they can be implemented in the MR system.

The degree of virtualisation of the entity is determined by the combined result of physical virtualisation and functional virtualisation.

4.1.2 Entity Model

It is common to consider an object in the real world as a high level representation of a combination of several smaller objects. For example, a simple table is composed of a flat surface and four supporting legs. Modelling of composite objects has been thoroughly addressed in the area of computer graphics, such as with the use of scene graphs to store a collection of nodes. Similarly, in the field of MR, hierarchical scene graphs are commonly used to model complex objects, not only for rendering, but also for computing and storing geometric information of all entities [233, 207, 164]. The MR framework builds on the same concept and treats a high level MR entity as a composition of multiple individual entities and other composite entities. The group of entities that form a high level MR entity is referred to as an *entity model*.

In a scene graph representation, an entity model is a tree of nodes. The root node of the entity model acts as a container for grouping entities. Each child node in the tree can contain zero or more objects, and these objects are geometric and/or physical representations of the associated entity. Inspired by the work of [42] that combines physical and virtual objects in an MR application, the proposed scene graph structure also enables an entity model to be composed of a mixture of real and virtual entities. The key difference in comparison to traditional scene graphs is that a child node in the entity model can be real or virtual depending on the level of physical virtualisation of the associated entity. Therefore, an entity model can be composed of a mixture of real and virtual entities and possesses an intermediate level of physical virtualisation, referred to as an *augmented entity*.

The class of reality of an entity model M can be one of the followings:

$$M = \{Real, Virtual, Augmented\}$$

Examples of entity models are shown in Figure 4.1. An entity model is classified as real or virtual if all of its successor nodes belong to the same class, e.g. Figure 4.1a) and 4.1b), and is classified as augmented if it has at least two successor nodes of different classes of reality, e.g. Figure 4.1c) and Figure 4.1d). Note that in the case of Figure 4.1d), the right child of the root node is not an entity model, therefore the node remains classified as real. On the other hand, an entity model can also be composed of other entity models, e.g. Figure 4.1e) and 4.1f).

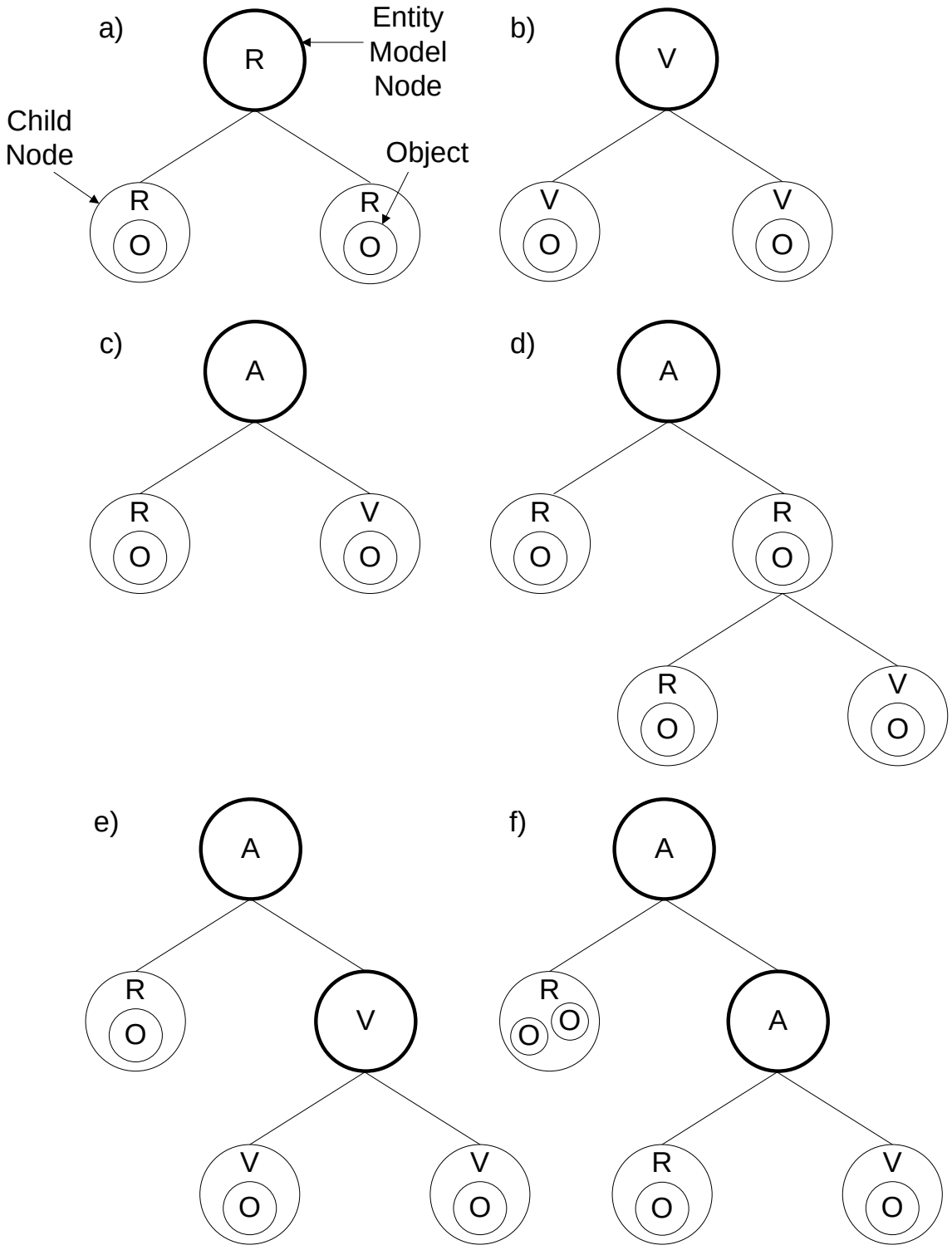


Figure 4.1: Entity models represented using scene graphs. R = Real, V = Virtual, A = Augmented, O = Object.

4.1.3 MR Interaction

While interactions between real objects occur naturally in the real physical world, interactions between real and synthetic (augmented and virtual) objects are more complex and require interventions from the system to model and realise the process. To facilitate interaction between real and synthetic entities, we need to model the process of transforming actions into effects [224]. It is necessary to know how a participating entity reacts to given stimuli, behaves under certain constraints, and also how its response can be generated. This is treated as a behaviour-based interaction problem.

When two agents interact, they exhibit different behaviours. Their behaviours describe the way they act and respond during the interaction. A typical expression of behaviour is shown in Figure 4.2.

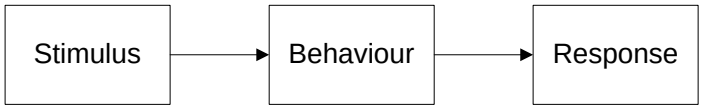


Figure 4.2: An expression of behaviour.

In an MR environment, an agent participating in the interaction can be any single real, augmented, or virtual entity, or an entity model consisting of group of entities, i.e. it can be a robot, a single sensor device, or an environmental object.

Simply put, if we know the interaction is possible between two objects in the real world, we can also try to reproduce the results for interactions with synthetic entities. To achieve this, it is required to model and generate the expected behaviours of the participating agents as if the interaction had happened. Different stages of an interaction are first derived based on the expression in Figure 4.2 then the requirements for a successful interaction between two agents are analysed.

The behaviour expression suggests three stages that must be executed for an interaction to occur: 1) Stimuli Processing, 2) Behaviour Modelling, and 3) Response Generation. The three interaction stages map to three sets of constraints that must be satisfied for each stage to be completed, see Figure 4.3.

Stimuli Processing: As two agents engage in interaction, input stimuli are processed with pre-conditions checking whether all necessary inputs to the modelling process are valid and sufficient information is available to initiate the interaction. Stimuli must be measurable and digitised for the modelling stage. Inputs can be visual, tactile, and audio in order to support most interaction means between humans, computer devices, and the environment.

Behaviour Modelling: Once the input data has been processed and interaction has been confirmed to proceed, the behaviours of the agents are modelled with the given

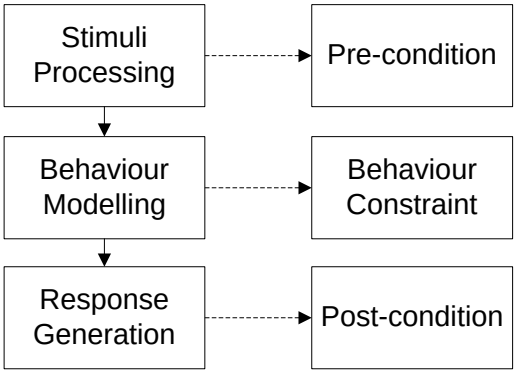


Figure 4.3: Three stages of behaviour-based interaction mapped to three sets of constraints.

inputs. These models describe the way the entity should respond to the inputs, and they can be developed using typical modelling methods including, but not limited to, state machines, rule-based modelling or other AI techniques, statistical approximations, or simple mathematical equations. Behaviour constraints are rules that govern the modelling process. They are typically laws of physics which the agents' behaviours are bounded by or other mathematical constraints that need to be satisfied.

Response Generation: The response generation stage is concerned with putting the modelled behaviour into effect. To achieve a full interaction, the response needs to be executed by propagating the results from the behaviour modelling stage to both the real and the virtual world. When executing the response of an entity, the level of its physical virtualisation indicates the class of reality it belongs to, which influences how an operation can be performed. For example, if an entity is virtual, an operation such as translation or rotation can be performed by simply applying transformations to it on the computer; on the other hand, an augmented node implies that one or more of its successor nodes are real, and custom commands or mechanisms may be necessary to move the object in the real world.

It is important to note that a full interaction may not always be achieved. Instead, a *partial interaction* may occur if the response can not be executed due to resource limitations or safety concerns. For example, it is not always safe to alter the path of a car during its travel after a simulated collision. In this case, the response can be generated by reporting the resulting behaviours to the user using alternative methods, such as textual or graphical output. Post-conditions are used to verify the completeness of the interaction. A full interaction is achieved only if mechanisms are available for executing the response, and the outputs from the executed behaviour match those from the behaviour modelling stage.

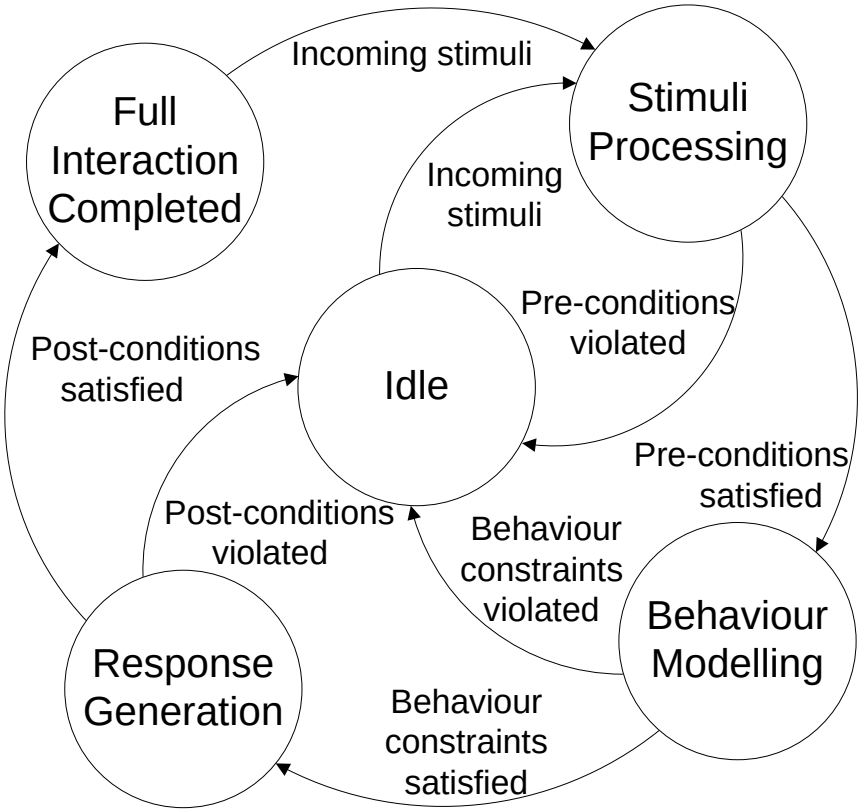


Figure 4.4: A condensed statechart of MR interaction.

Figure 4.2 and Figure 4.3 illustrate the interaction process without a feedback loop. However, feedback to the system can be important for creating accurate interactions. A feedback loop determines whether the output of the response generation stage matches the behaviour modelling stage, and uses this observation to decide how the subsequent input stimuli should be processed to achieve the desired effects of the interaction. Figure 4.4 illustrates the state transitions within an MR interaction that contains a feedback loop. The interaction statechart can be decomposed to the one shown in Figure 4.5. Studies in behaviour-based robotics [32] provided the inspiration for this work, and led the author to create these two statecharts.

4.2 Robot Simulation

The MR framework sets up the foundation of creating a world where real and virtual objects co-exist and interact in real time. When applied to robot simulations, it provides the standardisation required for experiments involving real and virtual components (as discussed in Section 3.2.2).

The idea of MR simulation is heavily based on the Reality-Virtuality continuum, which distinguishes it from many existing HIL simulation or hybrid simulation approaches. MR

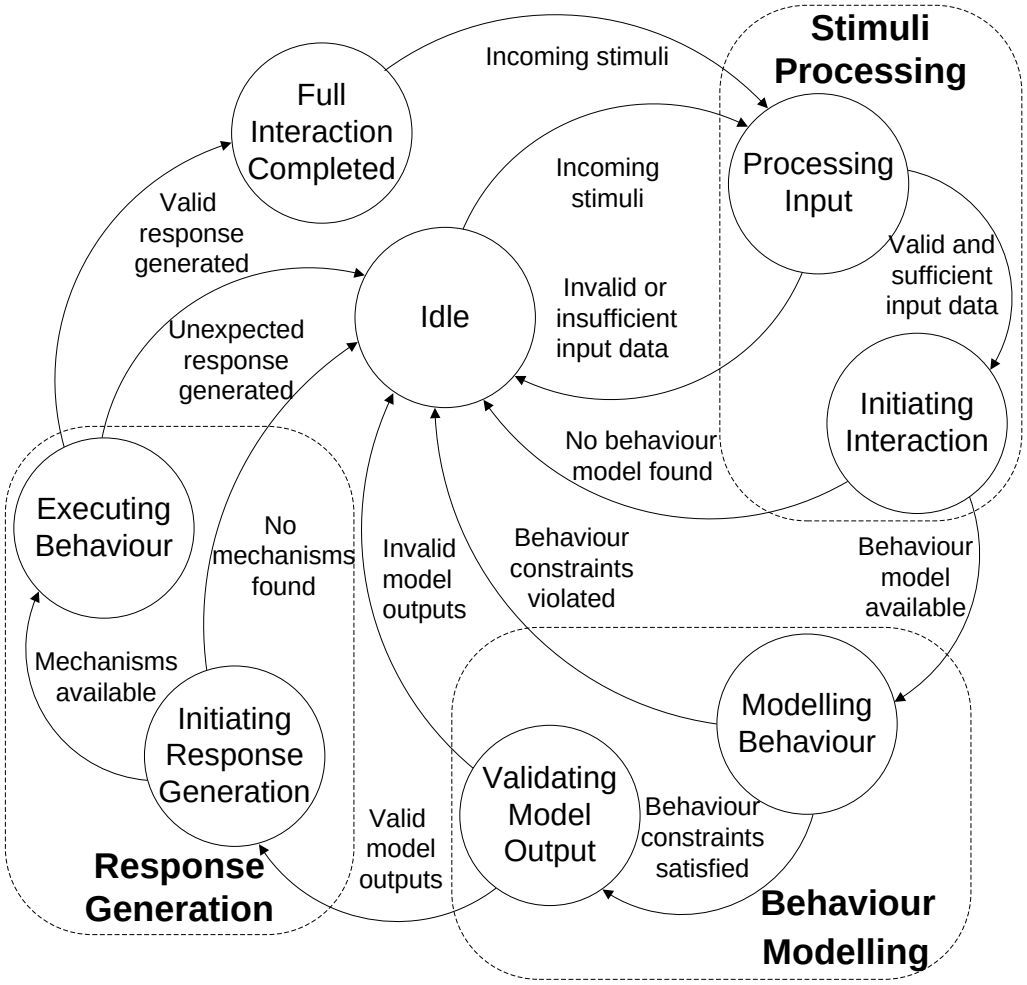


Figure 4.5: An expanded statechart of MR interaction [76].

simulations do not have an explicit classification on what components in the experimental setup need to be physical or simulated (this is a different interpretation of the Reality-Virtuality continuum from the work of Davis *et al.* [86]). The Reality-Virtuality continuum should be smooth and continuous, spanning from the virtual world to the real world. The new approach enables any object in the simulation environment to be real or virtual, and this significantly increases the developer's flexibility in designing experiments.

Consider the example of testing an obstacle avoidance algorithm on a UAV using vision. There would be a significantly high level of risk involved in testing of the UAV system in the real world. We can not always make the assumption that the UAV would avoid the obstacle the first time it is tested. MR simulation enables various configurations of the experimental setup to be designed to allow intermediate tests to be taken before the actual real world test. The experiment can take place in the physical environment with a real obstacle, a tree for instance, while replacing the real UAV with a virtual counterpart. Once testing has been completed with a reasonable confidence of success, the next stage

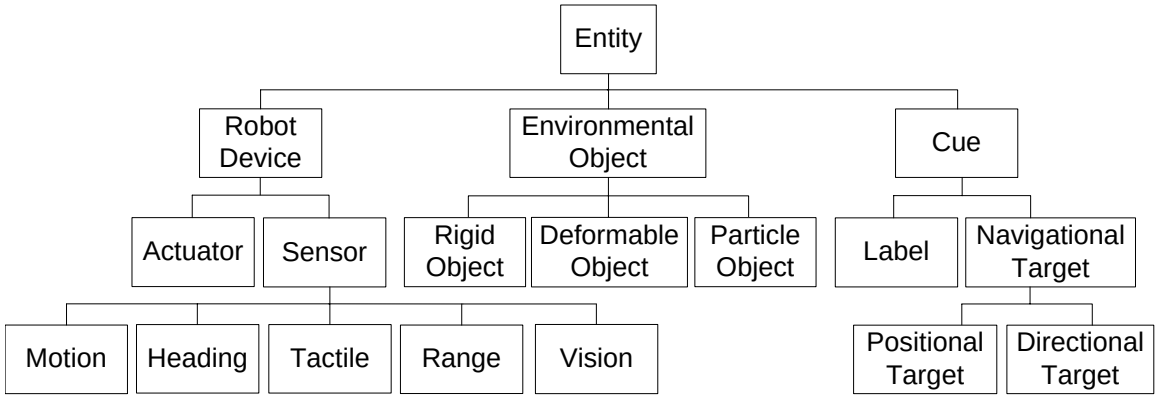


Figure 4.6: An inheritance diagram of an entity in an MR simulation.

may involve testing the real UAV in the physical environment filled with a virtual tree, before finally moving onto completely real world tests.

The remainder of this section illustrates how the MR framework is applied to creating MR environments for robot simulation and facilitating interactions in common robot tasks.

4.2.1 Simulation Environment

The simulation environment in an MR simulation is essentially an MR environment filled with MR entities representing the components in the experiment. The fact that an MR entity can be real, augmented, or virtual gives users the flexibility of creating various test scenarios involving real and simulated components in a similar manner as HIL simulations. Robots, environmental objects (including humans), as well as objects that do not possess a physical form can be extended from the MR entity. An example is shown in Figure 4.6.

An entity model gives the flexibility of choosing what parts of a robot or an environmental object are to be virtualised. Consider a Pioneer robot as an example of an entity model as illustrated in Figure 4.7. Each node in the tree can be real or virtual. For instance, when a Pioneer arm is unavailable, a virtual counterpart can be used instead. However, constraints are also placed between the parent and the child nodes depending on the application and the resources available. For instance, simulating a real gripper on a virtual Pioneer arm may require employing surrogate devices for moving the gripper according to the calculated motions of the virtual robot arm, however, in some cases this may not be possible due to the unavailability of hardware mechanisms for generating such responses. Therefore, an additional constraint could be specified to avoid attaching real nodes to the node of the virtual robot arm.

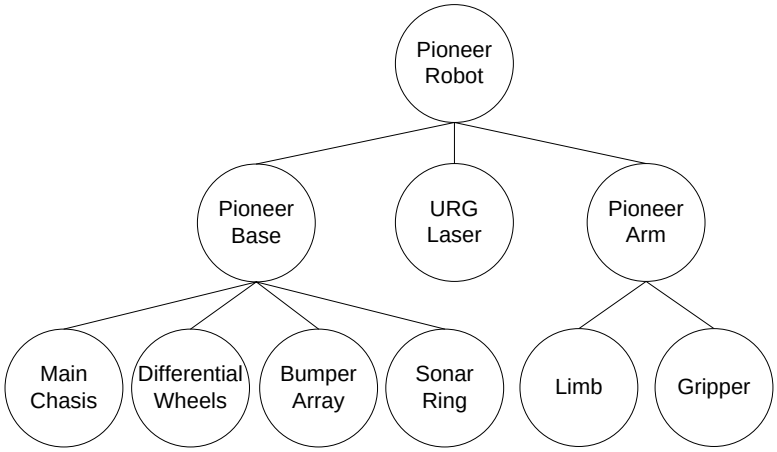


Figure 4.7: An example entity model of a Pioneer robot.

4.2.2 Interaction Types

A robot platform can be considered as a collection of sensors and actuators, both of which are interfaces that the robot uses to interact with the world it inhabits [201]. Sensors are used to collect information about the world, while actuators alter the state of the world. The MR interaction scheme will be demonstrated on how it can be applied to model these two forms of interaction, and specifically, each form of interaction is illustrated with an example of interaction that is common in robot tasks. By understanding how the MR interaction scheme is applied in the two examples, it helps to extend the interaction scheme to capture other robot interactions because the three stages of the interaction that occur are similar and thus can be modelled accordingly.

Sensor based interaction is described by looking at interactions through exteroceptive sensor devices on robots. An example of actuator based interaction is a physical contact or collision that commonly occurs from the robot's actuated motions.

Sensor based Interaction

Sensor based interaction is concerned with interactions between a sensor device and other entities in the environment. Sensor devices are essential components of a robot system for robots to interact with and gain understanding of the environment. In particular, the interest is placed in robot interactions through exteroceptive sensors, both active and passive. This includes range, vision, thermal, sound, and tactile sensors.

Consider an interaction between a real range sensor device and a virtual wall, see Figure 4.8. Pre-conditions check the validity of the input stimuli data before modelling the range sensor behaviour. The necessary data for this interaction include configurations of the range sensor device, positive sensor values, and known position, orientation, and dimensions of the virtual wall in the environment. The behaviour modelling pro-

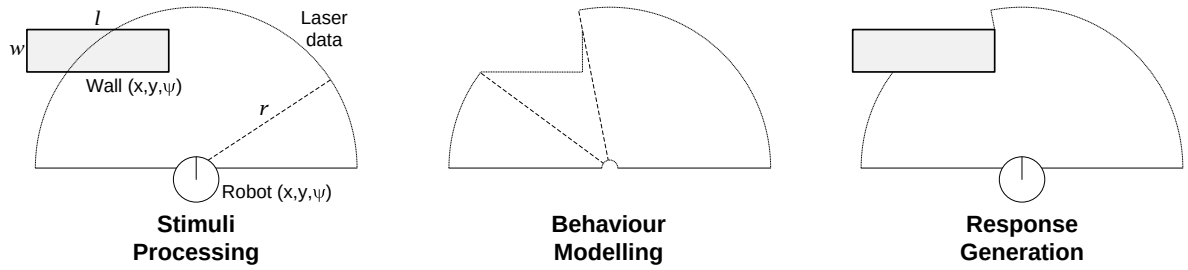


Figure 4.8: An example sensor based interaction between a range sensor device and a wall. Stimuli Processing: collect pose of both entities, and the raw range data and configurations of the range sensor device. Behaviour Modelling: compute expected obstructed volume from raw range data. Response Generation: alter the output of the range sensor device accordingly.

cess requires the range values of the sensor device to be modified according to certain mathematical models in order to reflect a new object in the range output, e.g. a boolean operation to subtract the expected obstructed volume from the original range readings. The resulting range values must also be checked if they are valid, e.g. no negative range values. Once the expected behaviour has been computed, it is necessary to propagate the results to the real world, i.e. making alterations to the real range output. It is assumed that during a sensor based interaction, physical effects of the sensor on the environment are neglectable, and thus behaviour responses from the other party in interaction do not need to be generated. Once the response is executed, post-conditions verify the new range readings against the expected output values computed during the behaviour modelling stage to ensure a full interaction.

Actuator based Interaction

Actuator based interaction is concerned with actuations from robot devices that result in a change in the status of one or both participating entities in the MR world. An example is contact interaction. A robot's actuators drive its motion which often creates physical contacts with other entities in the environment and changes their poses. Contact interaction is common in robotics, e.g. collisions between moving objects, and is often necessary, e.g. in manipulation tasks.

Consider the example of a real robot colliding with a virtual ball, see Figure 4.9. Before initiating the interaction, pre-conditions check whether the geometries and dynamics of the real robot and the virtual ball are known. The behaviour modelling process first performs collision detection by detecting intersections between the entities' geometries. If a collision is detected, rigid body physics may be employed to compute the collision response over time. This can achieved by applying impulses to the two bodies in the collision based on their incoming velocities and corresponding masses. Additionally, it is also important to propagate the effects to all nodes in the entity models involved in an

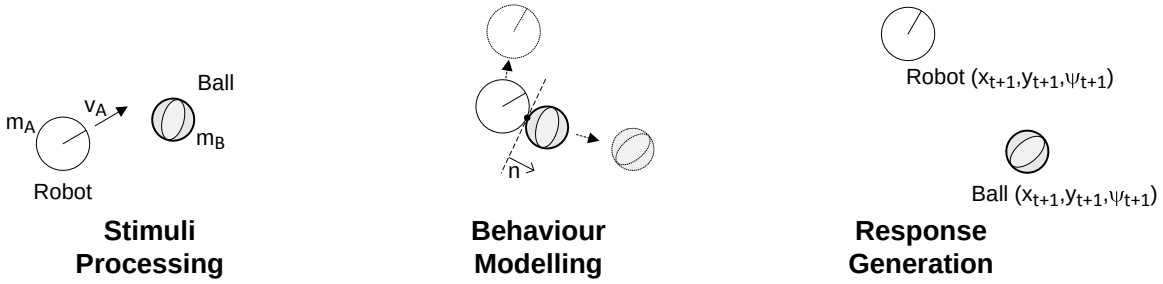


Figure 4.9: An example actuator based interaction (collision) between a robot and a ball. Stimuli Processing: collect properties such as entity bounding boxes, masses and velocities. Behaviour Modelling: identify colliding point, collision normal, time of collision, and calculate impulses to be applied to bodies in collision. Response Generation: move the two entities accordingly over time.

actuator based interaction. For instance, if the virtual ball collides with a virtual robot arm mounted on the real robot, then this also affects the motion of the real robot as it is the parent to the arm in the entity model. Lastly, to execute the response, checks are needed to identify whether mechanisms are available for safely interrupting the motion of the real robot or deflecting its direction of travel to result in a new motion that resembles the outcome of the behaviour model. The virtual ball can simply be animated to move according to the modelled behaviour over time. Post-conditions are optional to verify and ensure that the two entities move in a similar manner to the output of the physics model for a full interaction.

Extending to Other Interactions

Other interactions in tasks required of robots can also be divided into sensor based and actuator based interactions, and modelled in a similar way to the examples described above. For example, an industrial robot equipped with a paint gun that sprays surfaces of automobiles can be considered as an actuator based interaction because the robot alters the state of another object in the environment.

To model high level interactions, sensors and actuators do not necessarily need to refer to the mechanical devices on robots. For example, bluetooth devices equipped on robots that exchange information for communication can be considered as a combination of sensor based interaction (receiving data) and actuator based interaction (transmitting data).

Note that the interaction examples presented are not limited to robotics. Sensors are increasingly used in technologies around us, such as in mobile phones which are becoming a popular platform to deploy MR systems, while contacts/collisions between real and virtual objects are important in MR tangible interaction studies.

Abstractly speaking, all entities can also be associated with sensing and actuating

capabilities, regardless of whether the entity is a robot device or an object in the environment. For example, a dog often senses with its eyes and nose (sensor) and is capable of moving in the environment by running with its legs (actuator). This makes it possible to extend the two forms of interaction to other MR applications.

4.2.3 MR Interaction Issues

Creating MR interactions may not be easy, and the interaction outcome may not always be realistic or correct. There are key issues that need to be considered for the two interaction types discussed in the previous section.

It is often difficult and time-consuming to collect information of objects that exist in the real physical world, and without sufficient information, we can not construct their virtual representations and model their behaviour. This can be the main cause of failure for creating interactions. The problem becomes apparent when simulating virtual sensors. In order to facilitate sensor based interaction between a virtual sensor and real entities, a model of the physical dynamic environment is necessary, and this information may require considerable effort to collect and can also be expensive to maintain during the operation of the system. For example, to create realistic coverage of a virtual range sensor, we need to know the dimensions and locations of real objects in the environment at any given time. To create a virtual temperature sensor, we may have to make predictions of the temperature of the target environment over time based on historic data. The same applies to contact interaction. Stimuli exhibited by real entities, such as external forces, acting on another entity in interaction can be difficult to measure. In principle, the greater the extent of world knowledge that we have of objects in the physical environment, the more complete the model of entities we can construct in simulation, which in turn influences the outcome of an interaction.

Achieving an actuator based interaction between entities can be more complex than achieving a sensor based interaction. The difficulty mainly lies in the behaviour generation stage. A contact/collision interaction suggests physically altering the behaviour of entities to generate the expected effects in response to physical actions exerted by the other entity in the interaction. However, without proper mechanisms for executing the responses, it can lead to undesired consequences. In the collision example, forcing a real object to physically change its pose or motion can produce unnatural results and may also cause serious damages. On the other hand, an example of a safe full contact interaction is demonstrated in the work done by Takahashi *et al.* [256]. In their hybrid simulation, the use of a nine DOF motion table safely generates the resulting motions from a collision between real robot manipulators and a target object. Nevertheless, hardware mechanisms for executing responses can be expensive, thus, a partial contact interaction is sometimes preferred depending on the requirements.

4.3 MR Simulation Design Guidelines

After applying and evaluating the MR framework in different industrial and research projects (to be described in Chapter 7), there are a number of lessons learnt which have been formulated into guidelines to help users create MR simulations that best suit their application and requirements. First, safety should be a priority in the experimentation of high risk robot operations. As MR simulations involve real world components in the simulation loop, it is important to ensure the safety of these components and the surrounding physical environment. The first guideline helps users to determine the appropriate use of real and virtual objects for designing safer MR simulations. On the other hand, it has been argued that simulations do not always need to be highly realistic to be useful [138]. Rather than focusing entirely on simulation realism, the second guideline proposes to design simulations based on the requirements of the task. Finally, given that MR simulation is also a HIL based simulation method, it is important to create simulations that overcome existing HIL simulation limitations that were identified in Section 3.1.2. The third guideline helps users to understand the appropriate use of MR simulation to maximise benefits.

Level of Virtualisation vs. Safety: The safety of the experiment is closely dependent on the choice of appropriate representations of entities in simulation. For example, a simulation that investigates the performance of a helicopter robot in navigational tasks may require more entities to be virtualised to avoid severe consequences from collisions compared to one that investigates the performance of indoor ground robots. However, care should be taken when deciding what objects to be virtualised. Landing of a real helicopter robot on a completely virtual platform could be more dangerous than any other means of simulation, since it suggests terminating the helicopter system in mid air. When designing MR simulations, there is a benefit from using a knowledge base or a reasoning component for dynamic selection of representations of a particular entity, especially in non-trivial applications. For example, if a user gives a potentially dangerous command to a physical robot, the appropriate system response is to switch to a non-dangerous virtual representation before executing the command and facilitate only a partial interaction.

Realism vs. Task Requirements: The level of virtualisation does not solely dictate the realism of the simulation and the reliability of the results produced. The degree of realism of an MR simulation is also influenced by the quality and correctness of the entity representations and the behaviour models. The time and effort spent on modelling for constructing scenarios should consider the nature of the robot task. In applications where consequences of malfunctions are too severe, a virtual environment with high fidelity simulation models should be used, whereas for other

applications involving a complex but low risk environment, modelling is unnecessarily costly and so the overall level of virtualisation can be reduced [272, 267].

The degree of realism of an MR simulation is also influenced by the completion rate of MR interactions. In order to achieve realistic simulation results, full MR interactions should be created unless safety becomes an issue or hardware mechanisms for executing responses are unavailable. In cases where partial interactions are preferred, the final outcome of the interaction should also be communicated to the users so that they are able to see and understand the resulting behaviour of the system being tested in simulation, e.g. subsequent robot movements can be visualised over the physical robot [93].

MR Simulation vs. Existing Methods: To maximise the benefit of using MR simulations, the MR simulator must be designed to be low cost and reusable so that there are evident advantages over using similar tools such as existing HIL simulators. The MR simulation environment should not require expensive equipment to set up. The design of the MR simulator should incorporate AR or MR techniques that can scale well to suit different robot platforms, frameworks, as well as various indoor and outdoor environments where robots commonly operate in.

Nevertheless, it is important to keep in mind that while MR simulations offer unique benefits for the development of complex systems, the advantages inherent in using existing simulation methods should not be neglected. When an MR simulation includes real components in the simulation loop, it becomes a real time simulation system and loses the ability to freeze and playback scenarios, or speed up and slow down phenomenon as offered in common offline simulation tools [41]. The sense of physical presence is also weakened in comparison to real world experiments when using instruments, interfaces, or visual displays to realise user interaction [47]. MR simulations should be used as a complementary step in the development cycle, and not as a replacement of all existing methods.

The first two guidelines emphasize that designing MR simulations require a good balance between the degree of virtualisation and realism depending on the application and requirements. Robot applications with higher safety, cost, and time requirements may typically demand simulations with greater realism and would benefit from a higher degree of virtualisation. For example, the analysis in Chapter 3 identified strict safety requirements in the development of aerial robotics, which determined the need for expensive or dangerous components to be replaced with simulated substitutes. However, in some cases high requirements do not necessarily impose the need for great realism. For example, in military applications, the development of a complex and expensive technology (missile firing robot) could have high cost and safety requirements but validations of

the technology could be carried out without using a high fidelity physics simulation (the objective is achieved as long as the missile hits somewhere near the target). In practice, requirements may differ between projects as well as the stage of the development, thus the degree of virtualisation and realism should be adjusted accordingly to maximise the benefit of using MR simulations.

4.4 Discussions

The generic MR framework presented is an essential feature for standardising the way to construct an environment composed of objects from different dimensions of reality. In its application to robot simulation, the MR framework effectively addresses Requirement B listed in Section 3.2.2 by providing the developers the flexibility of virtualising components in an experiment for cost and safety reasons. The MR framework also allows the level of physical and functional virtualisation of a simulation environment to be varied to suit the requirements of different applications and development stages, providing a common simulation environment as described in Requirement A in Section 3.2.2.

The behaviour-based interaction scheme formalises the interaction process between entities in the MR environment, and this enables real and virtual robots, sensors devices, and environmental objects to physically participate in MR simulations. The interaction scheme has been applied to two forms of interactions, sensor based and actuator based interaction, and illustrated with an example each. While the examples do not represent all the possible interactions that a robot is capable of, the process of interaction that is carried out is believed to be similar to those of other interactions found in common robot tasks. A working implementation of the MR framework and the interaction scheme will be demonstrated in Chapter 5.

The diverse field of MR requires any MR frameworks to be general and extendable to suit different operating contexts, and so there is still a need to investigate whether the framework supports other forms of interaction found in non-robotics MR applications. For example, many AR/MR applications support physical interaction with virtual information using tangible interfaces, as well as social interactions through natural modalities such as speech and gesture. Extensions of the MR framework to applications outside robotics are considered as future work.

5

Mixed Reality Robot Simulator

It was shown in the previous chapter that the generic MR framework can be effectively extended to model robot simulations and capture different forms of interaction. This chapter describes an implementation of this extension of the MR framework, and refers to the implemented system as an MR robot simulator [74]. The MR robot simulator is a key part of this research and will be heavily used later in the evaluation process to help validate the concepts proposed in this thesis.

The MR framework presented in the previous chapter forms the first essential feature, a Generic Interaction Framework, identified in Section 3.3 for building an improved robot development environment. In this chapter, it will be shown that the MR robot simulator integrates an implementation of this essential feature, named MRSim. The simulator also supports the three remaining essential features. The second feature is a Simulation Platform. The MR robot simulator is built on an existing virtual robot simulator, namely Gazebo [155], for providing sensor, actuator, and environment models. The third feature is a set of Standardised Communication Interfaces. The design of the system places a strong emphasis on software reusability. It promotes standardisation by integrating widely used robot software frameworks for communication with different robot systems. The fourth feature is the support of the MR robot simulator for providing Real Time, Intuitive Visualisations. The simulator's interfaces incorporate real time visualisations designed to help users interpret information from the MR environment where simulations take place, and improve the users' awareness of the simulation.

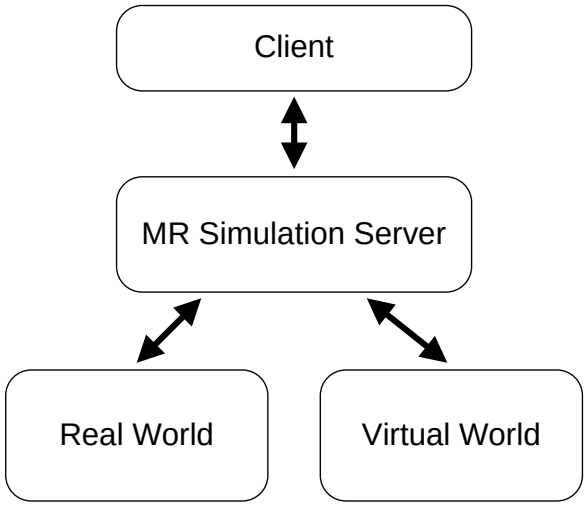


Figure 5.1: Overview of elements in MR robot simulation.

An overview of the MR robot simulator is given in Section 5.1, describing the necessary elements for creating an MR simulation system. Section 5.2 first introduces the virtual robot simulation platform that the MR robot simulator is built on, then Section 5.3 describes in detail the system architecture of the MR robot simulator. Section 5.4 describes MR interfaces designed to accompany the MR robot simulator, and finally, Section 5.5 presents a preliminary functional validation of the system.

5.1 Overview

An MR robot simulation can be divided into four elements: 1) Client, 2) MR Simulation Server, 3) Virtual World, and 4) Real World. Figure 5.1 shows the elements and the data flow between them.

- The **client** is the robot program being developed and tested in simulation.
- The **MR simulation server** is the primary component responsible for constructing the MR environment and facilitating interactions between real and virtual entities. It monitors the states of the real and the virtual world and seamlessly fuses data collected from the two worlds.
- The **real world** is essentially the physical environment where experimentation takes place. A model of the real world can be formed from prior measurements taken before the simulation begins, as well as data sensed by real robot devices during the operation of the system.
- The **virtual world** is the virtual environment created by a virtual robot simulator. It provides robot sensors, actuators, and environment simulation models.

As can be seen from Figure 5.1, the client robot program now exchanges messages with the MR simulation server, instead of exchanging directly with real or simulated devices. One advantage of using an intermediate layer is that it provides an abstraction of the underlying devices. It merges the real and virtual worlds in a way that is transparent to the client who sees real and virtual entities as part of the same coherent world. This allows the client program use the MR robot simulator without making modifications to its software code. Another advantage is that the MR simulation server will gain a more direct control to these real and simulated components, which will be necessary for executing real-virtual interactions (more details on creating interactions will be given in Section 5.3.1 and Section 5.3.2).

An MR Simulation toolkit, MRSim, has been implemented, which plays the role of the MR simulation server. MRSim does not have its own graphical user interfaces which the users can interact with. It is composed of a logic and a data layer for managing how the mixing of the two worlds should proceed, and what data are necessary for the process. This maximises the reusability of the software, making it independent of any client robot programs or simulation tools.

MRSim alone does not provide MR robot simulations. To construct a fully featured MR robot simulator, MRSim is integrated with an existing virtual robot simulation tool. Before presenting MRSim, it is first necessary to understand the design of the selected virtual simulation platform and how it creates robot simulations.

5.2 Simulation Platform

Instead of taking the time-consuming process of building the MR robot simulator from the ground up, it is desirable to exploit and extend existing robot simulation tools. This section first considers a number of criteria for selecting the base robot simulation tool to build on, then describes the components of the selected simulator. An overview of the original simulation framework is also introduced to provide an understanding of how client robot programs communicate with the selected robot simulator. Note that this understanding is important as later in Section 5.3 it helps to identify the differences and the modifications that have been made to the original simulation framework for creating MR simulations.

During this research, the selected virtual robot simulator has also been extended to support simulations of robot programs created from a new and different robot software framework called OpenRTM, see Section 5.2.3. The work is part of the overall effort to enable the final MR robot simulator to work with different robot systems, which is important for the evaluation process of this research.

5.2.1 Extending Existing Robot Simulator

An extensive survey on existing robot simulators was conducted. Details are given in Appendix B. Each simulator was thoroughly analysed and evaluated based on its documentation, programming support, extendibility, range of supported sensor and actuator models, portability of the developed software, as well as specific requirements for integrating MR technology.

Preferences are given to robot simulators that satisfy important criteria for providing the Simulation Platform essential feature outlined in Section 3.3. The simulation platform must be modifiable and extensible, and support integration of new technologies. For this reason, open source projects with wide community support are preferred. The simulation platform must also offer a rich database of simulation models, thus a general purpose robot simulator capable of supporting a wide variety of sensors and robot platforms is also preferred. Most importantly, the software developed using the simulation platform must be portable to real robots. Additionally, preferences are also given to robot simulators with customisable visualisation tools or real time graphics rendering support that help to provide the Real Time, Intuitive Visualisation essential feature.

After reviewing the list of robot simulators, three candidates were found to meet these criteria, which are Gazebo [20], OpenRAVE [92], and USARSim [61]. The table presented in Appendix B shows that the three simulators offer comparable functionalities, so to narrow down the selection, a qualitative comparison was performed and summarised in Table 5.1[1].

Gazebo was found to be a more suitable choice for this research. Gazebo offers a more general purpose simulation environment, while the focus of OpenRAVE has been centred on motion planning algorithms for robot manipulation tasks. Gazebo is open source, modular, highly modifiable, and has independent rendering and physics subsystems which facilitate the integration of MR technology. On the other hand, an analysis of USARSim's system architecture identified that modifications to the physics and rendering modules of USARSim's underlying game engine could be difficult which would potentially constrain the integration and development of MR simulations and visualisations. Finally, the observed high development activity in Gazebo has a positive contribution to the selection of this robot simulator.

Gazebo

Gazebo is an open source 3D robot simulation tool widely supported and used by many research organisations. Gazebo is part of the Player Project [20] for the development of

[1]The qualitative comparison is based on the status of these robot simulators at the time the analysis was performed. However, all three simulators have evolved since then and have received significant improvements.

Table 5.1: Qualitative Comparison of Gazebo [20], OpenRAVE [92], and USARSim [61]

	Gazebo	**OpenRAVE**	**USARSim**
Simulation Focus	General	Motion Planning	General
Integration Effort	Low	Low	High
Development Activity	High	Medium	Medium

robotic and sensor applications and it adopts a similar architecture design as Player. It features a publisher/subscriber, a form of client/server, model of communication. The design of Gazebo enables the simulation platform to work seamlessly with Player robot systems. Nevertheless, the operation of Gazebo is independent of Player's framework, which greatly increases the reusability of the simulation tool.

Gazebo uses open source rendering and physics libraries that also have wide community support. The user can select the Open Dynamics Engine (ODE) [245] or the Bullet Physics Library [4] for rigid body physics simulation, while OGRE3D [19] provides high quality 3D graphics rendering.

Each simulation entity, e.g. a robot model, in Gazebo can be associated with zero or more *controllers* that handle commands for controlling the associated entity, and generate data describing the states of the entity. Gazebo enables clients to access its data through the use of shared memory (SHM). Data generated by controllers are published through data interfaces, known as `Ifaces`, to SHM where it can be read also using `Ifaces` by other processes (see Figure 5.2 for an example). This enables inter-process communication between Gazebo and the client programs independent of their platforms, frameworks, and programming languages used. The C++ library `libgazebo` is provided by Gazebo for accessing the simulation data stored in SHM.

5.2.2 Player/Gazebo Simulation

Gazebo is designed to be highly compatible with it's parent robot software framework, Player [20]. Player is a socket based device server that provides abstraction to robot hardware devices. It enables distributed access to robot sensors and actuators and allows concurrent connections from multiple client programs.

A Player server running on the robot platform is associated with *drivers* that directly access the hardware layer. Each driver can be bound to a number of *interfaces* to provide a *device* which Player clients can subscribe to using a *device proxy*. For example, the `p2os` driver distributed by the Player library provides gripper, position, and sonar devices for Player clients to exchange messages with a Pioneer robot.

In the Player/Gazebo simulation framework, Player drivers provide the abstraction

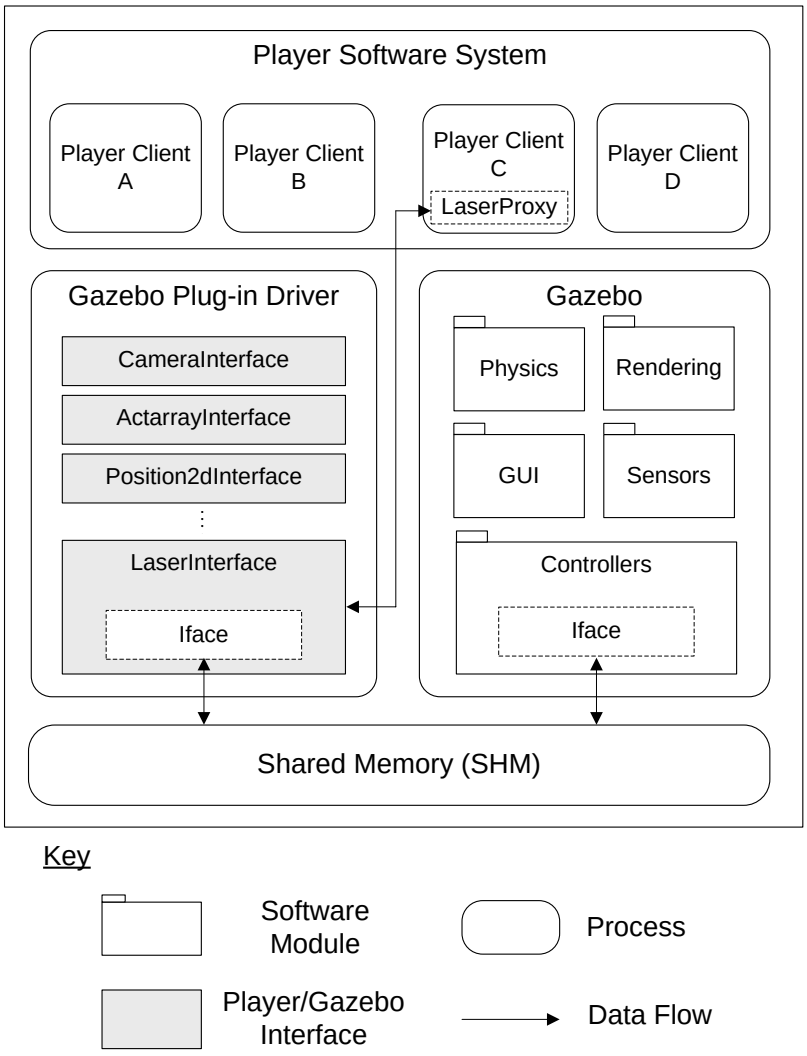


Figure 5.2: Data flow between a Player client, the Gazebo plug-in driver (a Player driver) and Gazebo. As an example, the Player client uses a laser proxy to subscribe to a laser device in Gazebo.

necessary to replace robot hardware devices with Gazebo's simulation entities. The Gazebo distribution includes a Player driver, named *Gazebo plug-in driver*, that acts as a medium for communication between Player clients with the simulated devices in Gazebo. The driver itself contains a set of interfaces which are responsible for a) processing messages sent to them by the Player clients into Gazebo compatible data formats before writing them to the SHM using `Ifaces`, and b) reading data from the SHM using `Ifaces` then reformatting and publishing them to the subscribing Player clients. Figure 5.2 shows the Player/Gazebo simulation framework and the data flow between the different components.

5.2.3 OpenRTM/Gazebo Simulation

Part of this research has extended the Gazebo simulation framework to support simulations of OpenRTM robot systems in addition to Player clients. The work is an effort to enable the final MR robot simulator to be more reusable to support simulation of different robot systems. The approach taken in the research is to integrate robotic software frameworks that implement international standards of robot technology. OpenRTM is one such framework which implements the Robotic Technology (RT) Component standard [10] developed by the Object Management Group [21]. Similar to Player, OpenRTM provides standardised communication interfaces which are considered to be an essential feature. By interfacing OpenRTM with Gazebo, it enables both the client program and the MR robot simulator to communicate with real OpenRTM devices and simulated Gazebo devices through a set of common interfaces. In the Player/Gazebo simulation framework described in Section 5.2.2, this interface layer was shown to be provided by the Gazebo plug-in driver.

The remainder of the section will describe in detail the new OpenRTM/Gazebo simulation framework [72] and show results from simulation examples.

OpenRTM

OpenRTM [185] is a robot software framework that has been widely used in Japan for robot development. OpenRTM-aist [185] is the open source implementation distributed by the National Institute of Advanced Industrial Science and Technology (AIST). OpenRTM-aist is based on the CORBA (Common Object Request Broker Architecture) distributed object middleware. The goal of the framework has been to promote standardisation, reusability, and interoperability of robot components.

In comparison to Player which uses a client/server model, OpenRTM is a component-based framework. RT Components form the basis of an OpenRTM-aist system. Components communicate via two types of ports. Data ports provide one-way, asynchronous communication. An *InPort* subscribes to an *OutPort* which publishes data. Service ports are used for request-response communication. Component introspection is supported, in keeping with the standard, which allows a range of component metadata to be discovered at run time, such as component capabilities and available ports. Internally, components execute a state machine. States include *activated*, *deactivated*, *execute*, and *error*.

OpenRTM-aist provides two graphical tools for developing robot systems. RTCBuilder is an Eclipse plug-in for generating the skeleton code of RT Components based on user-provided specifications. RTSystemEditor is another Eclipse plug-in, which is used for linking data and service ports between multiple RT Components to create a complete robot system.

Interfacing OpenRTM with Gazebo

To interface OpenRTM-aist with Gazebo, a set of interface components has been developed. The set of interface components plays a similar role as the Gazebo plug-in driver in the original simulation framework. It abstracts the simulation platform and enables other software RT components to exchange data with Gazebo without modifications to the underlying software code. The software RT components that are tested in simulation will be referred to as client RT components.

For each type of simulation device in Gazebo, such as a sensor device (a laser rangefinder or a camera, for example), a corresponding RT Component is created; this will be referred as an interface RT Component. The interface RT Components are responsible for exchanging data between controllers in Gazebo and the client RT Components. An interface RT Component can have zero or more InPorts that accept commands sent from client RT Components. There can be one or more OutPorts depending on the types of data each simulation device is capable of generating. The simulation framework is shown in Figure 5.3; the diagram shows a client RT Component reading data generated by a Gazebo simulation device through an OpenRTM simulation interface layer that contains the set of interface RT Components.

The life cycle of an interface RT Component is described as follows:

- Initialisation – The interface RT Component is initialised with a default Gazebo server ID and simulation device ID that it will subscribe to. These values can be configured using RTSystemEditor.

- Activation – The interface RT Component subscribes to one simulation device using `libgazebo`.

- Execution – The interface RT Component continuously monitors the SHM for new data published by the subscribed simulation device. If new data are available, they are processed into data formats consistent with the RT Component's interface guidelines [185] before being forwarded to the client RT Components through its OutPorts. Similarly, commands sent by client RT Components are processed into Gazebo compatible data formats before writing them to the SHM. Service requests from client RT Components are handled in a similar manner.

- Deactivation – The interface RT Component unsubscribes to the simulation device.

RTSystemEditor is used to create connections between the interface RT Components and the client RT Components. In cases where there are multiple simulation devices of the same type in Gazebo, such as two laser sensors, instances of an interface RT Component can be created and configured in RTSystemEditor to map to the corresponding device

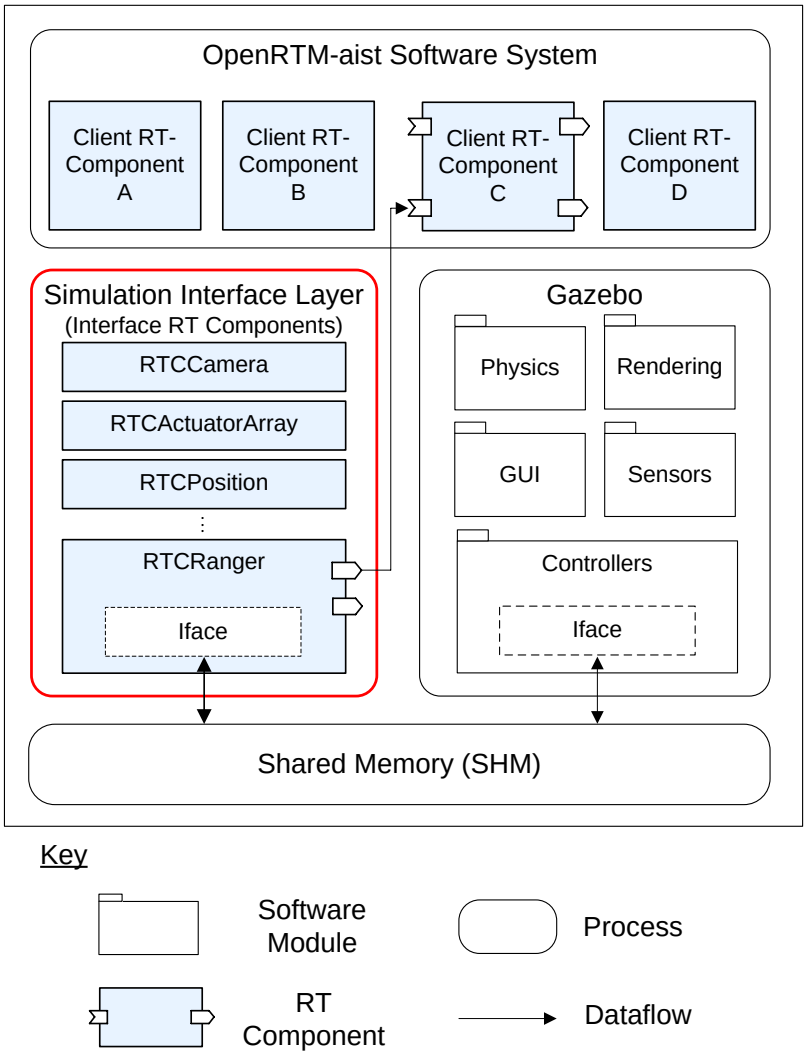


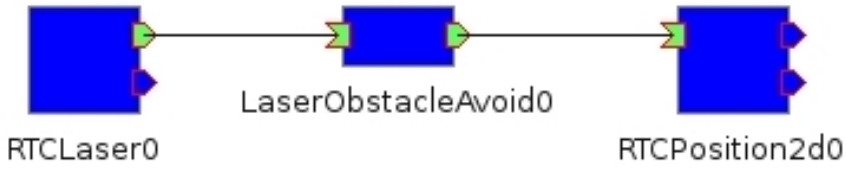
Figure 5.3: Data flow between client RT Components, interface RT Components, and Gazebo. As an example, the diagram shows a rangefinder interface RT Component forwarding simulated range data read from SHM through its OutPort to the InPort of a client RT Component. The InPorts and OutPorts of other RT Components are not shown for clarity.

based on its unique ID. The configuration of RT Components can be saved as XML files which can later be imported back to RTSystemEditor to restart the simulation.

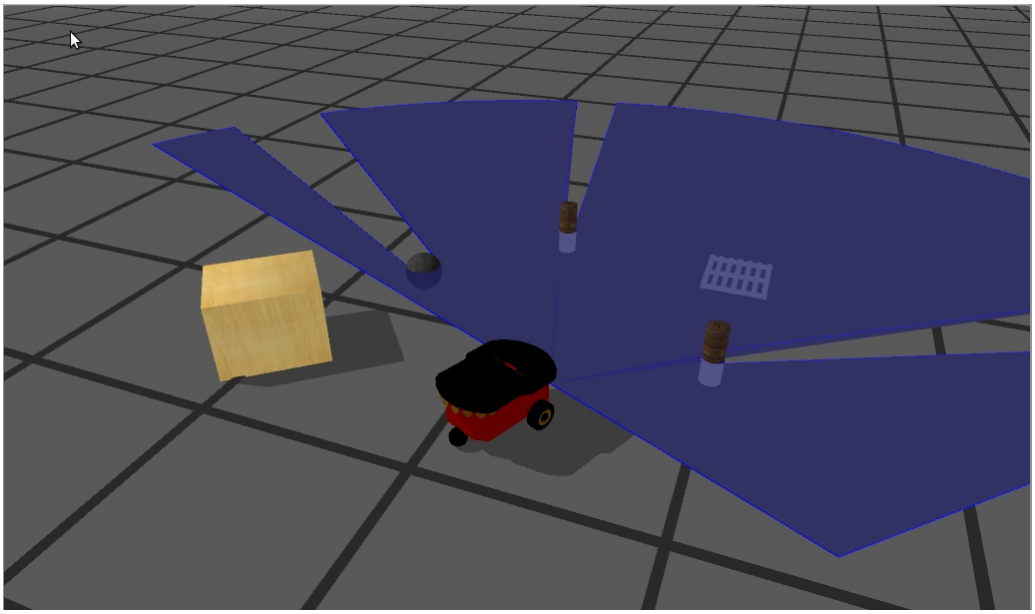
OpenRTM/Gazebo Simulation Examples

A number of interface RT Components have been implemented in C++ for OpenRTM-aist-1.0.0. Three simulations of OpenRTM-aist robot systems in Gazebo will be demonstrated.

The first example is a simulation of an obstacle avoidance algorithm. An obstacle avoidance RT Component has been created; it is a client RT Component to be tested



(a)



(b)

Figure 5.4: (a) The RTSystemEditor showing the connections between the laser interface RT Component (RTCLaser), the obstacle avoidance RT Component (LaserObstacleAvoid), and the position2d interface RT Component (RTCPosition2d). (b) A screenshot showing the Pioneer2DX robot avoiding various obstacles in Gazebo

in Gazebo simulation.[2] The obstacle avoidance algorithm controls a robot to navigate towards the open space while avoiding obstacles detected by the laser rangefinder. Two interface RT Components are used. The RT components are connected using RTSystemEditor as shown in Figure 5.4(a). The laser interface RT Component reads range data from SHM then forwards the data through its OutPort to the InPort of the obstacle avoidance RT Component. The obstacle avoidance RT Component uses the range data to determine the appropriate velocity and turn rate that the robot should move at. The commands are then sent to the position2d interface RT Component which is responsible for forwarding them over SHM to the simulated robot in Gazebo (the same algorithm is used later in a preliminary validation of the MR robot simulator and more details are given in Section 5.5.1). In simulation, the Pioneer2DX robot model provided by Gazebo is

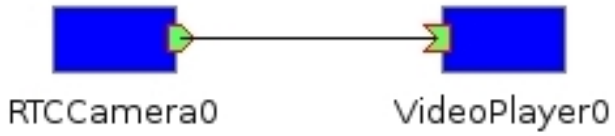
[2]The obstacle avoidance algorithm has been ported from the Player Project.

used. The robot is equipped with a simulated Hokuyo URG laser rangefinder which generates laser data (read by the laser interface RT Component), and has simulated differential drive capability (controlled by the position2d interface RT Component). Figure 5.4(b) shows the Gazebo simulation.

The second example shows a video player RT Component simulated in Gazebo. The video player displays image frames captured by a camera device. A camera interface RT Component reads image data from SHM then sends the data through its OutPort to the InPort of the video player RT Component. In the video player RT Component, the video rendering window is created using GTK and runs in a separate thread from the main execution cycle of the RT Component. At runtime, it monitors the RT Component InPort data buffer for new incoming image data, reads the latest image data into a pixel buffer, then renders the content of the pixel buffer onto the display window. In simulation, a camera sensor is mounted on an unmanned helicopter model for monitoring a cow moving in an agricultural environment. Figure 5.5(a) shows the components in the RTSystemEditor, and Figure 5.5(b) shows the Gazebo simulation and the video player RT Component displaying the view captured by the onboard camera.

The last example demonstrates a simulation involving an OpenRTM-aist robot system and a Player client robot system cooperating on a task in the same Gazebo simulation instance. This simulation uses both the OpenRTM simulation interface layer and the Gazebo plug-in driver to interface these client programs with Gazebo. The example shows two Pioneer robots in a maze-like environment navigating in a leader-follower formation. The obstacle avoidance RT Component is used to control the lead robot to explore the environment by sending commands over the OpenRTM simulation interface layer, while the Player client program controls the second robot to follow closely behind by sending commands over the Gazebo plug-in driver. The object following algorithm used in the Player client program operates by 1) segmenting the robot's laser range data to identify all possible cylindrical objects with the specified diameter, 2) identifying the closest cylindrical object by calculating the distance from the robot to the centre of each cylindrical object (target distance), 3) calculating the angle of the closest cylindrical object relative to the robot (target angle), then 4) sending to the robot a linear velocity command proportional to the target distance and an angular velocity command proportional to the target angle. Figure 5.6 shows a screenshot of the Gazebo simulation. The result shows the flexibility of Gazebo to act as a single, shared simulation environment for testing robot systems created using different robot software frameworks. This increases the reusability of the simulation tool, and provides interoperability in simulation between heterogeneous robot systems.

The examples in this section demonstrate how interface RT components are used to facilitate communication between client programs and simulation devices. The case



(a)

(b)

Figure 5.5: (a) The RTSystemEditor showing the connection between the camera interface RT Component (RTCCamera) and the video player RT Component (VideoPlayer). (b) A screenshot showing the agricultural simulation environment consisting of a helicopter robot and a cow. The robot is equipped with a downward looking camera that captures images displayed in the window shown on the left.

study evaluation in Section 7.2 will describe a more complicated example of using RT components in MR simulation.

5.3 MR Robot Simulation Framework

The previous section introduced the base simulation platform, Gazebo, and illustrated examples of virtual simulations of robot systems in Gazebo. The section describes how MRSim is integrated into Gazebo to provide MR simulations. It first describes the MR simulation system architecture, showing the modifications to the original simulation framework. The core components of MRSim are then introduced and a number of implemented *entities* and *behaviours* provided in the MRSim library are described.

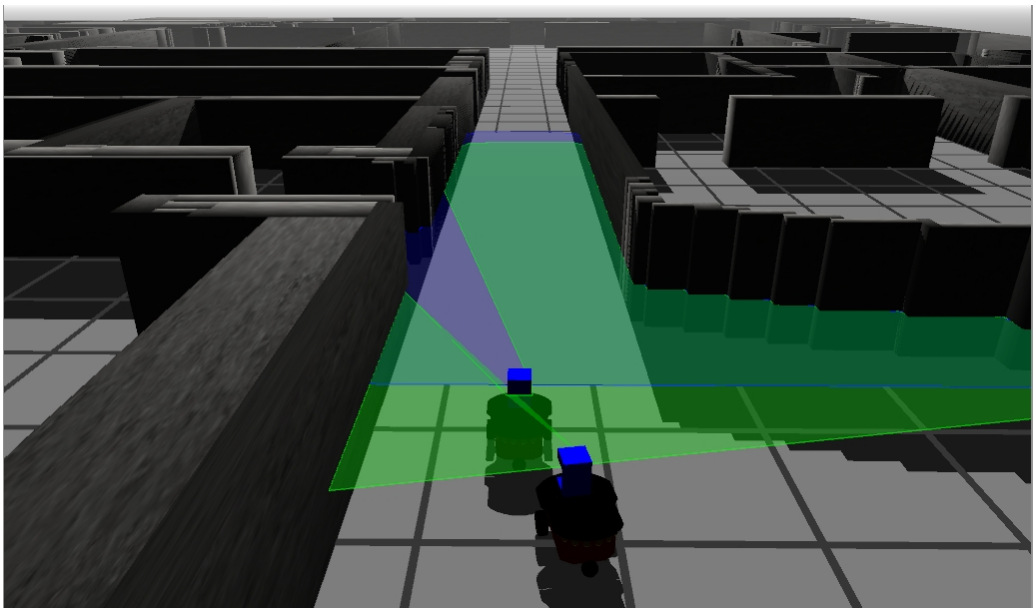


Figure 5.6: A screenshot showing an OpenRTM-aist controlled robot (leading robot with blue laser scan) working with a Player controlled robot (following robot with green laser scan) in Gazebo. Both Pioneer2DX robots are equipped with a simulated SICK LMS 200 laser rangefinder.

5.3.1 System Architecture

MRSim is a toolkit designed to integrate well into Gazebo, but it is also independent of Gazebo and uses its own XML file for configuring various properties of MR entities. MRSim is essentially a library of functions that is called inside the main execution loop of Gazebo.

Modifications to the Gazebo simulation framework are shown in Figure 5.7. MRSim has been integrated as an independent software module (in the source code level) without disrupting other Gazebo subsystems. From the diagram, it can be seen that MRSim exchanges messages with a device interface and a Gazebo interface. The two interfaces provide MRSim communication channels with real and simulated objects. The device interface (potentially a Player server or an RT Component running on the physical robot device) provides data captured by robot devices in the real world, while the Gazebo interface (potentially the Gazebo plug-in driver or the OpenRTM simulation interface layer) provides data generated by the simulation entities in Gazebo.

MRSim uses the information collected (and also other information specified by the user) through the two interfaces to create object representations, i.e. MR entities, and construct the MR environment. MRSim is also able to continue to monitor the two worlds through these interfaces and update the states of the MR entities over the execution cycle of the MR simulation. Moreover, the interfaces allow MRSim to send commands to control the behaviour of real and virtual objects associated to an MR entity and this is

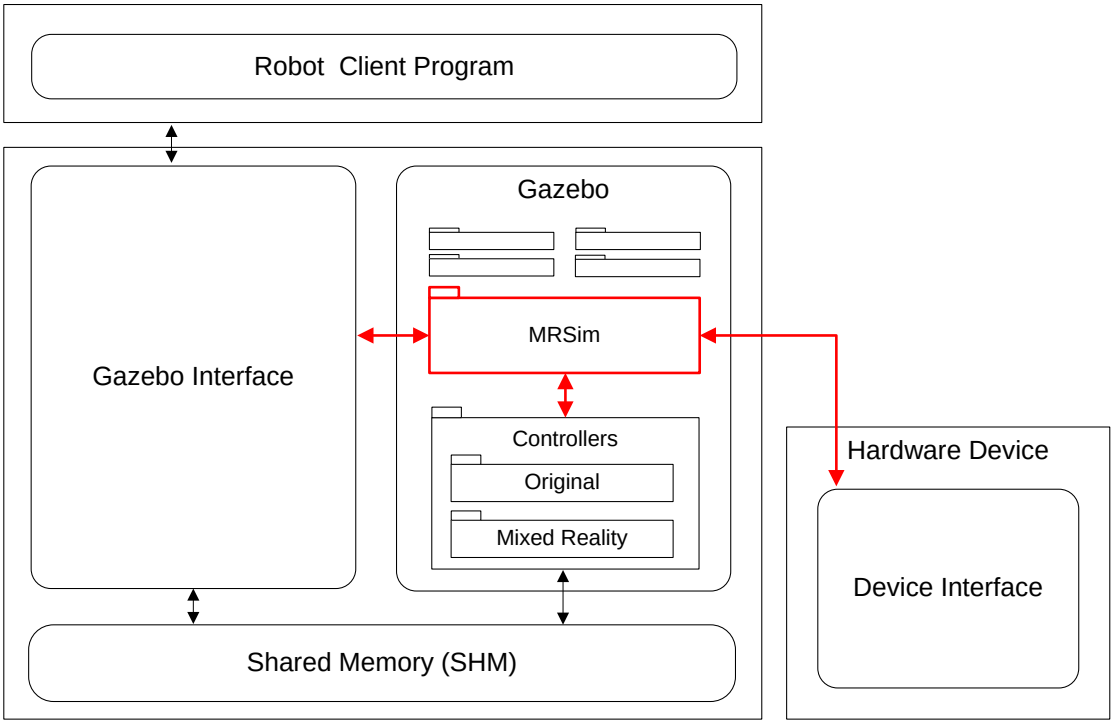


Figure 5.7: MRSim integrated into Gazebo. The new module and the associated data flows are highlighted in red. A set of mixed reality controllers have also been added to Gazebo to facilitate communication between MRSim and the client program.

often necessary for executing the response of MR interactions. For example, sending a command to stop the motion of a real robot gripper in order to simulate grasping a virtual object.

A set of MR controllers have also been added in addition to the original controllers provided by Gazebo. These controllers are responsible for publishing data generated by MR entities to SHM, as well as forwarding client commands read from SHM to MRSim. Since MRSim now handles all client commands, it is responsible for updating the corresponding real and/or simulated objects associated to MR entities accordingly when a command is received.

Note that the implementation is in accordance with Figure 5.1. The device interface provides data that form a representation of the real world, while the Gazebo interface provides access to data generated by Gazebo which represent the virtual world. MRSim is the MR simulation server that merges the two representations to create an MR world which the client communicates with over the set of MR controllers.

5.3.2 MRSim

The core components of MRSim are implemented based on the generic MR framework introduced in Section 4.1. The most unique elements of MRSim compared to other MR

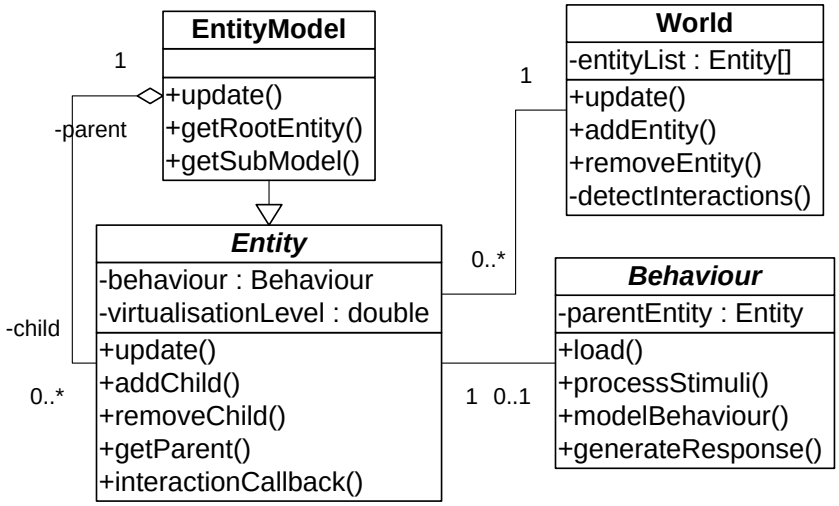


Figure 5.8: A UML class diagram showing the core classes of MRSim.

frameworks or tools identified in 2.4.2 are 1) the entity modelling method based on the degree of physical and functional virtualisation, and 2) the behaviour based interaction scheme formulated terms of state transitions and constraints. A UML class diagram capturing the core components of MRSim and the proposed behaviour-based interaction scheme is shown in Figure 5.8.

In an MR simulation, the robot carries out tasks while interacting with objects from different dimensions of reality. The MR simulation server, MRSim, is responsible for this phenomenon. Looking at Figure 5.8, the *World* is responsible for managing all *Entities*, including *Entity Models*, and updating them at each iteration of the simulation loop. For each real and virtual entity in the world, a corresponding representation is created in Gazebo. The world is then able to keep track of any physical interactions between the entities by monitoring Gazebo's physics simulation. An interaction map is also stored in the world and it contains information about the pairs of entities that have collided in the current iteration of the simulation. The interaction callback function associated with each entity is called when a physical interaction is detected. The *Behaviour* attached to an entity describes how interactions should be handled. In addition to physical interactions, the developer can implement customised behaviours that facilitate sensor based interactions and actuator based interactions.

Over the course of the work, a number of entities and behaviours have been implemented for creating MR simulations. The set of entities implemented in this work correspond to robotic devices that are commonly used for perception and navigation by mobile robot systems[3]. Although the set of implemented entities is far from being a complete database of the diverse robot devices available in the field or the different types

[3]The range of entities implemented was also limited by the simulation models available in Gazebo

of environmental objects that exist in the world, they were sufficient for demonstrating the concepts introduced in this research. The system is designed to be modular and extendible, and other entities and behaviours corresponding to devices, such as ultrasonic, tactile, and audio sensors, can be easily added in the future.

Entities

The entity class is responsible for managing the entity's representations. It contains information about its level of virtualisation, and this information is used by its attached behaviour when facilitating interactions. An entity can be associated with only a single parent entity but can have multiple child entities.

Each entity comprises two types of objects, physical and data objects. An entity can contain zero or more physical objects e.g. ODE bodies and joints, and OGRE meshes, that form its physical and graphical representations. An entity also includes a data object that stores messages being exchanged between the entity and the client program. The messages are essentially the commands sent by the client program and the entity's state information which are to be published. The entity class does not process these messages itself. The processing and data generation is carried out by its attached behaviour. Figure 5.9 zooms into the MRSim module inside Gazebo and shows the data flow to and from entities.

Conforming to the extension of the MR framework to robot simulation described in Section 4.2, an entity can be a sensor or an actuator device or an object in the environment. Gazebo provides the simulation models. Note that if the entity is real, i.e. zero level of virtualisation, a virtual counterpart also needs to be created in Gazebo so that we can model its behaviour. This is later shown to be necessary for creating MR interactions.

The design of the entities implemented in this research generalises notions from Player devices and OpenRTM component interfaces. Descriptions are given below.

Position Entity: A position entity is an entity with locomotion capabilities. The entity can be considered as a combination of proprioceptive sensors for measuring pose and velocity information, and actuators for driving its motion. It typically corresponds to a mobile robot platform. For example, an aerial robot that outputs pose information given by its GPS and Inertial Measurement Unit (IMU), or a wheeled robot that outputs odometry readings measured by its wheel encoders. A position entity can be controlled to move using pose and velocity commands, and it publishes data describing its actual pose and velocity status.

Actuator Array Entity: An actuator array entity is a robot device that consists of an array of actuators. This commonly represents a robot manipulator with multiple links and joints. The number of joints equals the length of the array of actuators. The entity can be controlled by an array of position, velocity, or current commands

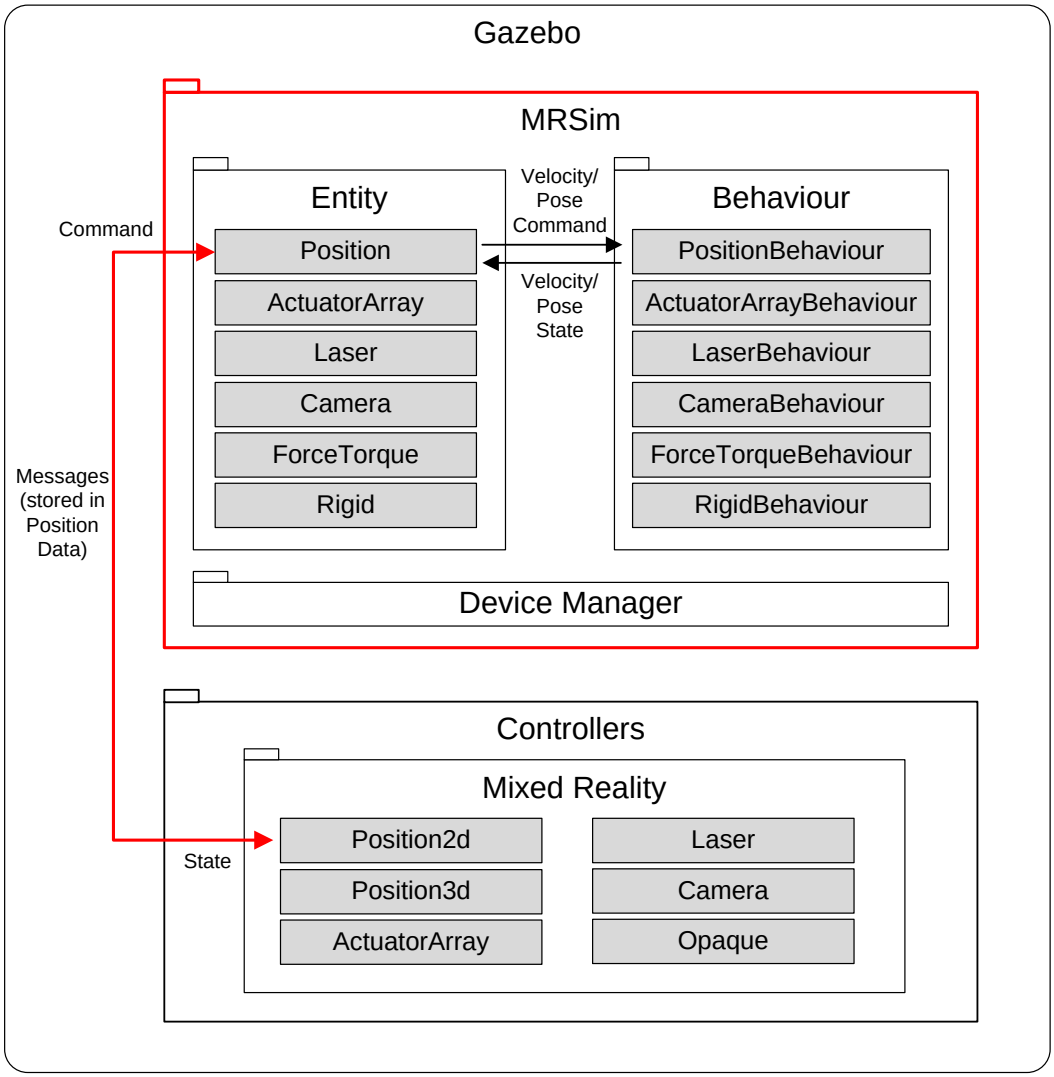


Figure 5.9: Data flow showing how MRSim handles commands from client programs and publishes entity's state information. As an example, position commands are forwarded to the position behaviour where it is processed, and state information generated by the position behaviour is sent by the position entity to the MR controllers where it is published to SHM.

in the joint space, and it publishes data about its actual joint positions, velocities, and currents.

Laser Entity: A laser entity represents a laser rangefinder device. The sensor uses laser beams (or rays) to determine the distance to an object. The entity does not take any commands. It publishes data describing the range and intensity values of each ray in the laser scan. Other configurations data associated to the laser entity include the number of rays, angular and range resolutions, and minimum and maximum angles of the laser scan.

Camera Entity: A camera entity provides views to the MR environment. There are two forms of a camera entity. The entity can either represent a camera device, e.g.

a webcam mounted on the robot, or a user camera for observing the simulation, e.g. a tethered camera following a robot. When a camera entity is used as a camera device, it publishes camera image data that are used by the client programs to complete their tasks. A camera device has intrinsic and extrinsic parameters. Intrinsic parameters include the focal length, principle point, skew and distortion coefficients, all of which can be configured. Extrinsic parameters are the position and orientation of the camera relative to its parent entity. When a camera entity is used as a user camera, it acts as a visual interface that provides users a view of the MR environment. More details on its use as a visual interface will be described in Section 5.4.1.

Force-Torque Entity: A force-torque entity refers to a robot device that measures forces and torques (or moments) acting on the device; it is used to represent a six DOF force-torque or force-moment sensor. The entity does not take any commands. It publishes data describing the three components of force in each axial direction and the other three components of moment around each axis.

Rigid Entity: A rigid entity represents a solid, non-deformable object which its physical behaviour can described by rigid body physics. This is usually used to represent an object in the environment, e.g. a wall, a task relevant object, e.g. a cube to be grasped, or a part of a robot body, e.g. a robot arm. Because a rigid entity is not a robot device, it does not take any commands or publish any data.

Behaviours

The behaviour class controls the logic of the entity, and implements the MR framework's behaviour-based interaction scheme described in Section 4.1.3. Each behaviour is different from another. It is noted that modelling of different entity behaviours and execution of their responses may require software- and/or hardware-specific implementations, thus the design supports custom behaviours to be extended and integrated into the framework.

To create a custom behaviour, the developer is responsible for implementing the three stages of interaction, i.e. stimuli processing, behaviour modelling, and response generation. The three stages are separated into three functions. Within these functions, the developers also need to implement their corresponding constraints. If a violation of the constraints is detected, i.e. a false or an error value is returned, the interaction process aborts and the developer can choose to output an error message to the screen.

Recall that in Figure 5.7, MRSim has two communication channels that interface it with the real and the virtual world. The behaviour class is one that makes use of these two communication channels (through the Device Manager, see Section 5.3.2). In a configuration file provided by MRSim, the user can specify zero or more real or simulated devices to

subscribe to in order to collect information necessary for modelling the behaviour during an MR interaction. Additional configuration parameters necessary for implementing the behaviour class can also be defined, and these configurations are loaded at runtime. In addition, a behaviour is also able to query the world which its parent entity exists in, and thus is able to obtain information about other entities that exist in the same world. The developer can use this information, together with the data retrieved from the specified devices, to model the behaviour of the entity. Appendix C.1 illustrates the classes that are related to the behaviour class, and Appendix C.2 shows how a behaviour attached to an entity is loaded in MRSim.

It is important to note that in the implemented system, not all interactions need to be manually created. Most often, only interactions between real and virtual entities need to be manually created. This is because virtual-virtual interactions are already modelled and generated in Gazebo, and real-real interactions occur naturally in the real world. However, in non-trivial situations, it may be advantageous to model real-real interactions. This relates back to the first guideline presented in Section 4.3. If a potentially dangerous command is sent to a real robot, the system could respond by modelling its interaction with other real entities and executing the command using a non-dangerous virtual representation.

The developer can check the class of reality of an entity by reading the entity's level of physical virtualisation. Once the behaviour has been modelled, the user has the choice of generating the response of the entity by sending appropriate commands to the subscribed devices (e.g. to move in a similar motion as the result from the modelling stage), and/or simply requesting the parent entity to update its physical and graphical representations. Finally, it is necessary to write to the parent entity's data object its new status.

The other task of the behaviour class is to process any commands sent by the client. These commands are handled in a similar way to how an interaction is created. Commands are also a form of stimuli that need to be processed and they go through the same three stages of interaction. Figure 5.10 illustrates the work flow of a behaviour object.

A number of custom behaviours have been implemented and they have been used to create MR simulations for testing robot systems in real world projects. The implemented behaviours are *Position Behaviour, Actuator Array Behaviour, Laser Behaviour, Camera Behaviour, Force-Torque Behaviour, Rigid Behaviour*. Note that the position behaviour and the actuator array behaviour initiate actuator based interactions, while laser behaviour, camera behaviour, and force-torque behaviour initiate sensor based interactions.

The interaction process carried out in each of the implemented behaviours is described below. Sequence diagrams illustrating examples of the interactions process in a number of behaviours are attached in Appendix C.3.

Position Behaviour: The position behaviour is responsible for handling commands to

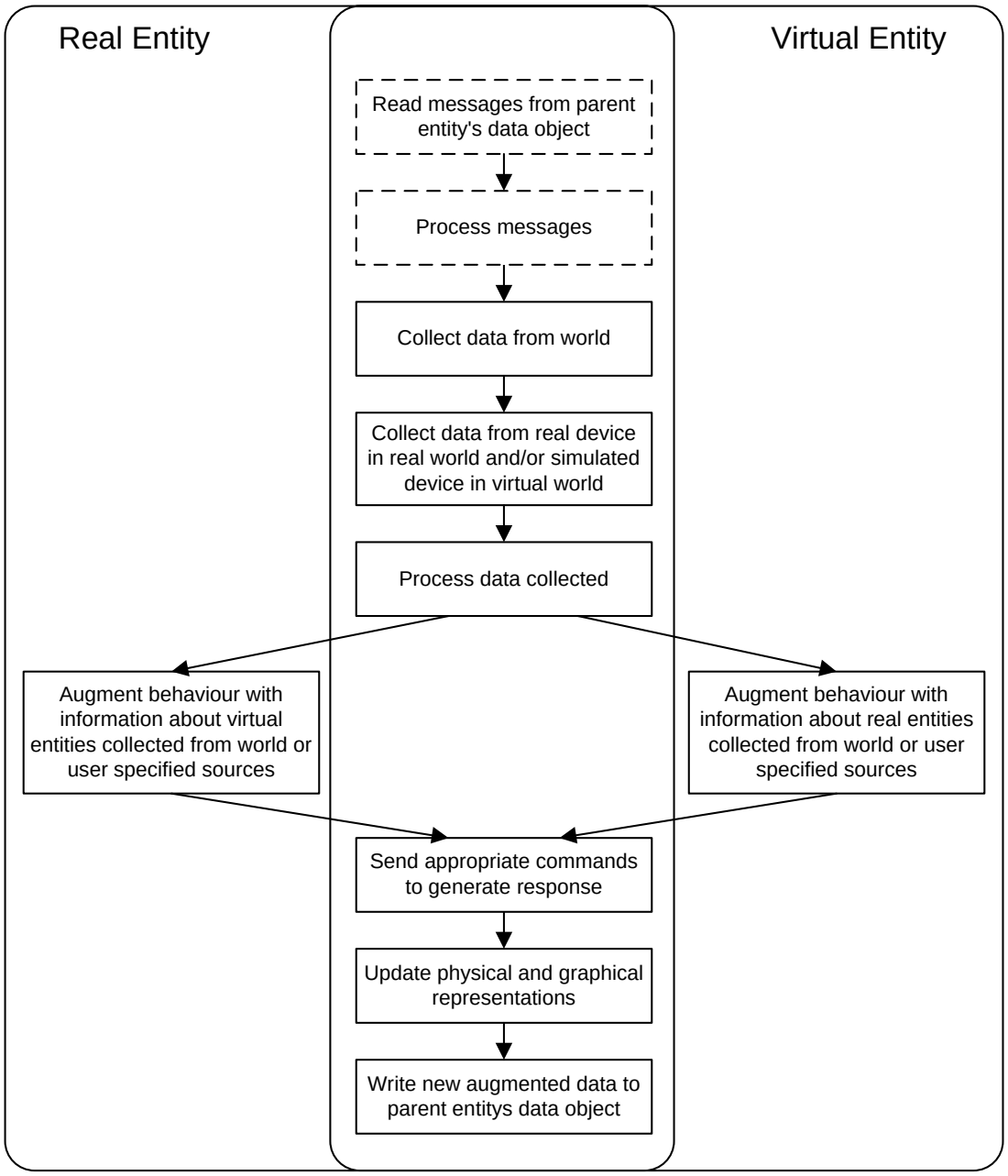


Figure 5.10: Work flow inside a behaviour object attached to an entity. The behaviour modelling step in the work flow varies slightly between a real entity and a virtual entity, as shown in the left and right columns respectively. The middle column contains steps that are shared by both real and virtual entities. Dashed boxes represent steps that are needed by entities that accept commands sent by clients.

control the motion of the position entity it is attached to, and generating the pose and velocity state data to be published by the position entity. Sequence diagrams illustrating the processing carried out in the position behaviour can be found in Appendix C.3.1.

Virtual Entity: If the position entity is virtual, the command is forwarded to the

simulated position device in Gazebo. No custom modelling is necessary as Gazebo already provides actuator models to simulate movements from position commands.

Real Entity: If the position entity is real, the command is dispatched to both the real position device and its virtual counterpart, i.e. the corresponding simulated position device in Gazebo. However, this may result in an inconsistent state between the real and the virtual world since discrepancies between the simulation and the real world are inevitable. It is important to keep the states of the two worlds consistent. The implemented position behaviour uses a rule-based strategy for handling commands to ensure consistency between the two worlds. There are two situations that the strategy addresses. The first is when a command arrives while no interactions are taking place. In this case, the implemented position behaviour lets user specify a third device that is able to provide pose information of the position entity in the real world, and this information is used to update its virtual representation whenever the discrepancy becomes too large. Different devices can be used to provide the pose information in practice, such odometry readings produced by the real device (provided that the drift is small), or pose estimates from a localisation algorithm. The virtual representation is updated either by directly setting the pose of the physical and graphical objects representing the entity in Gazebo, or by sending appropriate commands to the simulation device to correct its movement. The second case is when a command arrives while an interaction is taking place. The implemented behaviour now gives priority to the outcome of the interaction and reverses the update strategy. As opposed to updating the virtual representation, the state of the real position is updated according to the modelled behaviour. Figure 5.11 shows the decision flow in an interaction process for a position behaviour attached to a real entity.

Consider the example where a real mobile robot is modelled using a position entity. Velocity commands sent to the robot can cause the pose of its virtual counterpart to diverge from the actual robot pose over time. To correct this discrepancy, the virtual representation can be updated using pose estimates from an external motion tracking device that constantly monitors the state of the real robot. If the robot now collides with a virtual obstacle, the strategy is reversed by sending pose or velocity commands to move the real robot in a similar manner to the result of the modelled behaviour.

Actuator Array Behaviour: The actuator array behaviour is responsible for handling commands to control the joints of the actuator array entity it is attached to, and generating data to describe the state of the joints to be published by the actuator

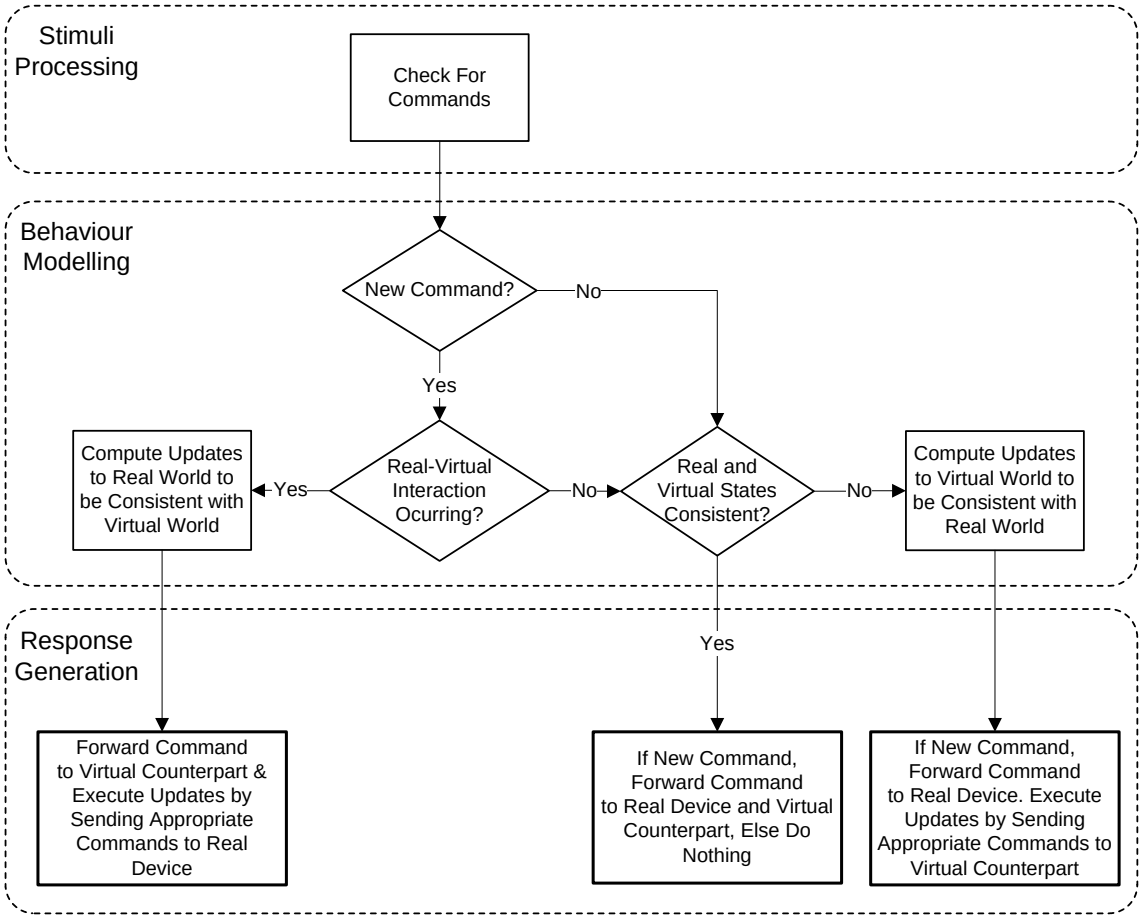


Figure 5.11: Decision flow of an interaction process in a position behaviour attached to a real entity. Depending on the situation, appropriate commands are generated to keep the real world and the virtual world in a consistent state. The same decision flow can be applied to other behaviours that need to handle client commands during interactions.

array entity. The actuator array behaviour carries out the three stage of interaction in a similar way as the position behaviour.

Virtual Entity: If the actuator array entity is virtual, the command sent by the client program is forwarded to the simulated device in Gazebo.

Real Entity: If the actuator array entity is real, the command is dispatched to the real device and its virtual counterpart in Gazebo. To synchronise the motion of the virtual counterpart with the real actuator array device, the implemented actuator array behaviour also takes a similar approach to the position behaviour and lets the developer specify a third device that is able to provide real joint positions for updating those of the virtual representation in Gazebo. This can either be the joint position readings from the real device, or joint positions computed by an inverse kinematics solver given that the pose of the actuator array's end-effector is available instead. In an interaction with a virtual entity, the modelled behaviour

is propagated to the real world by sending position, velocity, or current commands to the real device to move the entity in a similar manner to the results of the modelled behaviour. The same decision flow illustrated in Figure 5.11 is used in the interaction process of the actuator array behaviour.

Laser Behaviour: The laser behaviour is responsible for generating the range and intensity data to be published by the laser entity it is attached to. Sequence diagrams illustrating the interaction process in the laser behaviour can be found in Appendix C.3.2.

Virtual Entity: If the laser entity is virtual, the behaviour needs to identify real entities in the world and whether their locations and dimensions are known during the stimuli processing stage, and use this information to generate the new laser ranges that reflect real objects in its scan. In the ideal case, virtual representations of all real entities would have been created in Gazebo, and the behaviour can then be modelled by Gazebo using its laser sensor models. In other cases such as simulations in unknown environments where not all objects in the real world are known in *a priori*, the developer can specify devices capable of detecting and providing pose and geometric information about objects in the real world online in order to dynamically create their virtual representations.

Real Entity: There are two ways to augment the real laser scan to reflect the presence of virtual entities in the world if the laser entity is real. The first being a similar approach as the virtual laser behaviour, which identifies virtual entities in the world and uses the information to generate the new range values. The second, which is the method implemented in the MR robot simulator, is to simply fuse the range values read from the real device with those from its virtual counterpart in Gazebo. The two range values are mixed by taking the minimum value of each ray in the laser scan. The resulting range data reflect both real and virtual entities in its laser scan. Note that this method requires the pose of the real laser device and its virtual counterpart to be consistent to generate accurate results, particularly when the laser is mounted on a moving entity (e.g. a position or actuator array entity).

Camera Behaviour: The camera behaviour is responsible for generating camera image data to be published by the camera entity it is attached to, or to be rendered and displayed to the user if it is a user camera.

Virtual Entity: If the camera entity is virtual, the command is forwarded to the simulated camera device in Gazebo. No additional modelling needs to be carried out to augment the virtual camera image data with real entities. This is because given that graphical representations of real entities have already been created in Gazebo,

the virtual camera image data provided by Gazebo would reflect the presence of these entities. On the other hand, it is also possible to register video silhouettes of real objects into their corresponding locations in the virtual camera image by using libraries such as 3DLive and MXR Toolkit [16].

Real Entity: For a real camera entity, the behaviour augments virtual information onto images captured by the real camera device. In the behaviour modelling stage, virtual entities can be overlaid onto the real camera image data using AR techniques. MRSim provides a markerless AR component that is able to track the camera pose by analysing natural features in the scene. More details on the markerless AR component can be found in Chapter 6. Note that the AR component is interchangeable, the developers can integrate other AR techniques they prefer. For example, the ARToolKit library can be used in indoor controlled environments if modifications to the robot environment are acceptable for the task.

Force-Torque Behaviour: The force-torque behaviour is responsible for generating force and torque data to be published by the force-torque entity it is attached to. The force-torque behaviour carries out the three stage of interaction in a similar way to the laser behaviour.

Virtual Entity: If the entity the behaviour is attached to is virtual, the behaviour first identifies the virtual representations of any real entities in the world that are interacting with the virtual force-torque entity, then models the behaviour using the force sensor model provided by Gazebo.

Real Entity: If the force-torque entity is real, a simple way to model the behaviour is to mix the force and torque values read from the real device with those from its virtual counterpart in Gazebo by taking the maximum value of each component of force and torque then thresholding these values by the load capacity of the real device. However, in complex interactions involving multiple forces and torques, it is important to verify whether the real device supports measuring forces or torques in opposing directions. For some devices, the net effect of opposing forces and torques is calculated instead of outputting the maximum values of all components.

Rigid Behaviour: The implementation of the rigid behaviour is slightly different from other behaviours. A rigid entity is a passive object and does not initiate interactions. The only interactions that occur are physical contacts from other entities in the world. Sequence diagrams illustrating the interaction process in the rigid behaviour can be found in Appendix C.3.3.

Virtual Entity: If the rigid entity is virtual, the interaction process relies on the Gazebo's physics engine to compute and generate the resulting behaviour.

Real Entity: If the rigid entity is real, the behaviour monitors the state of the corresponding rigid object in the real world so that when a change in its state is detected, its virtual representation can then be updated in Gazebo to keep the two worlds consistent. A number of possible devices for monitoring and tracking the state of real world objects include cameras, motion capture systems, GPS and IMU devices. For example, if a real rigid object was knocked over by a real robot in the real world, the behaviour checks the physics simulation results from Gazebo, then updates/corrects the pose of the rigid entity's virtual representation accordingly based on pose outputs from an external vision based tracking system. The response generation stage may require controls to devices or other mechanisms for moving the corresponding rigid object in the real world according to the modelled behaviour. Depending on the availability of these mechanisms, it determines whether a full interaction can be achieved or not. If a full interaction is not created, inconsistent states between the real rigid object and its virtual representation may occur, and the behaviour needs to keep track of the inconsistent states and acknowledges the inconsistencies to the users if necessary, e.g. by using visualisations.

Device Manager

MRSim provides a device manager that manages different robot software frameworks used for communicating with other robot devices or components outside MRSim. The job of the device manager is to instantiate and manage connections with devices specified by the users. As described earlier, these devices are used in the behaviour class to access data and control objects in the real and virtual worlds.

Device connections are managed through the two robot software frameworks, Player and OpenRTM, and more can be added by implementing a set of predefined interfaces for each device. In the design of an MR simulation, the developer is not restricted to use only one particular framework. The developer is able to specify in the configuration file the robot software framework to use for each device. Thus, different frameworks can be used in the same simulation session for communicating with different robot systems. Figure 5.12 shows the inter-process communication handled by the device manager through Player and/or OpenRTM.

The list of implemented devices are *Position2d Device, Position3d Device, Actuator Array Device, Laser Device, Camera Device, Force-Torque Device, and Opaque Device*[4].

The device manager is simply to provide convenience to the developers for accessing robot components outside MRSim. Robot software frameworks such as Player and OpenRTM have been chosen and integrated because they provide standardised interfaces

[4]The notion of an *opaque* device is inherited from its use in Player, which refers to a device for exchanging generic messages with arbitrary content.

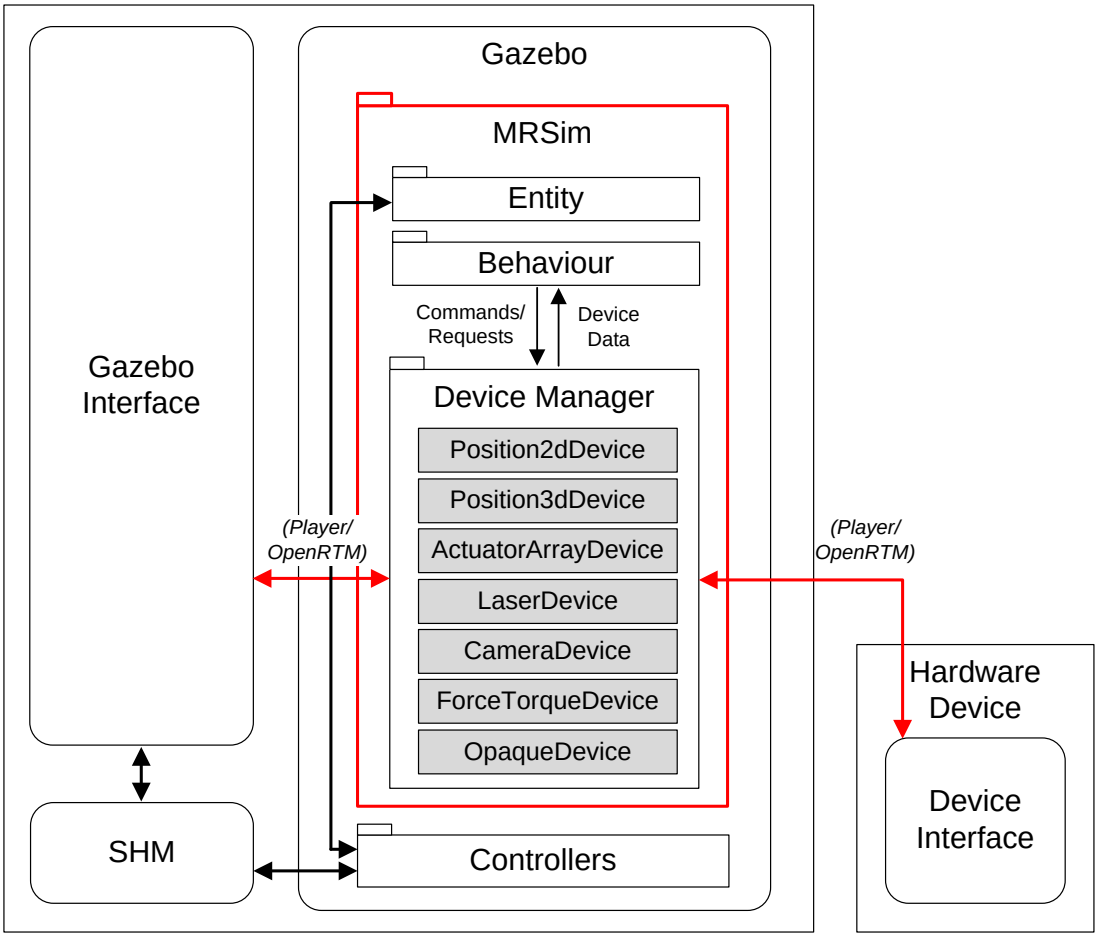


Figure 5.12: Data flow managed by the device manager in MRSim.

for communication, and it is believed that standardisations in robotics are important for increasing the reusability of robot software systems. If the developer wishes to use custom libraries to communicating with robot device, he/she can also do so within the behaviour class to be implemented.

5.4 Visualisation

Until now, this chapter focused on the operation and the data flow between components in the MR robot simulator. This section now describes visual interfaces designed to help users observe MR simulations. The literature review identified a number of MR visualisation techniques and their applications in the field of robotics. Many of these techniques have been proven to provide users a more intuitive way of interpreting complex robot and task relevant data. This research investigated the suitability of the different MR visualisation techniques and how they can be applied or modified in order to create visual interfaces for the MR robot simulator.

This section describes the design of the visual interfaces and assumes an AR algorithm

is available to create these interfaces. Details on the markerless AR algorithm used in the design will be presented in Chapter 6.

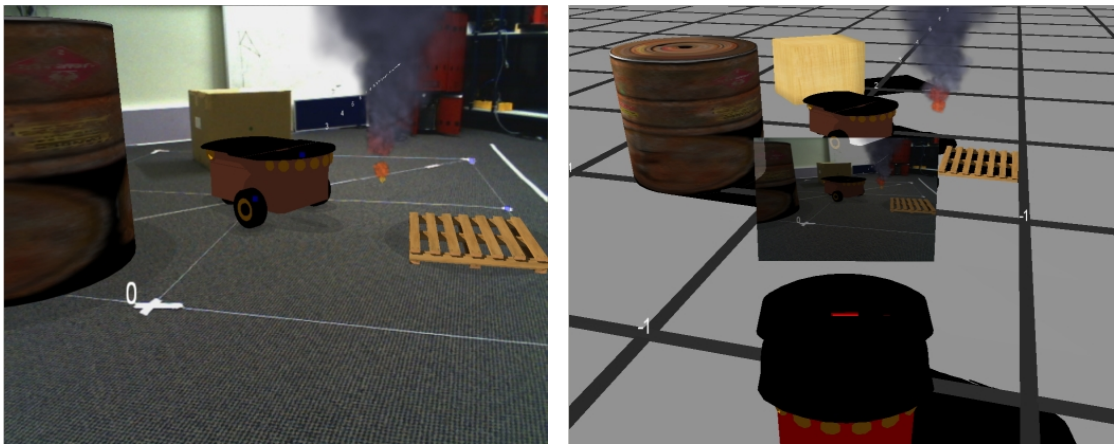
5.4.1 MR Interfaces for Robot Simulation

As mentioned in Section 5.3.2, visual interfaces of the simulation environment are provided by views seen by a camera entity, and its attached behaviour determines how the visualisation will be created. This research presents two forms of visualisations to help users observe the MR simulation environment. Their designs are motivated by the requirement to improve the user's context awareness of the robot development environment as described in Section 3.2.3.

One of the most common MR visualisation techniques applied in robotics is AR visualisation and its benefits have been presented in Section 2.4.3. The ability to present synthetic or virtual information in context with the real physical environment in real time is a unique advantage of AR, which makes it particularly suitable for creating a coherent visualisation of entities in MR simulation environments. The users are able to view the MR simulation environment with virtual entities registered in geometric registration with the real world scene. To create AR visualisations, a real camera entity is required. The camera behaviour of the real camera entity uses a markerless AR algorithm to generate AR image data which are rendered to the screen and displayed to the users; this forms the AR interface. In the AR interface, virtual entities in the MR simulation environment are shown overlaid in geometric registration with the real world scene captured by the real camera. Figure 5.13(a) demonstrates the AR interface created for an MR simulation of a robot teleoperation task.

Nevertheless, there are limitations when relying on a single AR interface for observing MR simulations. Development of robot applications needs to allow users to observe the simulation environment from different perspectives. This essential requirement is one that is difficult to satisfy with an AR interface alone. AR relies on the use of a physical camera to provide images on which visual augmentation takes place, but only from a single view. The experimental setup could use an overhead camera to capture a view of the physical environment where MR simulation takes place, but this becomes infeasible in large environments, especially outdoors. While multiple cameras could be used, for example in the MR environment setup in [192], the setup is expensive and may be difficult to port to a different environment. The developers could also be equipped with a wearable AR system and HMDs so they have the freedom of moving around and observing the simulation environment, but it raises ergonomic issues concerning the load the users have to carry while navigating in a potentially large environment.

The limitations of AR can be compensated using an AV interface. The AV interface offers a virtual view of the MR environment augmented with information from the real



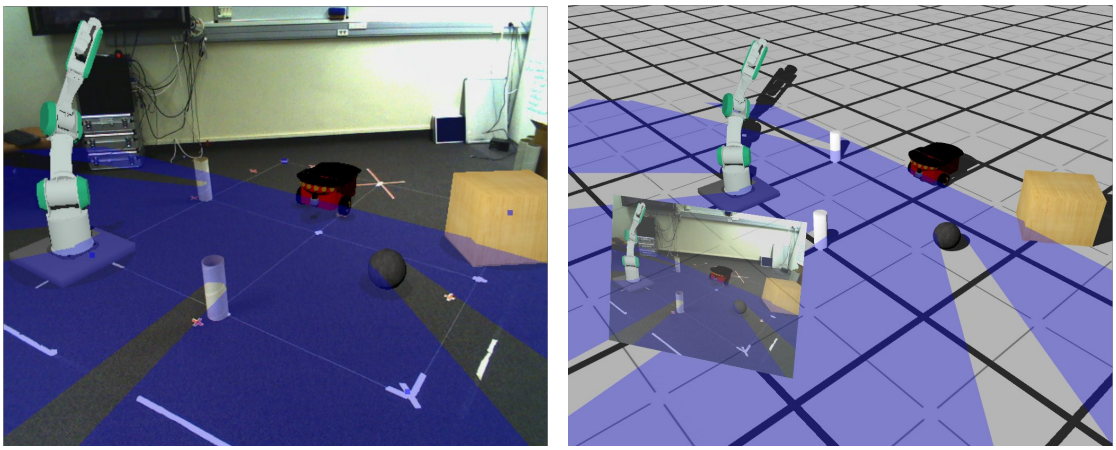
(a) (b)

Figure 5.13: An MR simulation of a robot teleoperation task. (a) The AR interface showing the view seen by a real camera entity mounted on the real teleoperated robot. In the AR interface, the barrel, the Pioneer robot (brown), and the wood pallet are virtual, while the yellow box at the back is real. (b) The AV interface showing the view seen by a virtual camera entity. In the AV interface, the AR view is shown rendered in a video panel and placed in front of the real teleoperated robot's virtual representation (red) to show its view of the world.

world. In MR simulations, the AV interface is essentially provided by a virtual camera entity that is able to freely move around the virtual simulation environment created by Gazebo. The users are able to see virtual representations of real entities in this interface, as well as all virtual entities. Moreover, the other advantage is the ability of creating multiple virtual cameras of arbitrary viewpoints that allows users to observe different regions of the simulation environment at minimal cost. Compared to the AR interface, the AV interface gives users a greater flexibility in observing the simulation environment, but from a virtual perspective. The AV interface helps to enhance the users' global awareness of the simulation components and potentially dangerous situations. Figure 5.13(b) demonstrates the AV interface created to provide an extended view of the simulation environment for the robot teleoperation task. A video, `AVInterfaceDemo.mp4`, demonstrating the AV interface shown in Figure 5.13 is enclosed on the accompanying CD-ROM.

The design of the AV interface is based on the ecological interface paradigm proposed by Nielsen *et al.* [190]. The paradigm stresses the importance of conveying information that supports the user's direct perception of the robot's *affordances*[5] in the environment in order to facilitate the decision making process. While Nielsen *et al.* integrate multiple sets of robot information in an AV display to enhance situation awareness of remote operators in robot teleoperation tasks, the same interface design can also benefit debugging and monitoring tasks in robot simulations. The developers need to be supplied with visualisations of robot data and the spatial relationships between the datasets in order to

[5]The theory of affordances was introduced by Gibson [115] who describes affordances as all action possibilities associated with an object.



(a) (b)

Figure 5.14: An MR simulation involving a virtual Pioneer robot randomly walking in the real physical environment filled with a virtual PA10 robot manipulator and other real and virtual objects. (a) The AR interface showing the view seen by an external real camera entity looking over the simulation environment. In the AR interface, the two robots, the sphere, and the box are virtual, while the two cylindrical objects (white) are real. (b) The AV interface showing the view seen by a virtual camera entity.

understand the robot's perception of the world and also how the perceived data influences the actions of the robot during the development process. Through the AV interface, the users are able to see the spatial relationship between real and virtual entities which are placed in their corresponding geometric locations in the virtual environment. The robot's sensory information is visualised and updated in real time. The AR view seen by a real camera entity is also rendered onto a video panel and immersed in the virtual environment; the video panel object itself is a graphical representation associated to a camera entity. Figure 5.14 shows another MR simulation example and its AR and AV interfaces.

The AV interface can also be created from different modes of a virtual camera. The implemented system supports three forms of a virtual camera: free-look camera, tethered camera, and fixed camera. Figure 5.15 shows a screenshot of the MR robot simulator with multiple camera modes. The AV visualisations are shown embedded in multiple render windows within the original Graphical User Interface (GUI) of Gazebo. Each AV visualisation window is associated with a virtual camera that is attached to a scene node in the Gazebo's virtual environment. In the virtual environment, visualisations of real world data are created by adding 3D renderings to the scene. For example, the left window in Figure 5.15 shows the addition of a video panel object placed in front of the robot. This is created by rendering the robot's onboard camera image data as a dynamic texture on a rectangular mesh, then positioning the mesh in the direction that the camera is facing. In addition to the 3D simulation environment, the Gazebo GUI provides other simulation information such as the frame rate, total run time, and geometric and physics

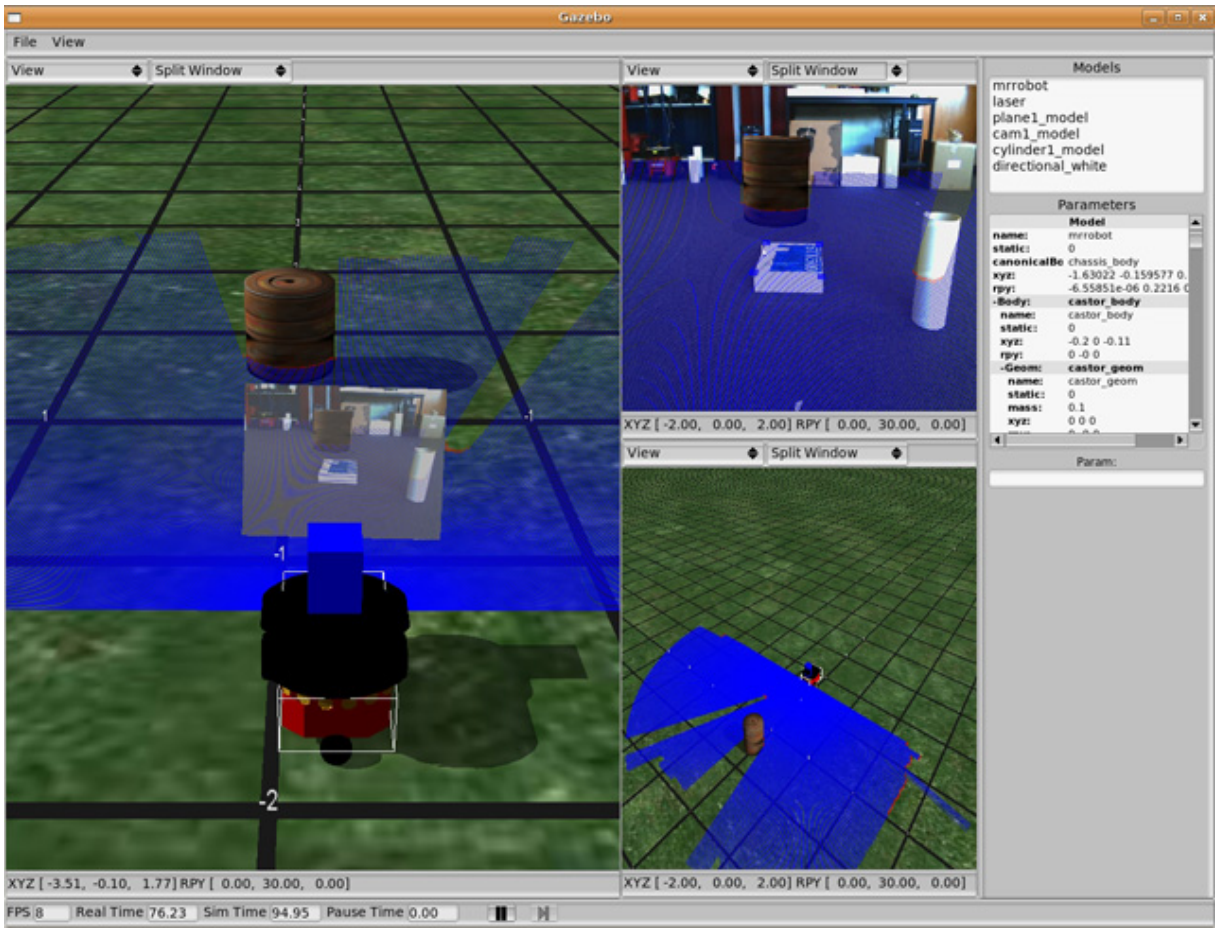


Figure 5.15: A screenshot of the MR robot simulator providing multiple camera modes. Left: tethered camera mode, Top Right: First person perspective using the AR interface, Bottom Right: fixed camera mode. In this MR simulation example, it shows a typical situation when objects in the real world have not been modelled. The AV interfaces show that while the virtual representation of a real cylindrical object (the white cylindrical object in the AR interface) was not created, its position is still reflected in the laser scan. The barrel further away from the robot is virtual.

properties of entities in simulation.

5.5 Functional System Validation

This section presents a preliminary validation of the MR robot simulation system. It demonstrates MR interactions based on the behaviour-based interaction scheme. As this is only a preliminary validation of the system, no formal evaluation methods were used. Quantitative and qualitative evaluations will be presented in Chapter 7.

5.5.1 Interaction Examples

First, it is worth pointing out that the images displayed within the AR and AV interfaces, such as the ones shown in Figure 5.13, can be considered as the result of sensor based

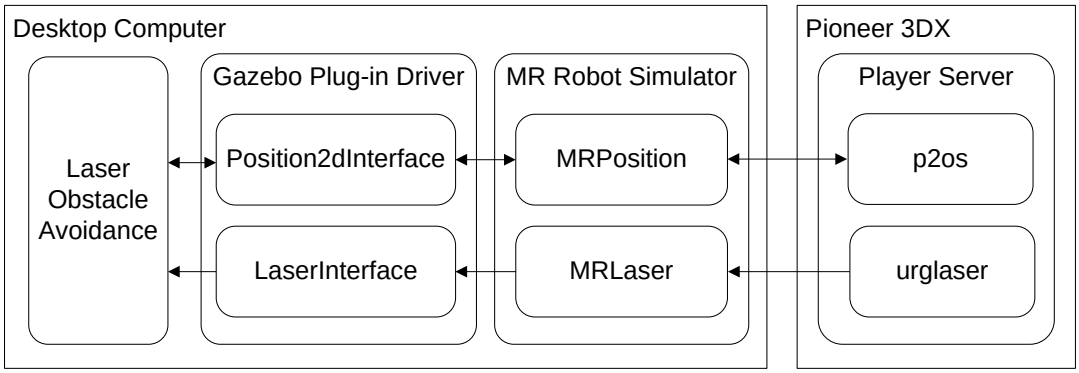


Figure 5.16: System diagram of the obstacle avoidance MR simulation. In simulation involving a real robot, the MR robot simulator connects to the Player server running on the Pioneer3DX robot and exchanges messages with its devices for creating interactions. In simulation involving a virtual robot, no connections will be made to the the Pionner 3DX robot platform; the MR robot simulator connects internally to the Gazebo simulation devices.

interactions between vision sensors and virtual objects in the simulation environment. The images are generated by behaviours attached to camera entities, and in simulation of vision based robot systems, these augmented images can be used to test client programs relying on vision data.

To demonstrate another example of sensor based interaction, an MR simulation of an obstacle avoidance algorithm is created. The algorithm instructs the robot to randomly navigate around the environment and avoids obstacles by analysing its laser range readings. In summary, the algorithms operates by first dividing the robot's laser scan into left and right sides and determining the minimum range values on each side. It then computes the linear velocity command based on the combined result of these two minimum range values, and the angular velocity command by taking the difference between the two minimum range values, i.e. the robot moves slower if an obstacle is close on either side, and turns faster to the left if the obstacle gets closer to its right. The same algorithm was run on a real and a virtual robot equipped with a real and a virtual laser sensor respectively. The interaction involves a laser entity and a number of rigid entities (obstacles) in the environment. Virtual obstacles were placed in the environment in the case of a real robot equipped with a real laser sensor, while real obstacles were used in the opposite case involving a virtual robot equipped with a virtual laser.

The client program runs on the Player framework. Figure 5.16 illustrates the data flow between the components in the MR simulation. The real Pioneer 3DX robot carries a mini-ITX machine that runs a Player server which the MR robot simulator connects to over a local wireless network. The Player server includes a `urglaser` driver that reads range data from the onboard Hokuyo URG laser rangefinder, and a `p2os` driver that writes velocity commands to the Pioneer 3DX robot platform. In the case of a virtual robot,

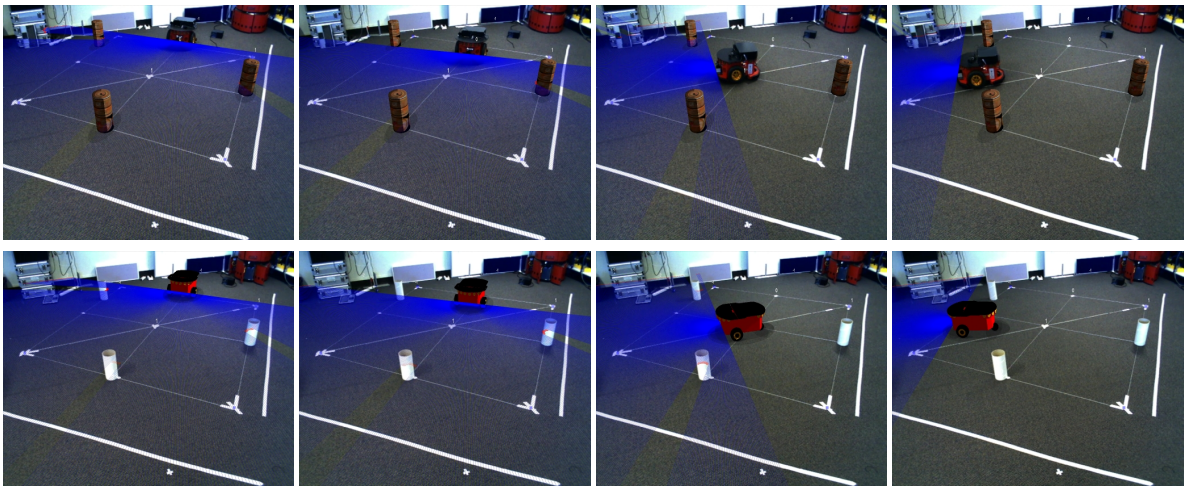


Figure 5.17: A sequence of screenshots illustrating a sensor based interaction between a laser sensor device and three cylindrical objects in the simulation environment. Top Row: A real robot detects and avoids virtual obstacles. Bottom Row: A virtual robot detects and avoids real obstacles.

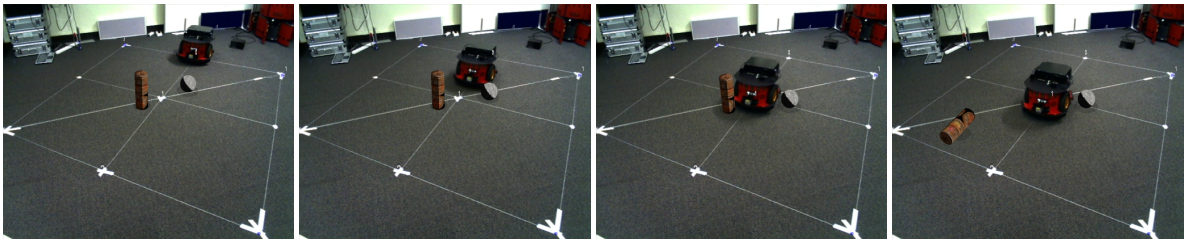


Figure 5.18: A sequence of screenshots illustrating a contact interaction between a real robot and two virtual objects.

the Pioneer robot model provided by Gazebo is used but modified to equip a laser sensor with the same configurations as the real one.

Figure 5.17 shows the sensor based interaction. The robot successfully detected the presence of the obstacles in both cases and navigated towards the open space. Two videos, `ObstacleAvoid_RealRobot.mp4` and `ObstacleAvoid_VirtualRobot.mp4`, demonstrating these sensor based interactions are enclosed on the accompanying CD-ROM. In Chapter 7, it will be shown that this sensor based interaction is applied in a robot search and rescue task, and a detailed quantitative analysis on the results of the obstacle avoidance MR simulation will be presented in Section 7.1.1.

An example of actuator based interaction is shown in Figure 5.18. A real robot was manually operated to move at a constant velocity until it collides with virtual objects in its path. The MR simulation system setup is similar to the previous sensor based interaction example except without the laser component. The physics engine in Gazebo was used to model the dynamic behaviour of the two entities in interaction. The response of the virtual objects were successfully generated and visualised. However, this example

did not alter the motion of the real robot according to the physics simulation, thus only a partial interaction was achieved. A full actuator based interaction example can be found later in Section 7.2.2. A video, `Collision_RealRobot`, demonstrating this actuator based interaction is enclosed on the accompanying CD-ROM.

5.6 Discussions

This chapter detailed the implementation of an MR robot simulator. It began by introducing the base simulation platform, Gazebo, which provides the simulation models of sensors, actuators, and the environment. It also showed how Player client programs as well as client RT components communicate with Gazebo over the two simulation interfaces. MRSim was then integrated into Gazebo as a separate module to provide MR simulations. MRSim implements the MR framework described in Chapter 4 and includes a number of custom behaviours to facilitate interactions between real and virtual entities. In particular, the camera behaviour of a real camera entity uses AR visualisation techniques to create the AR interface for helping users to understand MR simulations, while virtual cameras with different camera modes have been used to create the AV interface. The next chapter will describe in detail the AR visualisation techniques implemented.

The implementation of the MR robot simulator has a number of key differences between closely related work such as the AR framework for underwater robot simulation [86], the AR Multi-UAV system [116], and the MR environment for humanoid robot development [248, 192]. First, the MR robot simulator builds on a general purpose robot simulator. This unique element of the MR robot simulator enables the simulation tool to be used across different robot applications. Moreover, MRSim is designed to be independent of Gazebo, so integration with application-specific simulators would also be possible. Second, this research acknowledges the positive impact of open source software in robotics, thus integrates widely used open source robot software frameworks to enable the MR robot simulator to support simulation of different robot systems. Third, MRSim creates real-virtual interactions between robot components, i.e. sensors and actuators, as well as environmental objects, i.e. rigid entities, as opposed to augmenting only the robot's sensory perception as seen in all existing work. Finally, MR robot simulator provides real time MR visualisations to aid users in observing MR simulations and interpreting complex robot data during robot development, and these are not available in recent hybrid/HIL simulation tools such as [86] and [116].

Of the four differences, the most unique element is the creation of real-virtual interactions. While there exist work in the field of MR on forming and combining representations from real and virtual objects [42, 193], there has not been a generic method or implementation demonstrating interactions between them. MRSim fulfills this gap. Its integration

in the MR robot simulator provides support for richer interactions, which enables the MR robot simulator to be used beyond simulation of mobile navigation tasks (a typical application of HIL simulations) to robot manipulation tasks that demand physical interactions. Notably, the general purpose nature of the MR robot simulator provides simulation models to support testing of these different robot systems.

A preliminary functional validation of the MR robot simulator has shown that the system correctly facilitates MR interactions (full and partial interactions). It was also shown that the custom behaviours implemented were sufficient to create a simple MR simulation of an obstacle avoidance algorithm. More entities and behaviours could be added in the future to enhance the system to support a wider range of robot applications. An evaluation of the MR robot simulator will be presented in Chapter 7 which demonstrates MR simulations of more complex scenarios.

6

Augmented Reality for Robotics

A major part of this research investigated the use of AR visualisation and identified various AR tracking and registration techniques. In the MR robot simulator, AR visualisation was used to create one of the simulator's visual interfaces. The previous chapter discussed its advantages and disadvantages, and examples of AR visualisation were shown. In addition, AR is also used to augment real camera image data in sensor based interactions with virtual entities. AR visualisation plays an important role in this research, and thus effective AR is crucial in the use of the MR robot simulator.

This chapter details the implementation of an AR system. The system has been implemented as a separate library independent of the MR robot simulator, as it is believed that many other areas of robotics (and potentially outside of robotics) could also benefit from the use of this library. The library is used by the custom camera behaviour in the MR robot simulator to provide AR visualisations.

This chapter is organised as follows: Section 6.1 describes a number of requirements that have been devised for applying AR in robot tasks. The requirements motivate for a real time markerless AR tracking algorithm. Two AR configurations are proposed, where one is built on another. Section 6.2 describes the first configuration, which creates AR based on tracking planar regions. Section 6.3 presents the second configuration, which adds an extension for tracking and mapping in unknown scenes.

6.1 AR Requirements

The primary task of an AR algorithm is to mix virtual information onto video imagery captured by a camera in the physical environment. The algorithm computes estimates of the camera position and orientation while the camera moves around the environment. These estimates are used to update the pose of a virtual camera so that renderings of virtual objects remain aligned with the live background video images, appearing as if they are part of the physical environment. The resulting augmented images provide a display of virtual information that is accurately registered in its geometric location in the real world.

Previous applications of AR in robotics have mainly employed marker based tracking methods to provide camera pose information for registering virtual objects in a real world scene, e.g. [80, 195]. Although these methods are a fast and low cost solution for creating AR, they suffer from limitations concerning occlusions and illumination changes. Other methods include using expensive equipment and modifying the experimentation environment, e.g. [248]. However, environment modifications may not be feasible in many robot tasks. For example, modifying the environment is particularly challenging for tasks that take place in large, outdoor environments. Moreover, since robots depend on the environment in various tasks, modifying the environment could affect the behaviour of the underlying robot software system. The main requirement is then the need for a markerless AR algorithm for visualisations in unprepared environments.

A common use of AR visualisation is in robot development and remote operation tasks. The design of the AR visualisation must take into the account that these users are often required to observe the robot environment from different viewpoints. Moreover, cameras may also be mounted on moving platforms with unpredictable motions. Therefore, there should be no restrictions on the movement of the camera used to provide images for creating AR visualisations. The camera can be fixed or moving. An essential requirement is to develop an AR algorithm that is able to track the pose of a free-moving, six DOF camera.

A user requirement pointed out in Section 3.3 is that visualisations need to be provided in real time to help increase user awareness. This is particularly important for improving the safety of experiments, especially in applications that require robots to operate in remote locations. This research defines real time performance to be >25Hz. On the other hand, creating AR for applications in field robotics raises the need to address possible high latency issues of video transmission, requiring AR to operate on video streams with a reduced frame rate caused by low bandwidth. For example, findings in the field of telerobotics recommend that 10 Hz be the minimum frame rate to avoid performance degradation [77]. Thus, this provides an indication of the video frame rate which AR

tracking algorithms would be required to operate on.

In an ideal situation, all features to be tracked would remain within the camera view throughout the whole video sequence. However, this does not always happen in practice. During tracking, features are easily lost due to sudden rapid camera motions, occlusions, motion blur, or changes in illumination in the environment. When tracking fails, virtual objects would become incorrectly registered or completely lost. Therefore, another requirement is the need for a failure recovery strategy. The algorithm needs to relocalise the camera whenever tracking failure occurs. The process should be automatic and minimise manual user interventions.

Based on these requirements, marker based AR techniques may not be the best solution, especially in the dynamic environments that robots operate in today. Existing markerless AR techniques reviewed in Section 2.3.1 demonstrate a viable approach to tracking in unprepared environments. However, robust tracking tends to be achieved at a higher computational cost. While highly optimised AR approaches found on mobile phones [270, 150] can be used, they are currently limited to operate in scenes with highly textured contents. This has motivated the first AR configuration proposed in this research, which focuses on providing an efficient AR tracking method with a failure recovery strategy. A more difficult problem of applying AR in robotics is the need for the system to operate in large, unknown environments. This has motivated the second AR configuration which incorporates the SfM based PTAM algorithm [148] to provide accurate, real time AR at a higher computational cost. A number of necessary enhancements are made to the algorithm in order to create geometric and scale consistent AR for robotics.

6.2 Configuration One: Markerless AR

The first AR configuration proposed in this research is a markerless AR algorithm based on tracking planar features in a sub-region of the scene for localising the camera [73]. The proposed method combines point feature tracking and object detection to provide an AR solution that is fast and capable of recovering from tracking failures. In contrast to strengthening the feature matching process with the use of computationally expensive feature descriptors, the algorithm uses an object classifier to detect trained regions and rapidly relocalise the camera whenever the tracking is lost. The algorithm not only provides the camera relocalisation capability as required, but it is also computationally efficient which ensures real time performance.

There are three assumptions that have been made. It is assumed that the target environment contains some planar structures. This is a reasonable simplification of the problem, since many applications of robots are in urban or rural environments and man-made

structures tend to have planar surfaces. The intrinsic camera parameters are assumed to be known and accurately calibrated beforehand. It is also assumed that a user can mark some features through mouse clicks (needed to initiate tracking, see Section 6.2.1).

The markerless AR algorithm is presented by first explaining how the camera pose can be obtained by tracking planar features in the scene, followed by a description of a tracking failure recovery strategy which relies on a planar object detection algorithm.

6.2.1 Feature Tracking for Camera Pose Estimation

Four co-planar points from the scene are required to be tracked in the markerless AR algorithm. The users can manually choose four 2D points on the input image that correspond to a planar region in the 3D world. Interest points are extracted from the input positions and tracked using the Kanade-Lucas-Tomasi (KLT) feature tracker [263, 237]; the tracked co-planar points will be referred to as KLT points. The point selection process is performed by the user online, and it does not require the camera to be stationary. Tracking of each feature point will immediately start as soon as it is selected.

The next task is to compute the world coordinates of these KLT points so that the 2D-3D correspondence problem can be solved to obtain the camera pose. This can be achieved by mapping the four KLT points selected on the 2D image plane to a rectangle on a planar surface in the 3D world using a method presented by Simon *et al.* in [240]. The idea is that if we assign the world coordinates of the four image points to be $(0,0),(1,0),(1,s),(0,s)$ (note that the rectangle lies on Z=0), then the problem is simplified to solving for the unknown aspect ratio, s, of the rectangle. First, it is necessary to know that the relationship between a set of image points, x_i, and their world coordinates, X_w, can be described by a planar homography H:

$$x_i = HX_w$$

and H itself is a projection matrix which can be decomposed to:

$$H = K[r_1\ r_2\ t]$$

where K is the intrinsic camera calibration matrix, r_1 and r_2 are the two columns of the rotation matrix, and t is the translation vector. For now, assume $s = 1$ in the calculation, and so the relationship between the 3D world points of the unit square and the actual 3D world points of the rectangle becomes:

$$X_w = \begin{pmatrix} 1 & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix} X_{w\_unit}$$

The mapping of the 2D image points to the 3D world points becomes:

$$x_i = K[r_1\ r_2\ t] \begin{pmatrix} 1 & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix} X_{w\_unit}$$

therefore:

$$H = K[r_1\ sr_2\ t]$$

Given that the intrinsic camera parameters are available, s can be solved using the equation:

$$s = \frac{||K^{-1}h_2||}{||K^{-1}h_1||}$$

where h_1 and h_2 are the two columns of H. Solving s provides the ratio of the rectangle, and we can then use this information to determine the 3D world coordinates of the rectangle.

Now that the 3D world coordinates of the KLT points are known, the camera translation and rotation parameters can be computed by solving the 2D-3D correspondence problem using the Robust Planar Pose (RPP) estimation algorithm [235]. The algorithm presents a robust solution to camera pose estimation from coplanar points. It addresses the pose ambiguity problem in pose estimation tasks by acknowledging that two local minima commonly exist in a pose estimation error function, and tries to determine the best pose with the lowest error. The RPP estimation algorithm is also the one used in the ARToolKitPlus library and has been demonstrated to provide robust pose estimation results [271].

Once the camera pose has been computed, AR can commence to project virtual objects onto the view of the camera. However, it operates in a space with ambiguous scale. It is then necessary to upgrade the results to a Euclidean space that corresponds to the scale of the real world in order to overlay virtual objects, such as range sensor readings, onto the scene with correct real world dimensions. To do this, the length of an edge of the planar region to be tracked is pre-measured for determining the real world scale.

6.2.2 Planar Object Detection

To recover from tracking failure, the proposed approach is to re-detect the previously selected planar region and resume tracking. The detection algorithm used is a fast, scale and perspective invariant planar object detector named "Ferns" [200]. Ferns provides real time recognition of previously selected (trained) objects. During tracking failures, the goal is to use Ferns to detect the planar region bounded by the four co-planar points and use this information to reinitiate tracking of these KLT points and resume AR.

The Ferns detector treats object detection as a classification task. Ferns takes a Semi-Naive Bayesian classification approach for recognising features from input images. The goal is to select the best match for a given image patch during the detection phase. A training phase is required before the online detection process and is typically performed offline. Training the classifier requires as little as a single frontal view of the planar target. Keypoints are first extracted from the target planar object and image patches around the keypoints are warped based on random affine deformations to generate many possible appearances of the planar object as seen from different viewpoints. The set of all possible appearances of the image patch surrounding a keypoint is known as a *class*. During the detection phase, the task is to assign an image patch extracted from the input image to the most likely class.

To recognise a new image patch, a large number of binary image tests, divided into groups called *ferns*, are performed on the image patch, and their outputs are combined to calculate the probability of the given image patch belonging to one of the previously trained classes. Not all input image patches need to be matched to those extracted during the training phase for a successful classification. Given a sufficient number of matches with high confidence, the planar object can be detected. This strengthens the system against partial occlusions.

The Ferns detector is integrated in the markerless AR algorithm and operates in a background thread. Immediately after the four KLT points are specified, the input image of the scene is stored. The rectangle formed by the KLT points defines the region of interest for training using Ferns. Once the training is complete, planar object detection begins and is used to correct and recover any KLT points that are lost during tracking.

6.2.3 Algorithm Overview

To put the algorithm into perspective, the flow of operations is described as follows. To initialise the AR system, the four co-planar points selected by the user are tracked using KLT. The region formed by the four KLT points is stored and used to train a planar object classifier in the second thread using Ferns. While training takes places, AR commences using the camera pose computed from the KLT points. As soon as the training is complete, the Ferns detector executes periodically to correct the positions of the tracked KLT points.

At any time a KLT point is lost, the algorithm checks to see whether the corresponding vertex of the planar region detected by the Ferns detector is still within the view of the camera. The KLT point is automatically reselected if the vertex is visible; otherwise, the vertices of the detected planar region are used as substitutes for the computation of the camera pose to allow AR visualisation to continue. Figure 6.1 gives a flow chart of the overall process.

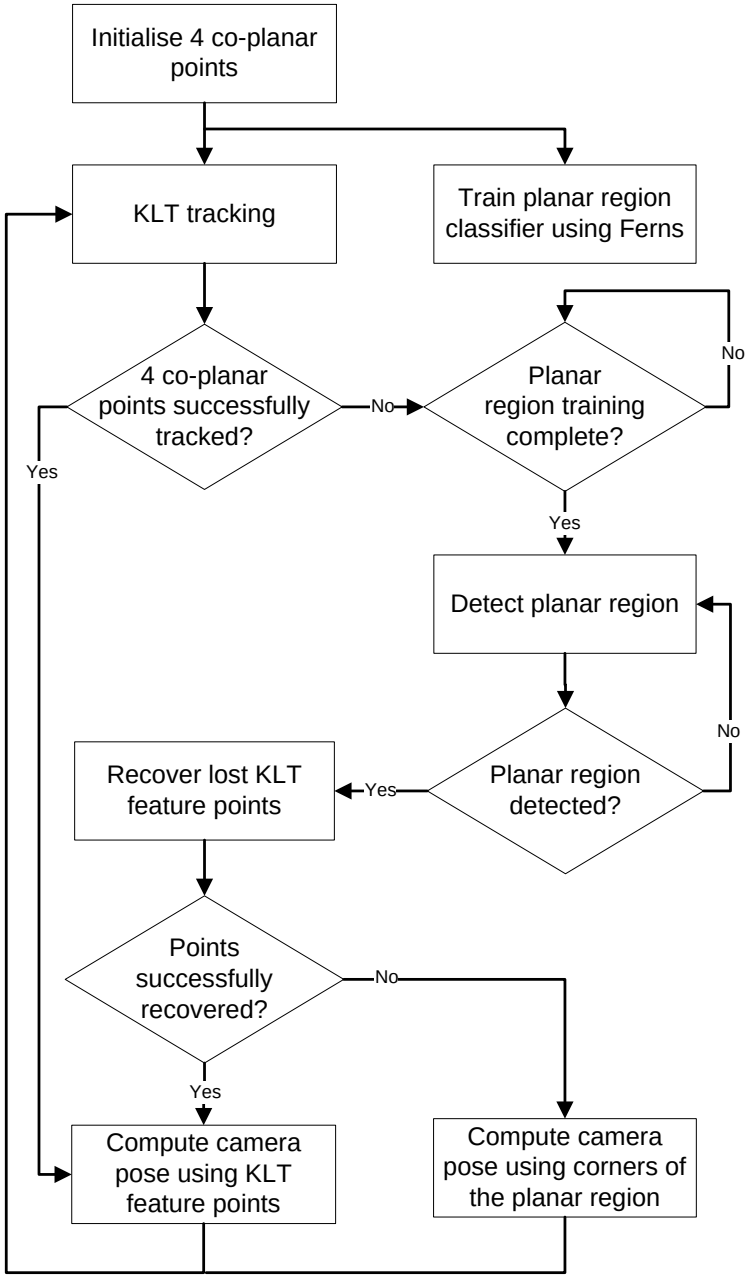


Figure 6.1: Flow chart of the markerless tracking and detection algorithm

The algorithm separates tracking and detection into two threads that execute in parallel. This allows augmentation of virtual objects to proceed over smooth live video sequences while expensive training operations take place in the background. Real time AR is therefore achieved.

6.2.4 AR Results

First, an experiment was conducted to evaluate the markerless AR tracking algorithm using a free-moving handheld camera. The evaluation investigated the algorithm's ability

Figure 6.2: The handheld camera sequence used in evaluating the markerless tracking algorithm. Two virtual objects are overlaid over the flat target. This illustrates tracking under partial target occlusion and different viewing distances, as well as recoveries from tracking failures.

to track the camera pose while undergoing different viewing distances and angles, and partial occlusions of the planar target. It also assessed the tracking failure recovery strategy. The system was deployed in Linux on a 2.4GHz Intel(R) Core(TM)2 Quad CPU with 1GB of RAM and a NVIDIA Quadro FX 3450/4000 SDI graphics card. A Logitech Quickcam Fusion camera was used to capture live video images in resolution of 640x480.

The video sequence consists of 2721 frames for a duration of 30.22 seconds. Screenshots of the experiment are shown in Figure 6.2. The video sequence, `ARConf1Recovery.mp4`, is enclosed on the accompanying CD-ROM. At the beginning of this video sequence, the planar surface of a book was seen tracked for registering virtual objects in the scene. The camera then moved closer to the planar target until it became partially occluded. During this period, KLT tracking became unsuccessful, however, the algorithm was able to continue providing AR visualisation by switching to use the Ferns detector for computing the camera pose.

The video sequence also demonstrates that the system was able to recover from tracking failures. There were three instances when tracking was lost. Figure 6.3 shows the camera trajectory over time which helps to identify these three instances. The first tracking failure was due to a rapid motion just before the camera closed in on the planar target (frame 385 to 391, 0.02s). The second was due to the planar target leaving the camera view (frame 725 to 923, 0.72s), and the third was due to a severe camera shake (frame 1266 to 1602, 1.19s). In all three instances, the algorithm recovered the system from tracking failures as soon as the Ferns detector succeeded in detecting the planar target and recovering the lost KLT points.

The processing time of the two main algorithm components, KLT tracking and Ferns object detection, for this video sequence is shown in Figure 6.4. The figure illustrates that the two components (running in two separate threads) do not operate at every frame. This

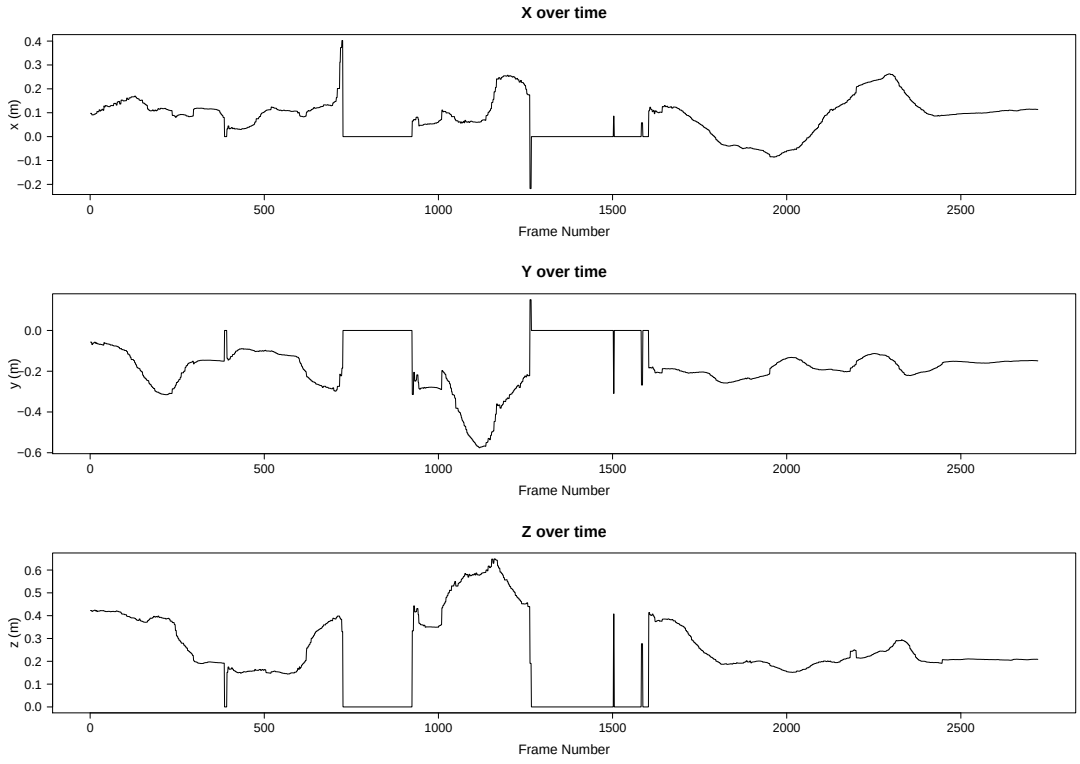


Figure 6.3: Camera trajectory over time for the handheld camera video sequence.

was primarily done to improve efficiency of the implementation. In this experiment, KLT tracking was specified to run at 30Hz. During KLT tracking failures, the Ferns detector also operated at 30Hz to recover the lost KLT points. It was observed that continuous attempts to reinitialise KLT tracking incurred slightly higher computational costs. The process can be improved in the future by implementing heuristic checks to assess the quality of the Ferns detection result before attempting to recover the lost KLT points. Figure 6.4 also shows that the Ferns detector executed periodically to correct potential tracking errors at an interval of 3 seconds. An instance when this was more apparent can be seen in the z over time graph in Figure 6.3 where a small jump in the z value was visible at frame 2199. Overall, the algorithm operated in real time ($>$ 30Hz).

It is important to note that the planar object training process is expensive and can take a considerable amount of time (up to 1 minute) depending on the size of training samples. However, this is a one-off overhead and the process is transparent to the user since it has no negative influence on the quality of the AR visualisation provided by the main KLT thread. A problem is that if KLT points are lost during this time, visual augmentation of virtual objects is temporarily lost until the training is complete. The users can choose to save the trained classifier data to skip the training phase in later experiments with the same planar object.

Another experiment was carried out to demonstrate a remote operation scenario where

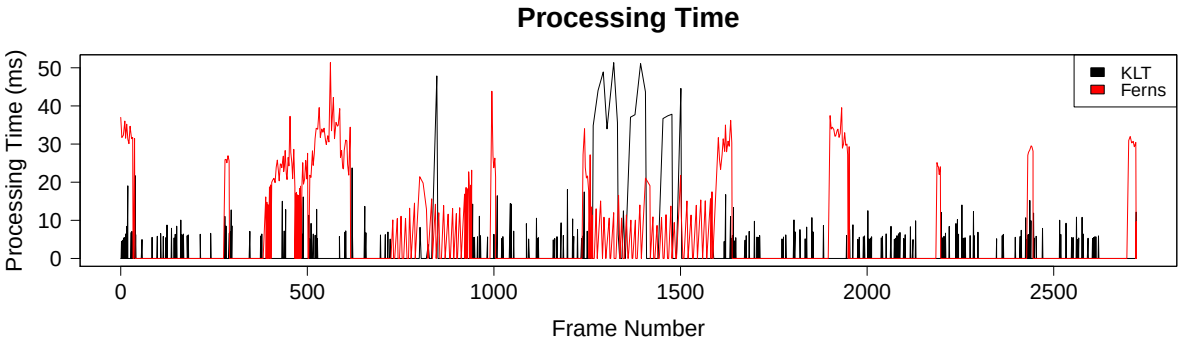


Figure 6.4: Processing time of KLT tracking and Ferns detection.

Figure 6.5: Pioneer 3DX robot with a camera mounted on the robot arm

AR visualisation is created over the live video feed from a camera mounted on a Pioneer 3DX robot, see Figure 6.5. Figure 6.6 shows the components in the system. Compressed JPEG image data from the camera sensor is transmitted through Player to the local machine over wireless LAN. The image data is then decompressed and used for tracking and planar object detection. Once the camera pose is computed, visual augmentation can take place by rendering the scene from the estimated camera viewpoint using the OGRE graphics rendering engine.

The robot was manually controlled to move within a radius of 2 metres of the tracked target object. Various virtual objects were augmented around the planar target. Both the camera and the robot were moved around manually to observe the target environment from different viewpoints, see Figure 6.7. Due to an ongoing wireless network issue (out of the author's control), the frame rate of video transmission was limited to 9-10 Hz. However, this helped to evaluate the performance of the algorithm under low frame rates.

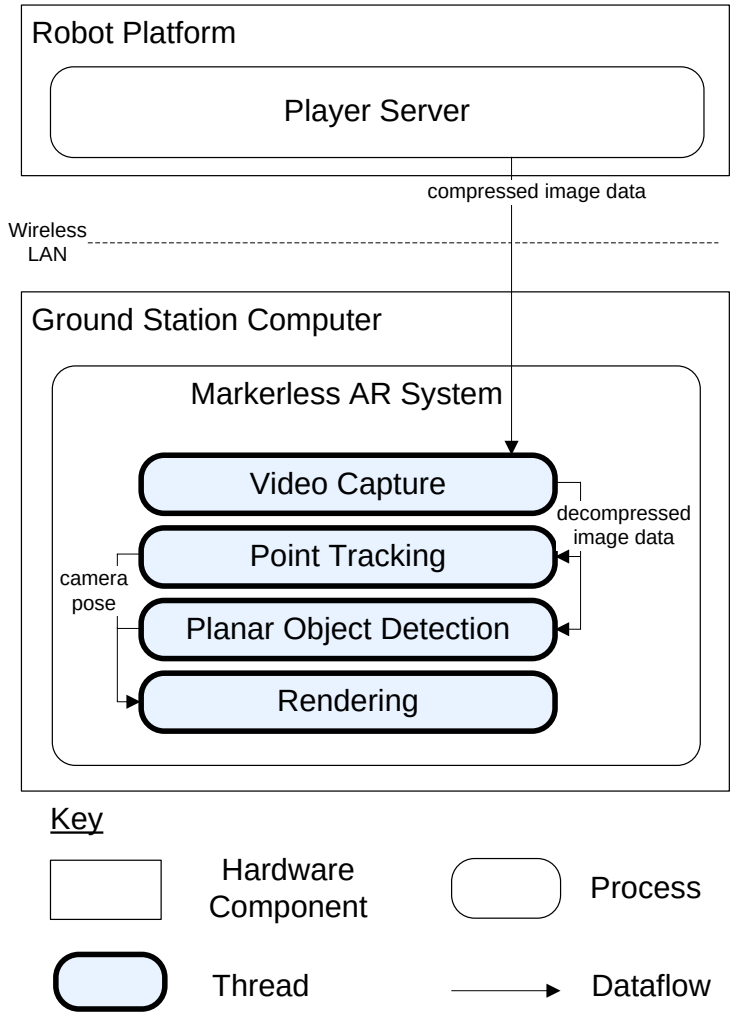


Figure 6.6: Dataflow between components in the remote operation experiment. The diagram also shows the multi-threaded AR system design

Visual results indicate successful tracking of the planar target within the specified working area and the virtual objects remained accurately registered, with an error between the real and the estimated camera positions measured to be approximately 0.012 metres within a range of 2 metres from the planar target. As the robot moved around the target environment, the planar target constantly exited the camera view. The Ferns detector was able to detect the planar target when it reappeared and automatically resumed KLT tracking, e.g. Figure 6.7(c), 6.7(d). Partial occlusion of the planar target was also acceptable with small jitters in the visual augmentation, e.g. Figure 6.7(f).

There are a couple of situations for which this markerless AR configuration can fail. This happens when both tracking and detection of the planar region are unsuccessful. The first situation occurs during the training phase of the detection algorithm as mentioned before. The second occurs when the camera observes the target environment from long distances or at small acute angles between the viewpoint and the planar surface.

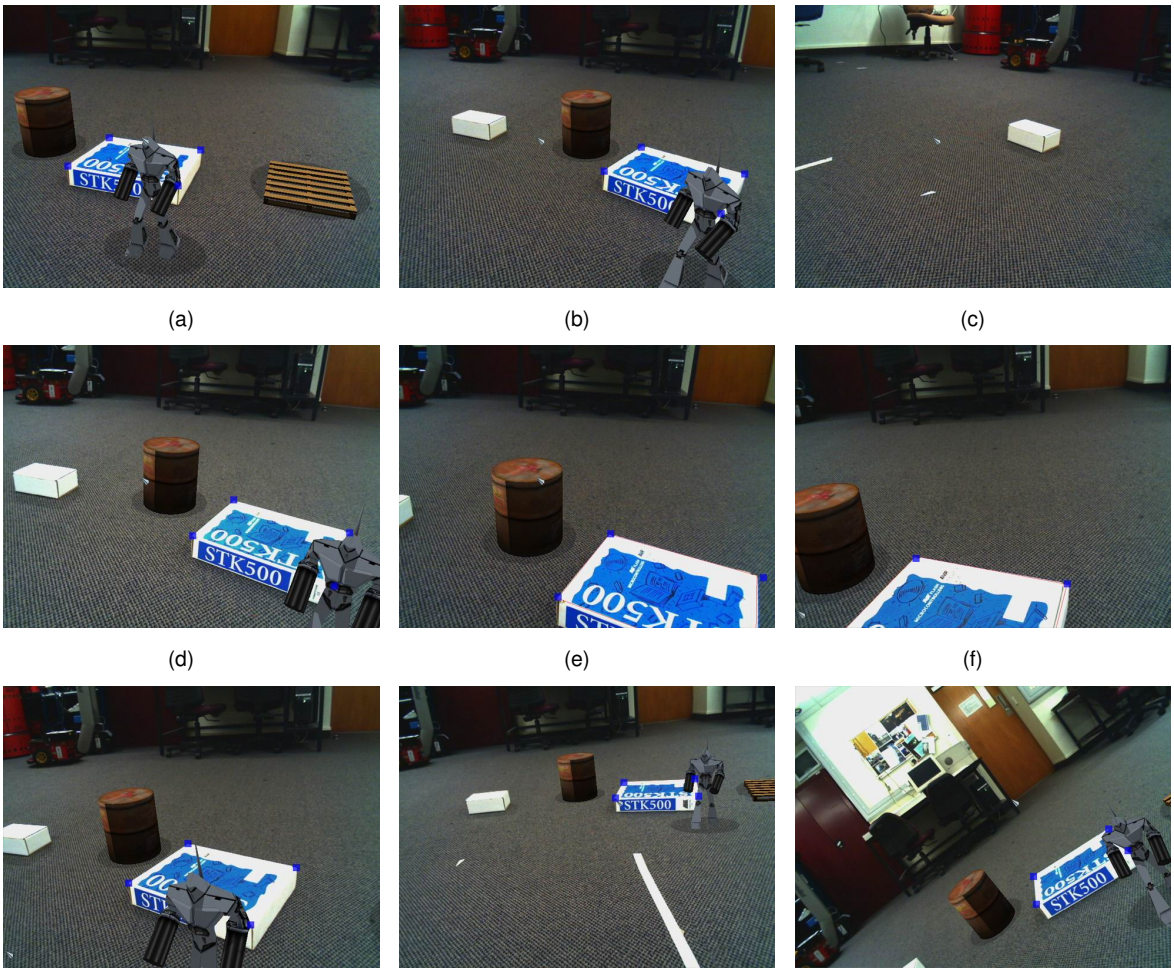


Figure 6.7: A barrel, an animated character, and a wooden pallet are introduced into the scene. (a) Four points (in blue) on a planar surface of a box are manually selected and AR rendering begins while the training takes place. (b) Robot turning away from the planar target. (c) Planar target exits the camera view and augmentation is lost, (d) Planar target immediately detected when reappeared and KLT points are recovered. (e) & (f) Planar target partially occluded and AR rendering continues using the estimated positions of the planar region corners. (g) KLT points are again recovered and tracking is resumed. (h) Camera viewing the planar target from a distance. (i) Camera viewing the planar target under different rotations.

Erroneous classification results are produced by the Ferns detector, and in some cases, the target object can not be detected. Visual augmentation is lost if KLT tracking also fails during this period.

6.3 Configuration Two: Extensible AR

Extensible tracking is a class of AR techniques that aims to extend the operating area of the AR system by allowing the system to continuously explore and add previously unknown scenes to its database, e.g. an initial map or model. Extensible tracking can greatly benefit the application of AR in robotics as many robot tasks take place in large and potentially unknown environments where building a map or a model of the environment is time-consuming or impossible.

This section presents an extensible AR solution that builds on existing work in the area of SfM for AR. The markerless AR algorithm described in the previous section is integrated with a real time parallel mapping and tracking algorithm to continuously construct a map of the environment and perform registration as the camera moves to unknown scenes.

6.3.1 Parallel Tracking and Mapping

The extensible AR solution builds on the Parallel Tracking and Mapping (PTAM) algorithm proposed by Klein and Murray [148]. PTAM is one of the state-of-art AR systems with robust, real time performance. This SfM based algorithm has many similarities with visual SLAM, both track the pose of the camera while constructing a map of the environment. PTAM tackles the problem using principles derived from photogrammetry. It performs 3D scene reconstruction by analysing the projective geometry between a set of images and applies bundle adjustment to estimate and refine the positions of map elements and the pose of the viewpoints.

There are a number of features of PTAM that distinguish it from other SfM based AR systems. First, PTAM exploits the idea of keyframes for mapping. While other SfM based AR systems build their maps using all image frames or a sliding window of most recent image frames, which can potentially deteriorate the system frame rate or map quality, PTAM builds its map from only heuristically chosen image frames that contribute to the estimates. These carefully selected image frames are known as keyframes. Image frames that are assessed as poor quality, e.g. motion blurred, are discarded. Second, the real time performance of PTAM comes from the separation of tracking and mapping in two separate threads. It is proposed that expensive but effective bundle adjustment optimisation does not need to be applied within real time limits, but rather, offsetting

it to a background thread is sufficient for refining and maintaining a sparse map of the environment. Third, tracking in PTAM is based on a large number (up to thousands) of FAST point features [226] and a motion model, which provide a level of robustness comparable to the use of computationally expensive feature descriptors.

The idea of keyframe based mapping, multi-threaded design, and motion model tracking had previously been explored separately in the literature, but the combined use of these features in PTAM makes it a successful implementation that offers superior performance over many other AR systems.

6.3.2 Map Scale and Coordinate Frame Initialisation

The original PTAM operates in an arbitrary scale space predominantly determined by the map initialisation process. The process requires the user to manually specify two keyframes to triangulate the base map. This is performed by pointing the camera at a roughly planar scene and pressing a key on the keyboard to acquire the first keyframe, then translating the camera horizontally by an offset to acquire the second keyframe through another key press. Salient points are tracked over the translation and the correspondences are used to triangulate the base map. This setup assumes the first keyframe has identity rotation and zero translation, while the second keyframe is assumed to have been taken at a pre-specified offset from the first. Bundle adjustment is applied to refine the positions of the map points and the pose of the keyframes. The best fit plane is then calculated using RANSAC [107] and the whole map is transformed so that the plane lies at $z = 0$. Once the map has been successfully initialised, tracking and mapping commence in parallel. The remainder of the PTAM algorithm is illustrated in Figure 6.8.

The original map initialisation approach is unsuitable for creating AR visualisation in robot tasks for two reasons. First, robot tasks often require the map to be constructed in the real world metric scale. However, in the original approach, the map is initialised in an arbitrary scale dependent on the user input (moving the camera and capturing the first two keyframes). Second, the original initialisation process creates an unpredictable location of the map origin, which makes aligning AR visualisations with the robot environment difficult. The map origin is dependent on the features seen at the time of map initialisation, locating itself at the center of the map points that are used to calculate the best fit plane. While this can be corrected by manually aligning the PTAM coordinate system with a user specified origin or the world origin used by the robot system, the process is time-consuming and can be inaccurate. Therefore, before PTAM can be applied for AR visualisation in robot tasks, modification to the map initialisation process is necessary.

The solution is to use the markerless AR algorithm described in the previous section for PTAM map initialisation. To set up the base map, the new approach associates the two

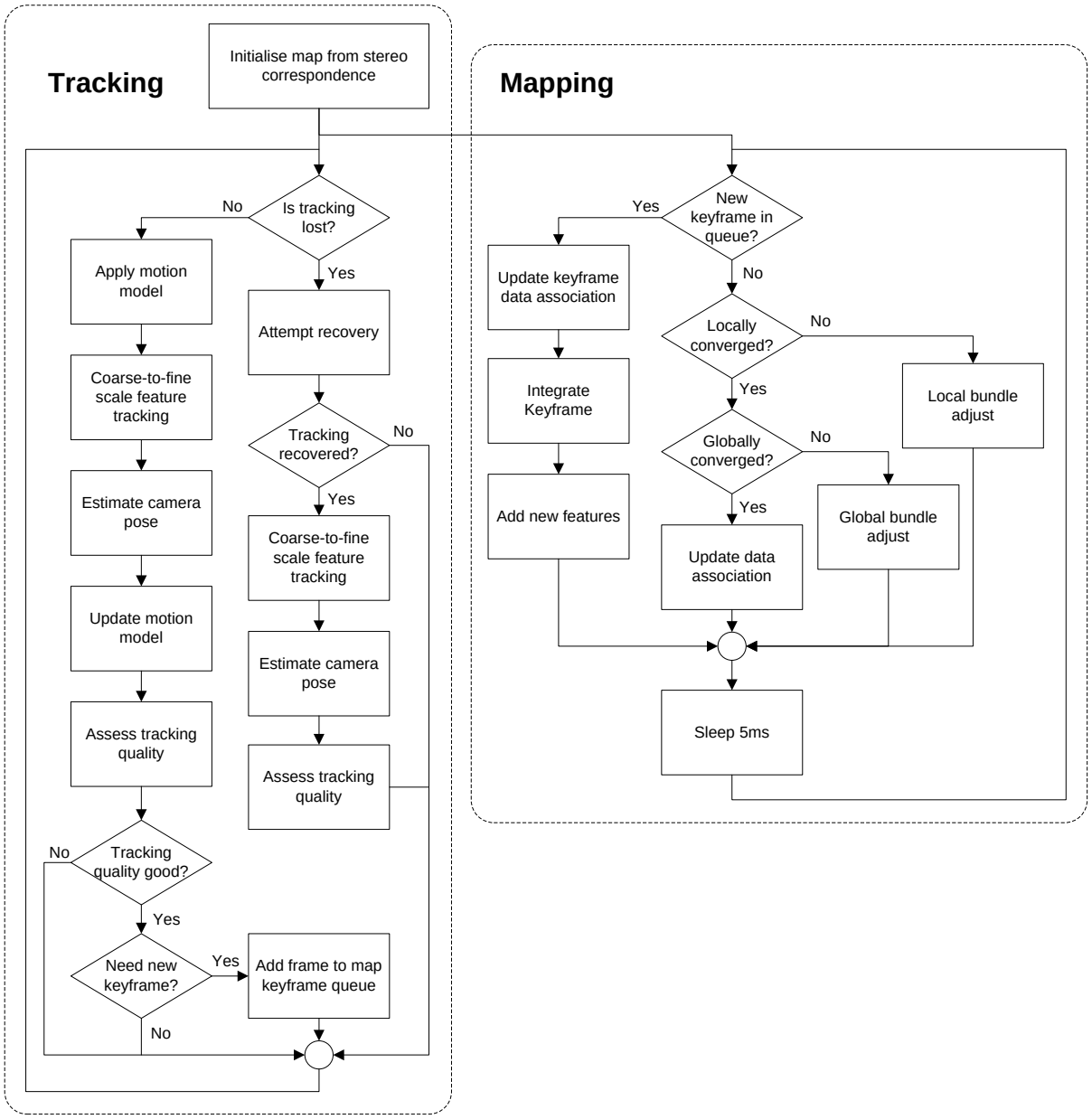


Figure 6.8: Flow chart of the PTAM algorithm. Tracking and mapping are divided into two threads. The mapping thread begins as soon as the base map is initialised, and builds the map using keyframes added by the tracking thread. (Part of this diagram is adapted from [148])

initial keyframes with pose estimates from the markerless AR algorithm, which allows the map to be constructed in the metric space and coordinate system defined by the markerless AR algorithm. Moreover, instead of letting the user specify the two keyframes for map construction, these keyframes are automatically selected by the system in the new map initialisation process described as follows. When the system is started, the markerless AR algorithm commences as usual and begins tracking and detecting planar regions specified by the user. As soon as the markerless AR algorithm produces the first pose estimate, the corresponding image frame is stored as the first keyframe for PTAM. This keyframe

is assigned the pose estimate produced by the markerless AR algorithm. The system then monitors the translational movement of the camera by looking at the output of the markerless AR algorithm. Once the translation exceeds a heuristically chosen amount, the second keyframe is stored and assigned its pose estimate. These two keyframes are then fed into PTAM to triangulate the base map. Since the pose estimates produced by the markerless AR algorithm are in the real world metric space, this guarantees that the map created from these two keyframes is also in the same scale space, subject to pose estimation error. Similar to the original map initialisation process, bundle adjustment is then applied to refine the map. However, the plane identification and map transformation step is no longer necessary since the triangulated map is based on keyframes with poses already defined by the markerless AR algorithm's coordinate system.

In summary, the steps are listed below:

1. Initialise planar region tracking and detection (configuration one)
2. Store first keyframe with first pose estimate
3. Monitor for translational camera movement
4. Store second keyframe with pose when movement exceeds threshold
5. Trianguate PTAM map using the two keyframes
6. Apply global bundle adjustment to map

6.3.3 AR Results

An evaluation of the extensible AR tracking configuration was carried out using live video input from a handheld camera. In comparison to the previous experiment, this system uses wired transmission of camera images instead of a wireless one. The reason for this is that PTAM requires a steady real time video feed of preferably 25Hz or more to achieve equivalent performance results as demonstrated in [148]. The video frame rate over wireless LAN was found to be low (9-10Hz) in our lab setup as previously observed in the experiment conducted for evaluating configuration one, and preliminary tests found that this significantly degrades PTAM's tracking performance. This is a shortcoming of PTAM that needs to be addressed in the future. For this experiment, a Point Grey Firefly MV FFMV-03M2C camera equipped with a 4mm lens was used and 640x480 images were transferred over a firewire cable to the ground station where computations were carried out.

The experiment involved the camera exploring a large region of a laboratory. The extensible AR system was initialised using the new method described in Secion 6.3.2. A

Figure 6.9: The PTAM map of the video sequence consists of a total of 112 keyframes, and a number of them are shown here to illustrate the operating environment. The top two rows of keyframes show the camera moving away from the initial desktop region to explore along the right wall. The bottom two rows of keyframes show the camera exploring along the left wall and eventually seeing the back wall.

planar target on a desktop is tracked and used to initialise the scale and the coordinate system of PTAM. Once the base map had been constructed, the camera started to explore unknown scenes and add keyframes to the map. The camera first moved away from the desktop and explored along the wall to its right. It then navigated back to the desktop region and continued to explore along the left wall. The camera then returned to the desktop at the end. The video sequence for this experiment, `ARConf2Results.mp4`, is enclosed on the accompanying CD-ROM.

The video sequence consists of a total of 4837 frames and a duration of 167.56 seconds. The PTAM base map was initialised at frame 62 (1.51 seconds into the operation). The final PTAM map consists of 111 keyframes and 6311 points. Figure 6.9 illustrates a subset of keyframes, and Figure 6.10 shows the constructed map in the real world metric scale.

To show that the proposed initialisation method has successfully set up PTAM's coordinate system, the AR drawing mode provided by the PTAM system was turned on after exploring the left wall. The camera returned to the desktop region to see a virtual object registered at the map origin. Figure 6.11 shows a sequence of AR registrations and demonstrates that the virtual object correctly centers over the bottom left corner of the

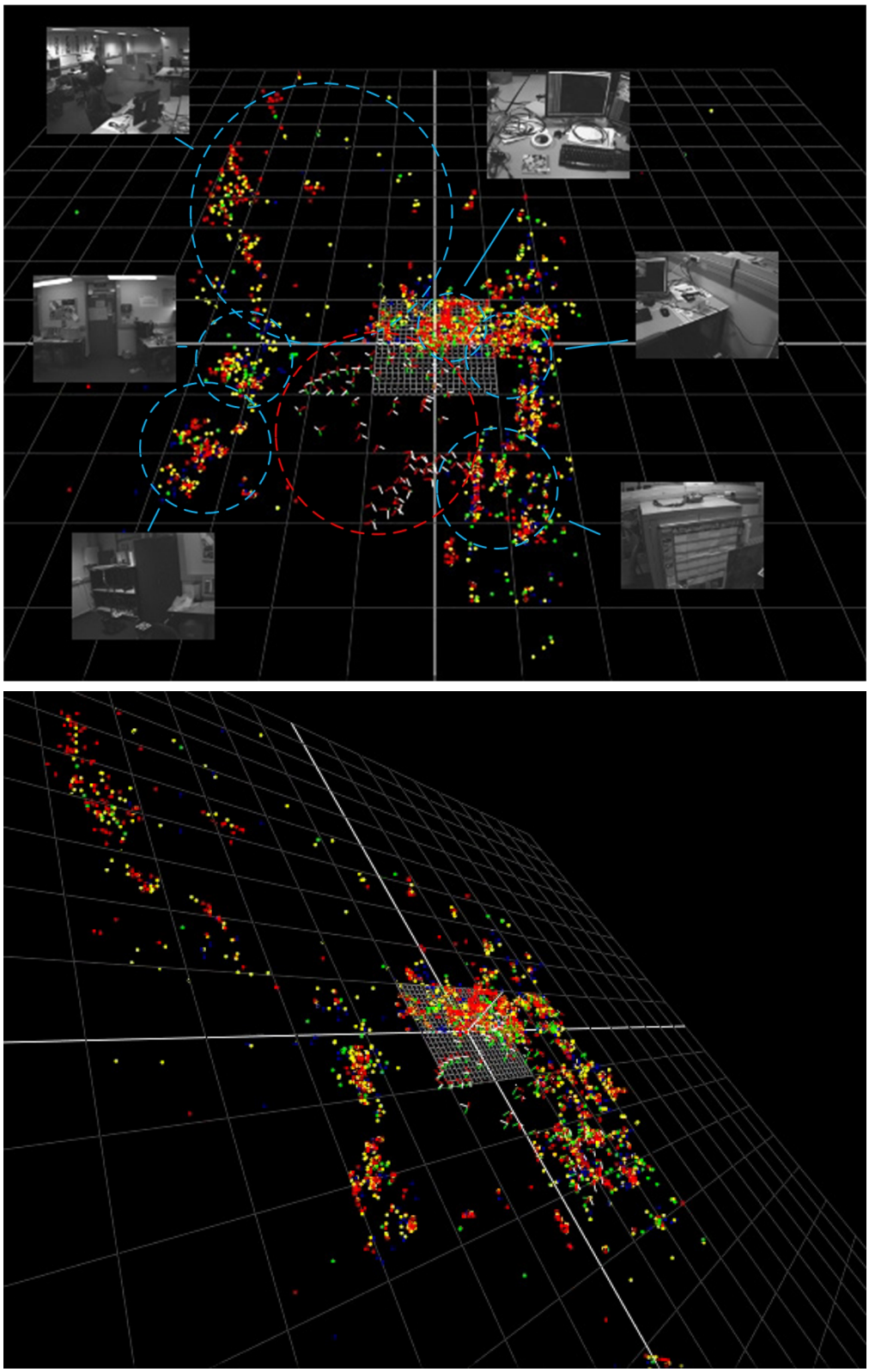


Figure 6.10: The final PTAM map constructed in the real world metric scale. The keyframes (bounded by the red circle in the top figure) are represented by three orthogonal axes in red, green, and white colours. The cluttered points in the map origin correspond to the desktop region. Other scenes have also been added in the top figure to illustrate the region of the environment which the points correspond to (see blue bounding circles).

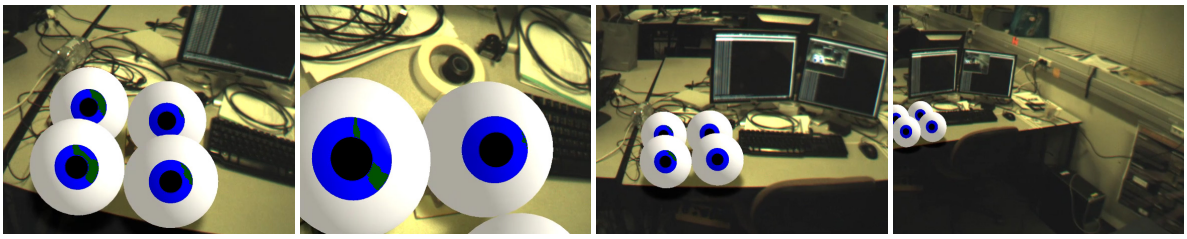


Figure 6.11: A sequence of screenshots illustrating a virtual object (the default PTAM AR object) registered at the planar target that was used to initialise the map.

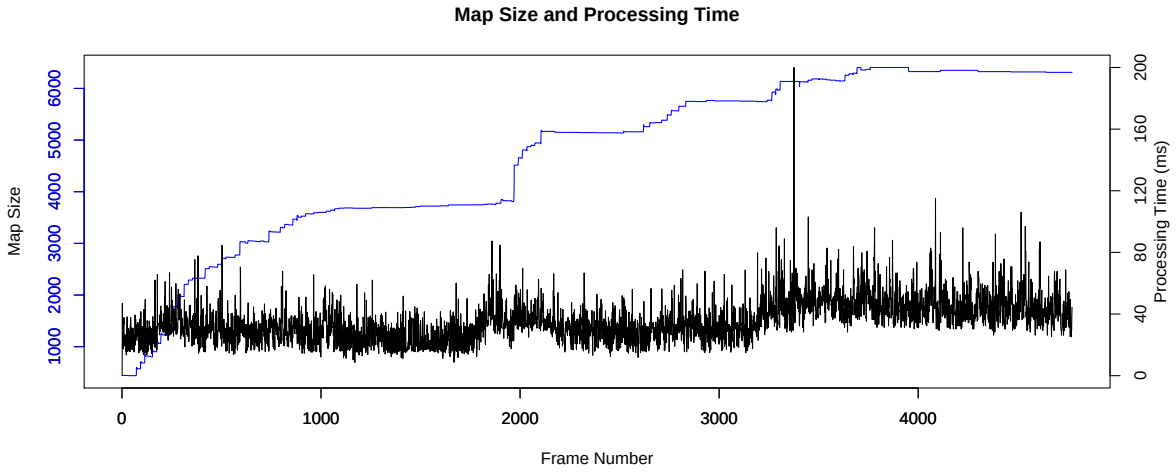


Figure 6.12: Evolution of the PTAM map size and processing time for the laboratory video sequence.

planar target where the origin is defined. A slight offset between the actual origin and the PTAM map origin was observed, and this is due to errors in the markerless AR pose estimates when setting up the keyframes for constructing the base map. The offset was measured to be approximately 0.01m in x, 0.02m in y, and 0.01m in z.

The system was able to run in real time, averaging 29 frames per second despite the increase in map size. Figure 6.12 illustrates the map size and the processing time for this video sequence. The map size increased steadily as new keyframes and points were added to the map. It can also be observed that the map size reduced by a small amount from time to time. This is the result of the PTAM algorithm removing bad points from the map based on heuristic quality checks. The spike in the processing time at frame 3379 is caused by the activation of the AR drawing mode.

An analysis of the tracking results reveals a short period of inaccurate tracking. It occurred as the camera was exploring along the left wall. As it moved to an unknown scene, new keyframes with very few map points were added. As the result, there were insufficient points in the scene to provide accurate pose estimation. PTAM then produced outputs that are far from the actual positions. Figure 6.13 shows the camera trajectory

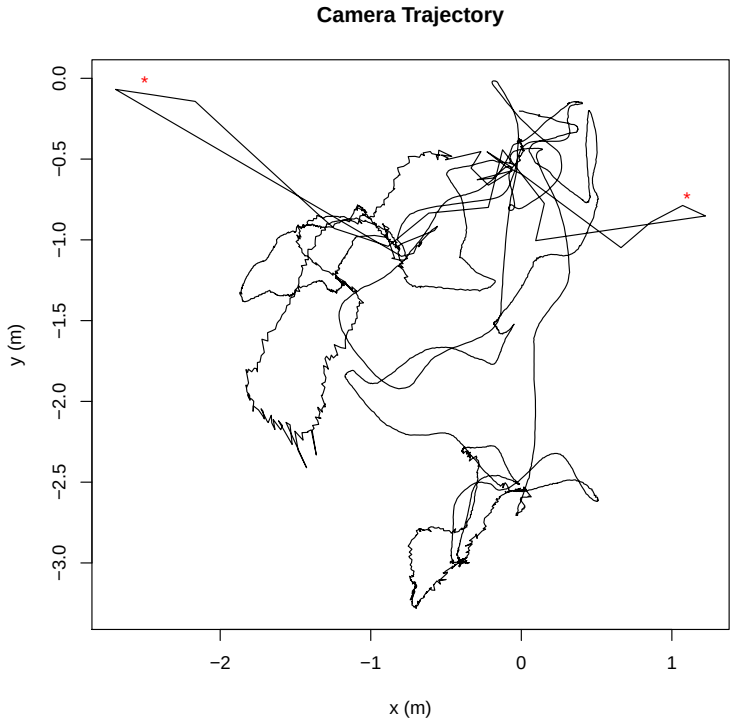


Figure 6.13: Camera trajectory based on PTAM pose output. The red asterisks (∗) mark instances of inaccurate pose outputs.

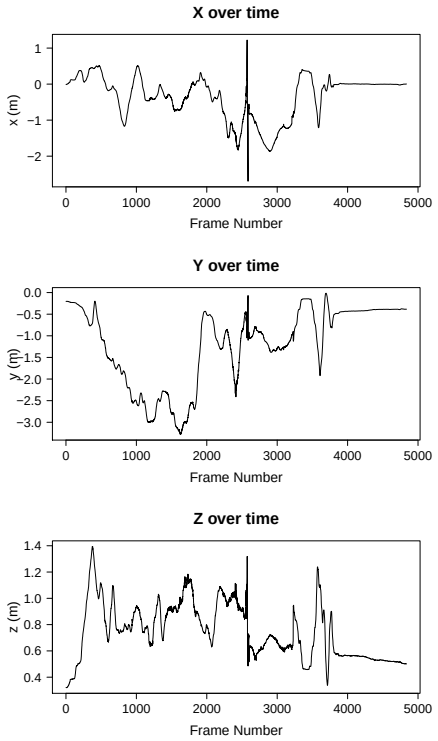


Figure 6.14: Camera trajectory over time. Erratic behaviour observed at frame 2571.

plotted based on PTAM pose outputs. The graph highlights a number of pose estimates that are clearly outliers from the smooth camera trajectory. The camera trajectory over time shown in Figure 6.14 also highlights the period of inaccurate tracking. The error is most apparent in the x over time graph which shows sudden changes in x values. The erratic behaviour of PTAM occurs from frame 2571 to 2587 and lasts for a period of 0.516 seconds.

Apart from the map initialisation process, the rest of the PTAM algorithm used in this AR configuration remains the same as the original implementation. Thus, the tracking performance after the map initialisation process were observed to be similar to the results reported in [148].

Overall, this AR configuration demonstrated robust and accurate tracking for the majority of this live video sequence. The proposed solution for map initialisation was able to construct a map in the real world scale and a predefined coordinate frame without harming PTAM's performance. However, it must be noted that while there are small movements of people in this video sequence (see row three of Figure 6.9), the majority of the environment is static. Robust AR tracking in dynamic environments remains a

challenge as found in the body of existing work, and this is an issue that needs to be addressed in the future in order to provide more effective AR visualisations for robot tasks that take place in largely dynamic environments.

6.4 Summary

Driven by the requirements of applying AR in robotics, two AR configurations were proposed. The first configuration is a markerless AR algorithm that combines tracking and detection of planar regions for camera pose estimation. The idea can be considered as an extension of traditional marker based AR systems, such as ARToolKit, where the marker is replaced by a textured planar region which naturally exists in the robot's operating environment. Point features corresponding to the vertices of the planar region are tracked over consecutive frames, while a planar region detector operates in the background thread to constantly correct the tracked points and also recover the system from tracking failures. The proposed markerless algorithm is simple but useful as it provides a good balance between speed and accuracy. It is capable of operating at low frame rates, and the tracking failure recovery capability greatly increases the operating lifetime of the system. An important note is that the manual feature selection process of this algorithm can be considered a limitation of the system. Future work should remove this burden from the user and provide an automatic tracking initialisation method. The use of this configuration has another limitation primarily when used in robot tasks that require AR visualisations in multiple different scenes. This requires multiple planar regions to be tracked. The markerless AR system needs to harvest new planar regions online but the process takes time due to the training time required for each planar region detector. Thus, this configuration should be used in applications that do not require the camera to constantly or rapidly explore unknown scenes.

The second configuration creates AR visualisations based on extensible tracking. It overcomes the limitation of the first configuration by allowing the AR system to continue providing registration as the camera moves to previously unknown scenes. The algorithm is based on a modified version of Klein and Murray's PTAM system. In essence, the algorithm takes an SfM approach to viewpoint tracking while constructing a map of the environment. The markerless AR algorithm in configuration one is integrated with PTAM and used for the map initialisation process to construct the map in the real world metric space with a predefined coordinate system. Once the base map is initialised, the algorithm then continues as usual. The algorithm incrementally grows the map and applies bundle adjustment for map refinement. Separation of mapping from tracking in two threads enables real time tracking performance. The result is high quality AR registration that was only previously possible in offline computer vision systems. This AR configuration

Table 6.1: Comparison between AR configuration One and AR configuration Two.

	Configuration One	**Configuration Two**
Tracking Technique	Point & Planar Feature	Structure from Motion
Operating Environment Size	Limited	Extensible
Occlusion Robustness	Partial Target Occlusion	Partial Scene Occlusion
Memory Requirement	Fixed	Proportional to Map Size
Video Feed Requirement	>9Hz	>25Hz
Algorithm Frame Rate	>30Hz	∼29Hz

is not without its shortcomings. The main limitation when applied in robotics is the need for the translational movement in map initialisation. In certain AR setups, the movement is difficult to achieve, e.g. AR based on an egocentric camera mounted on a non-holonomic robot vehicle. A possible workaround for this problem is to pre-build the map prior to starting the robot task. Another limitation is the increase in memory and computational requirement for the *mapping* thread as the map grows. Map maintenance becomes an expensive task, and global bundle adjustment fails to converge within the allowed time [148]. Nonetheless, this can be solved by maintaining multiple sub-maps [63] or limiting global bundle adjustment to a sliding window of nearest keyframes.

Table 6.1 summarises the two AR configuration properties. The frame rates are based on the system specification given in Section 6.2.4. The choice of which AR configuration to use depends on the task and the operating environment. Overall, the first AR configuration is more suited to monitoring tasks in a more controlled environment due to the limited operating area and its susceptibility to occlusions of the planar target. An example application of using this AR configuration is in manufacturing industries where AR can be created for providing visual feedback on robot manipulation operations which are typically limited to a small area. The approach would also be ideal for creating AR visualisations for debugging tasks in robot development environments similar to those in [79, 192] where experimentation of robot operations, such as simple navigation tasks, can be monitored from an overhead camera. On the other hand, the second AR configuration allows the operating area to be extensible, therefore is more suitable than the first AR configuration for mobile robot tasks. This includes exploration operations in potentially unknown environments, provided that large memory and processing budgets are available. There is, however, the problem of deploying the extensible AR system in applications such as telerobotics because the underlying PTAM algorithm used requires a high video frame rate (or smaller inter-frame camera displacements) which may not always be possible in field tests due to network latency issues.

The next chapter presents a number of case studies for using the MR robot simulator

and shows how the AR configurations are used to provide AR visualisations in each different application.

7
Evaluation

Thorough evaluation of a simulation tool is important to extrapolate its use in practice. In the area of virtual simulations, USARSim [62] is a robot simulator that has undergone extensive validation. Efforts have been put into assessing the accuracy of its sensor, actuator, and environment models. However, similar efforts have not yet been applied to evaluation of MR/hybrid environments designed for robot development.

This chapter presents an evaluation of the MR robot simulator through a series of case studies. Case study evaluations are particularly useful as they help to assess the simulator's ability in solving real world problems commonly faced during the robot development process. The MR robot simulator is also evaluated in terms of its ability to meet the requirements and address issues presented in Section 3.2. In summary, there are three main issues which the solution, MR simulation, must address:

- First, the solution needs to improve the reliability of simulation results. To evaluate this, this chapter presents comparative evaluations that were conducted to compare the MR simulation results with those from other simulation methods as well as the real experiment. This helps to obtain a measure of the MR simulation accuracy with respect to a complete real world test, and assists in determining whether there are any improvements compared to existing simulation approaches.

- Second, the solution must be reusable. The case study evaluations effectively address this by analysing the use of the MR robot simulator for the development of three robot systems in three different applications.

- Lastly, safety must be ensured in high risk and expensive experiments. This is a more difficult aspect to evaluate by using quantitative performance tests. Therefore, the case study evaluations include user studies that were conducted to evaluate whether the MR simulation and its visual interfaces could help address safety issues in real world projects.

It is also considered important to evaluate the user acceptance of the MR robot simulator to identify whether the tool is perceived to be useful for robot development. The user studies conducted in the case study evaluations were also designed to measure the user's experience in using the MR robot simulator and the provided interfaces, and to identify whether there are preferences in using MR simulation for carrying out certain robot development tasks over the use of virtual simulation.

Other issues to consider include evaluating whether using MR simulations would benefit robot development in terms of cost and efficiency. The evaluation in this research provides a glimpse of these aspects by illustrating designs of MR simulations that address these issues (particularly in case study two), as well as collecting subjective opinions using questionnaires in two user studies. However, a more thorough evaluation is required in future work to verify these findings. This could potentially involve comparing development costs and durations between a condition in which the MR robot simulator is used and another condition in which the MR robot simulator is not available.

Three case study evaluations are presented in this chapter. The first case study evaluates the MR robot simulator as a whole through a search and rescue scenario, see Section 7.1. The second case study analyses the use of MR robot simulator for creating test scenarios during the development of an industrial robot system, see Section 7.2. The last case study analyses the use of the MR robot simulator for developing components of a UAV system that is to be deployed for agricultural tasks, see Section 7.3.

7.1 Case Study One: Search and Rescue Scenario

This case study presents a simple MR simulation of a search and rescue scenario to assess the functions of the implemented system as a whole, including the markerless AR component and the MR interfaces.

The scenario requires a robot to navigate in a simulated hazardous environment to search for a specified target. It provides a good study for evaluating the MR robot simulator because it involves a certain degree of risk and thus is a representative scenario which could benefit from the use of MR simulation. In real robot exploration tasks, such as robot search and rescue, robots maneuver in unknown environments while exposed to various threats. Most often, extensive testing and experimentation in highly controlled

environments and the use of expensive resources are required. MR simulation aims to relieve some of these requirements by using virtual simulated components.

The evaluation is divided into two parts. The first provides a thorough evaluation of the MR robot simulator for simulating the navigation component of the robot search and rescue system, in this case, the same obstacle avoidance algorithm described in Section 5.5.1 is used. Quantitative results are presented on the MR simulation accuracy in comparison to a virtual simulator. The second part of the evaluation presents an analysis of the MR simulation for the search and rescue task, describing a number of limitations identified during the process.

7.1.1 Obstacle Avoidance Evaluation

The evaluation begins by measuring the accuracy of MR simulations results. The main component being tested in simulation is the laser based obstacle avoidance algorithm. This algorithm is the main navigation component that will later be used in the robot search and rescue operation to avoid obstacles in a hazardous environment.

An experiment was conducted to quantitatively compare the behaviour of the robot in virtual simulation, MR simulation, and a real experiment. The objective is to determine whether the MR robot simulator was able produce realistic simulations that reliably represent the real experiment in comparison to the virtual robot simulator Gazebo.

While one might argue that it is obvious for MR simulations to yield more accurate results since more real components are incorporated into the setup, this is not necessary true. Without strong coherency and synchronisation mechanisms, such HIL-based simulations could generate unnatural behaviour and produce incorrect results that are less reliable than other forms of simulation. Thus, a validation of the implemented system is important.

The experimental setup uses the same layout as the sensor based interaction example described in 5.5.1. The experiment involved a Pioneer 3DX robot equipped with a URG laser rangefinder avoiding three cylindrical obstacles in a 2m $\times$ 2m area. The cylindrical obstacles each have a radius of 0.06m. The same obstacle avoidance algorithm is run under the three different conditions (virtual simulation, MR simulation, real experiment). In virtual simulation, the 3D robot simulator Gazebo is used. A simulated URG laser model with the same configurations as the real one is created and mounted on the default Pioneer robot model available in Gazebo. In the MR simulation, the robot is real while the three cylindrical objects are virtual. The robot is modelled using a position entity, and the attached position behaviour fuses camera pose estimation output with the robot's odometry to deduce the robot pose, which is then used to keep its virtual representation in a consistent state.

The evaluation analysed the trajectories taken by the robot in the three conditions

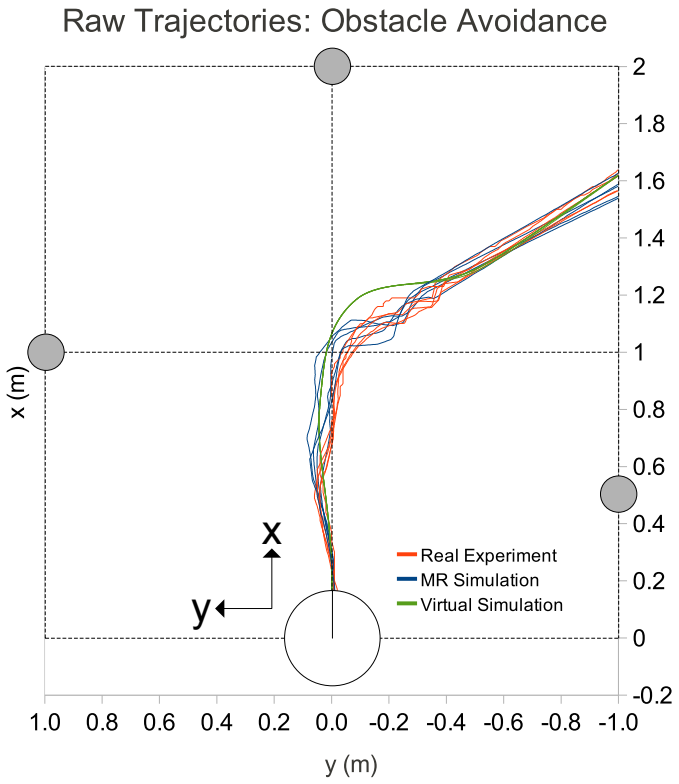


Figure 7.1: Raw trajectories taken by the robot while avoiding three cylindrical obstacles (gray circles).

over five runs. The quantitative validation uses an error measure based on the path difference between the robot in simulation and the one in the real experiment [40]. In virtual simulation, the trajectory of the robot can be obtained by requesting Gazebo for the absolute world position and orientation of the robot over time. To obtain the ground truth of the trajectories of the real robot in MR simulation and the real experiment, an overhead camera was used to track an ARToolKitPlus marker attached on top of the robot. This provided pose information of the robot over time with an error of less than 0.01m, which was considered acceptable for the task.

7.1.2 Obstacle Avoidance Results

The raw trajectories taken by the robot in the three conditions are plotted in Figure 7.1. In all five runs of the MR simulation, interactions between the real robot and the virtual obstacles were successfully created, resulting in robot behaviours that were very similar to the ones in the real experiment. On the other hand, the robot in the virtual simulation also took on a similar path to avoid the obstacles, but there were very little variations between the trajectories as can be seen from the graph in Figure 7.1. The average trajectories are plotted in Figure 7.2 to provide a clearer comparison between the results from the three conditions. One thousand evenly spaced sample points were taken from each of the five

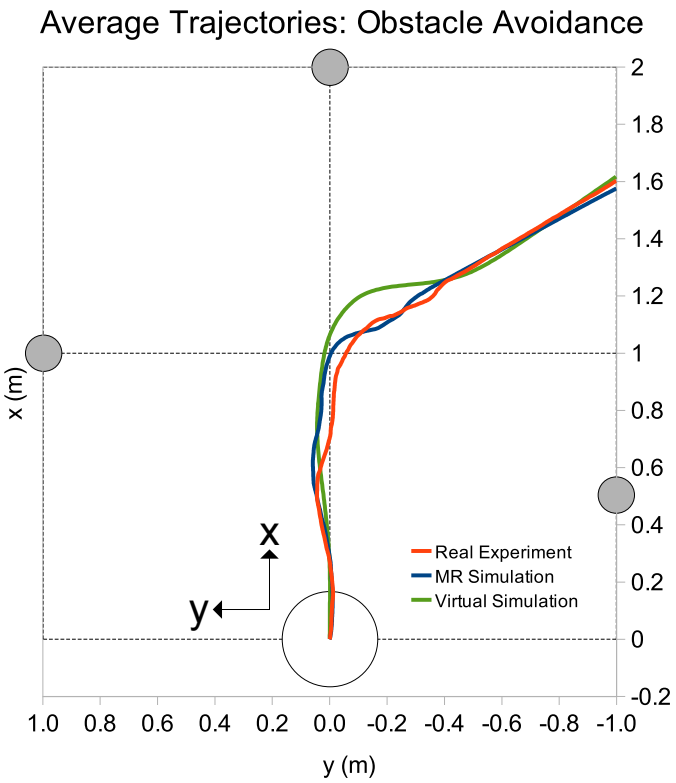


Figure 7.2: Average trajectories taken by the robot while avoiding three cylindrical obstacles (gray circles).

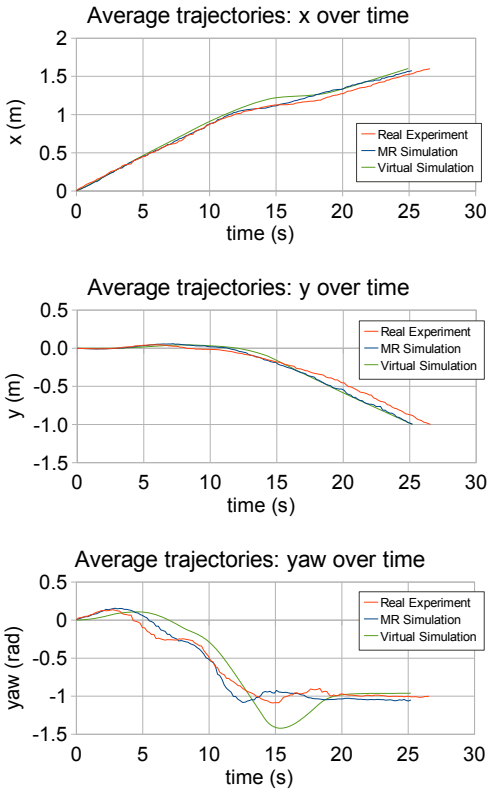


Figure 7.3: Average trajectories of the robot over time

trajectories to calculate the averages. The main differences occur at around the 1m mark in x when the robot decided to turn to its right to avoid the obstacle in front. The real robot in the MR simulation and the real experiment made a sharper turn in comparison to the virtual robot. This indicates discrepancies between Gazebo's actuator model for simulating the differential drive capability of the Pioneer robot and the actual real world robot.

To measure the similarity between the trajectories produced in the MR simulation and those produced in the real experiment, all the raw trajectories from one set were compared with all trajectories from the other, i.e. yielding 5*5 comparisons. Each comparison is made by measuring the Euclidean distance between the corresponding sample points, then calculating the Root Mean Square Deviation (RMSD). The average RMSD for all comparisons was then calculated to give an indication of similarity. The results show the MR simulation produced robot trajectories with small deviations from the ones in the real experiment, with an average RMSD of 0.02m in x and 0.03m in y. This is approximately 1.9% with respect to the average total distances travelled by robot, which are 2.17m in the MR simulation and 2.16m in the real experiment. Comparing with the virtual simulation, which has slightly higher average RMSDs of 0.03m in x and 0.04 in y, the MR simulation

was found to yield results closer to the real experiment.

The robot's average trajectories over time are shown in Figure 7.3. The x and y motion over time remained similar between the three datasets, but it can be seen that the robot's yaw motion in Gazebo is inconsistent with the results from the real experiment. Moreover, Gazebo was again found to produce minimal variations between the trajectories in terms of time. This is the result of missing noise models. Similarly, friction and slippage between the wheel and the ground were not yet supported. These factors are the common cause of discrepancies of a simulator's results with the real world. While improvements could be made to Gazebo, the MR robot simulator provided an alternative solution which enabled real components to be incorporated in simulation, and this relieved the need to simulate complex real world variations. The solution proved to be viable as illustrated by the results.

This simple evaluation validated the MR robot simulator's accuracy and showed its improved reliability in comparison to Gazebo. However, improvements to Gazebo's simulation models are still important. The MR robot simulator is built on Gazebo, and therefore, MR simulations involving virtual robots (navigating in real environments for example) would also suffer from the same limitations as the underlying simulation platform. Implementing more accurate and realistic sensor and actuator simulation models would benefit both Gazebo and the MR robot simulator.

7.1.3 Search and Rescue Experiment

The first part provided a quantitative evaluation of the MR simulation accuracy for a small sub-task of the overall mission. In many cases, robot developers may be more interested in testing their robot system's performance in accomplishing the overall goal than the performance of an individual component. The second part of the evaluation now attempts to assess the MR robot simulator for simulating the entire robot search and rescue operation.

The final robot search and rescue system integrates the obstacle avoidance component with a vision based object detection component. The robot relies on vision to identify the target object while navigating in the environment. The robot should slowly approach the target object when found. In this simulation, the target object is represented by an ARToolKitPlus marker placed in a laboratory environment. The MR environment consists of virtual hazards that are potential threats to the real robot. These include fire, a barrel, a small wood pallet, as well as another search and rescue robot moving in the environment. Real objects, such as boxes of different sizes, are also placed in the MR environment to represent obstacles. Figure 7.4 shows the layout of the simulation environment.

In the experiment, the robot's onboard camera images were sent through Player to

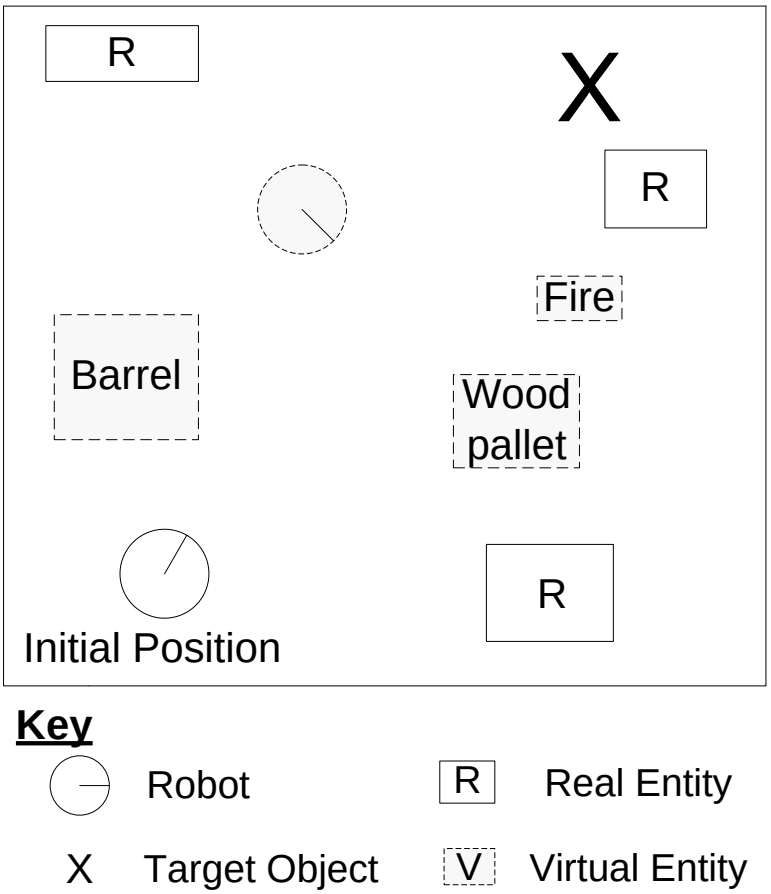


Figure 7.4: Layout of the MR environment for simulation of a robot search operation.

the client program located on another computer where the object detection algorithm was run. The provided AR interface corresponded to the same onboard camera imagery. The reason for using an onboard camera was to simulate a remote operation which required users to monitor the robot's actions through the interfaces. AR visualisation was created using AR configuration one (see Section 6.2). A planar object on the back wall was tracked to compute the camera pose for registering the virtual objects into the real world. Once the tracking was initialised, the client program would then connect to the MR robot simulator to begin simulation. Screenshots from the experiment are shown in Figure 7.5.

7.1.4 Observations

Informal exploratory trials were carried out to identify potential limitations of MR simulation. The trials involved the author, who was also the main software developer in this project, testing the search and rescue robot system using the MR robot simulator. Subjective observations made by the author are described as follows. As expected, interactions between the real robot and the virtual objects were successful and the robot navigated in the environment while avoiding real and virtual obstacles detected by the laser sensor.

(a) (b)

(c) (d)

Figure 7.5: Screenshots of the robot search and rescue MR simulation. (a) An AV view of the MR simulation environment, (b) Switching to the AR interface; AR is initialised by tracking four feature points (in blue) corresponding to the four corners of the notice board on the back wall; robot starts moving while avoiding real and virtual obstacles, (c) & (d) Robot slowly approaches the target object in the corner; despite partial occlusion of the tracked planar region in (d), AR continues with small jitter.

The robot eventually found the target as it moved closer, and slowly approached the target to complete its task.

The use of MR simulation helped to provide previews of different potential causes of damage to the robot if it were deployed in a completely real experiment, particularly collisions with the small wood pallet which could not be detected by the laser sensor because it rested below the height of the horizontal 2D laser scan. In both MR interfaces, the real time laser visual overlays provided an understanding of the robot's perception, which helped to identify the reason for collisions. The visual overlays also highlighted a number of close encounters between the real robot and the moving virtual robot.

Introduction of virtual objects in a real physical environment allowed rich simulation of resources, which some of these objects can also be very difficult to emulate or recreate in real world experiments, e.g. smoke produced from fire. In one experiment, the smoke actually occluded the target from the view of the robot, which caused the robot to turn away from the target.

The combination of AR and AV views was able to provide visualisation of robot information and simulated objects from different perspectives. However, it was noted that the chosen AR configuration caused short periods of AR discontinuity. Visual augmentation of virtual objects were temporarily lost when the planar object exited the camera view but resumed when it reappeared. The AV interface helped to compensate this weakness by providing a view of the simulation in the virtual environment when AR was lost. However, it was noted that frequent AR discontinuity may place a burden on the simulation user who was then constantly required to manually switch between interfaces. An alternative solution would be to use the extensible AR configuration; this will be investigated in the third case study evaluation.

7.1.5 Summary

The case study presented in this section demonstrated the use of MR simulation for simulating a search and rescue scenario involving a mobile ground robot. A major part of the evaluation focused on validating the MR simulation accuracy against the real experiment based on trajectory results from testing of the robot's navigation component. Quantitative comparisons suggest an improvement in the reliability of its results over the virtual robot simulator Gazebo. The MR simulation was also able to support simulation of the robot search and rescue operation, which helped to provide a preview of possible causes of harm to the robot during the operation. Observations also suggested that, for continuous AR visualisation, the extensible AR configuration could be used in future MR simulations that require tracking continuously changing viewpoints in potentially large environments.

This case study evaluation provided observational results in the view of the author.

These findings will be validated using user studies in the remaining two case study evaluations.

7.2 Case Study Two: Screw Remover Robot

In the field of industrial robotics, simulations are often used to reduce the development cost and improve efficiency. However, the value of using simulations would only be appreciated if the benefit is evident and exceeds the expense of investing in creating and setting up the simulations. To examine the use of the MR robot simulator for industrial applications, the simulator is deployed to aid the development of a robot manipulator system for a building interior demolition and renovation task [75]. The goal is to investigate whether the use of MR simulations can help the testing process of the robot system by minimising resource requirements while providing accurate simulations.

This case study also evaluates other aspects of MR simulations which the previous case study did not cover. First, in comparison to the search and rescue MR simulation presented in the previous section where it mainly focused on simulating a single type of sensor based interaction, the MR simulation that will be presented in this section involves multiple types of interactions necessary for the robot to complete its task. The accuracy of the MR simulation resulting from the different MR interactions will be evaluated. Second, equally important is the need for evaluating the visual interfaces of the MR robot simulator. This is performed through a user study. Similar work has been carried out by Nielsen *et al.* [190] who evaluated their 3D AV interface for assisting robot teleoperation tasks by comparing it with a traditional 2D interface. In contrast, this evaluation conducts a user study to compare the AR and AV interfaces provided in the MR robot simulator to identify the strengths and weaknesses of each approach. Finally, another important purpose of this case study was to evaluate the effectiveness of the MR robot simulator outside our laboratory, on a real world project, with a different research group, in a different environment, and on a different software architecture.

The screw remover project is a real commercial project for the National Institute of Advanced Industrial Science and Technology (AIST) in Tsukuba, Japan. The work described in this case study was carried out as a collaboration for the development of a screw removal robot on a funded visit at AIST.

7.2.1 Screw Remover Project

In Japan, building interior renovation commonly requires removing equipment, such as lights and air conditioning vents, mounted on suspended beams from the ceiling. The old ceiling panels attached to the beams by self-tapping screws also need to be removed.

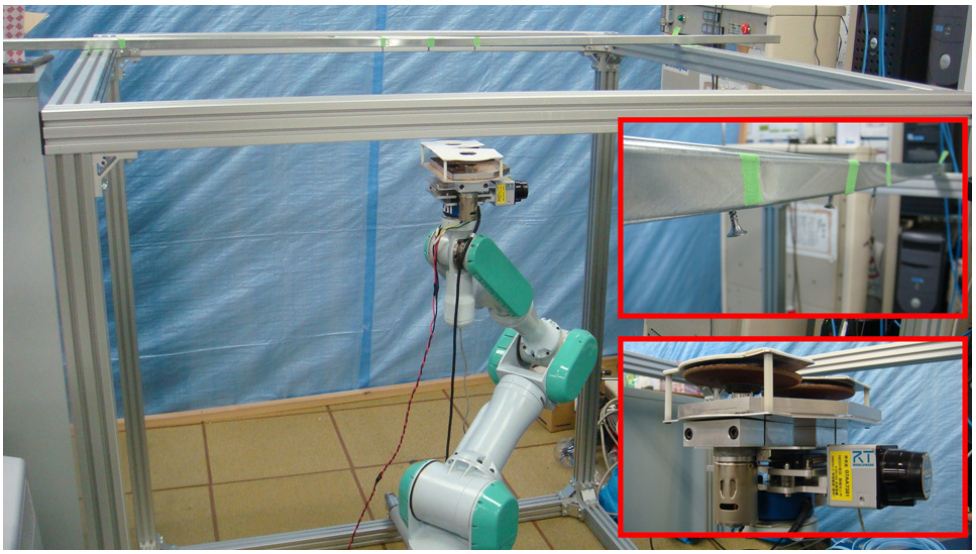


Figure 7.6: A typical screw remover system test setup. Highlighted in red on the right are the ceiling beam with screws, and the custom screw removal tool.

A construction worker must then remove these screws by hand. This is a long and physically demanding process, due to the position of the screws above the worker's head. It is, however, simple and repetitive, making it ideal for automation.

A ceiling beam screw removal robot [48] is proposed for the screw removal step. Rather than cutting or pulling the screws from the beam, which damages both the screws and the beam, the system uses a more sustainable solution, using a custom screw removal tool to unscrew the screw, leaving the materials in a reusable condition.

The robot used is a Mitsubishi PA10 seven DOF industrial robot manipulator. Mounted on the robot's end-effector are a custom screw removal tool, a force-moment sensor, and a laser scanner, see Figure 7.6. The custom screw remover tool consists of two rotating wheels that grip and turn the screw to remove it as the tool moves along the ceiling beam. The force-moment sensor is used to detect contact between the tool and the screws and in keeping the tool pressed against the beam. The laser is used to locate the beam above the robot and align the tool with the beam.

Screw Removal Operations

A diagram illustrating the layout of the laboratory test setup is shown in Figure 7.7. A ceiling beam is mounted on a frame at a fixed height above the ground, and three screws are fitted on the beam at 100mm apart.

To perform the screw removal task, a human operator is required to press buttons on a GUI to start and stop the system. Once the operator starts the system, the robot will autonomously carry out the screw removal process. The operator monitors the robot operations and stops the system when the screws have been removed. The operations are

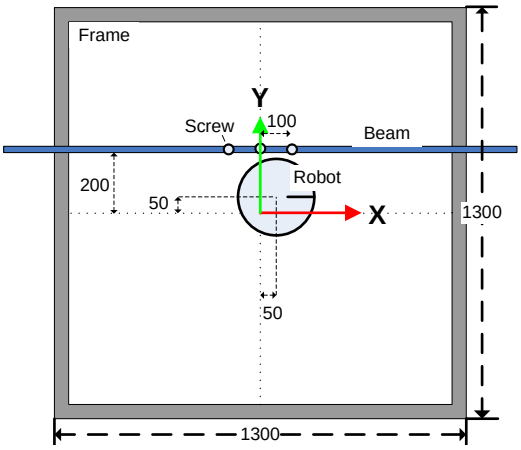


Figure 7.7: Experiment layout (units are in millimeters).

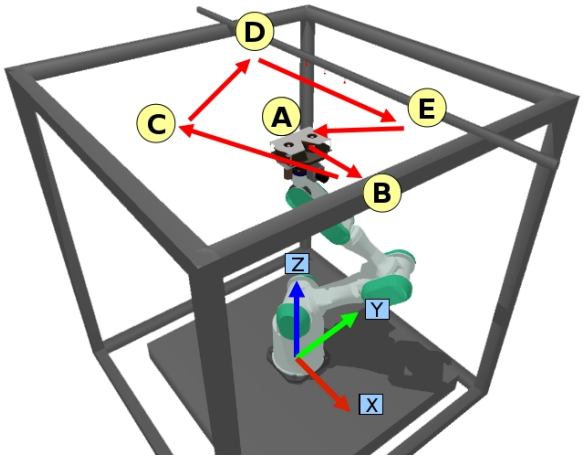


Figure 7.8: Screw removal operations.

shown in Figure 7.8.

- **A to B**: The operator starts the system. The end-effector moves in the x-direction and takes three to five laser scans to locate the beam above.
- **B to C**: The end-effector moves to a position slightly below the beam.
- **C to D**: The end-effector approaches the beam at constant z-velocity until the force-moment sensor detects contact with the beam.
- **D to E**: The end-effector moves along the beam to remove screws.
- **E to A**: The operator stops the removal process and sets the robot to return to the starting position.

7.2.2 Mixed Reality Simulation

MR simulation integrates virtual resources into the real experimental setup to provide a cost-effective solution for experimenting with robot systems. The use of MR simulations can aid in the testing of the screw remover system because:

- there is a reasonable level of risk involved. The PA10 is a powerful robot and can cause harm to the operators and other objects in the surrounding environment during the testing phase of the development. Dangerous or expensive components can be virtualised in MR simulation to ensure safety.

- testing of the system consumes resources such as spare beams and screws, which can be damaged by the robot due to errors in the system. The tool's wheels become worn out over time and need to be replaced. MR simulation is able to minimise the resources consumed by simulating these materials.

In this MR simulation, the main focus is in studying the behaviour of the robot in the real world, making sure the controllers are generating the correct movements before moving onto a completely real world test. Resource requirements also need to be minimised in the testing process. Therefore it has been decided to virtualise the ceiling beam and the screws in the MR simulation. The components in this setup are:

Real: robot manipulator, screw removal tool, laser sensor, force-moment sensor, and frame (for holding the ceiling beam).

Virtual: a ceiling beam fitted with three screws.

It is important that in an MR simulation, real and virtual objects are able to seamlessly interact with one another as if they exist in a coherent environment. To facilitate interaction between the real robot components and the virtual beam and screws, the sensors and actuators of the screw remover system need to be augmented with virtual information. The real laser needs to detect the location of the virtual beam as well as the real frame, while the real force-moment sensor needs to detect contact between the screw removal tool and the virtual beam as well as other external forces in the real world. The motion of the real robot manipulator will also be influenced by friction between the real screw removal tool and the virtual beam, and physical contacts between the screw removal tool and the virtual screws. Interaction between real and virtual components has been created based on the MR framework described in Chapter 4.

Exact simulation of the actual screw removal process (wheels gripping and turning the screw) is considered unnecessary for this application as we are interested in simulation at an integration level. Thus, the screw removal process is simplified at the cost of lower simulation accuracy. In the simulation, once the screw has a force exerted on it exceeding a threshold over a certain amount of time, the screw falls from the beam.

A video, `ScrewRemover.mp4`, illustrating the MR simulation created for this case study is enclosed on the accompanying CD-ROM.

As the original screw remover system runs on OpenRTM [29], the MR simulation created is based on the OpenRTM/Gazebo framework described in Section 5.2.3. Figure 7.9 shows the system diagram of the RT Components in the MR simulation. The screw remover system now exchanges data with MR sensors and an MR robot manipulator, instead of reading data from a real laser or force-moment sensor and sending commands to the real PA10. In summary, the two MR sensors are modelled as a laser entity and

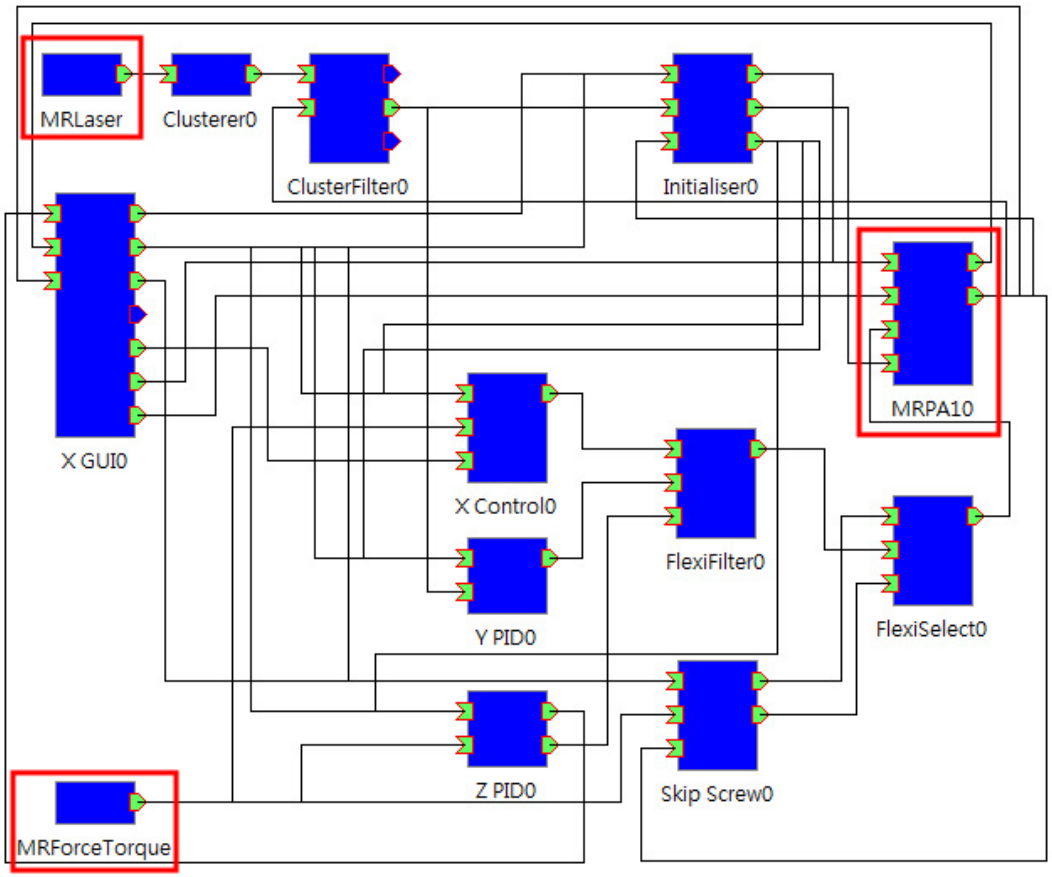
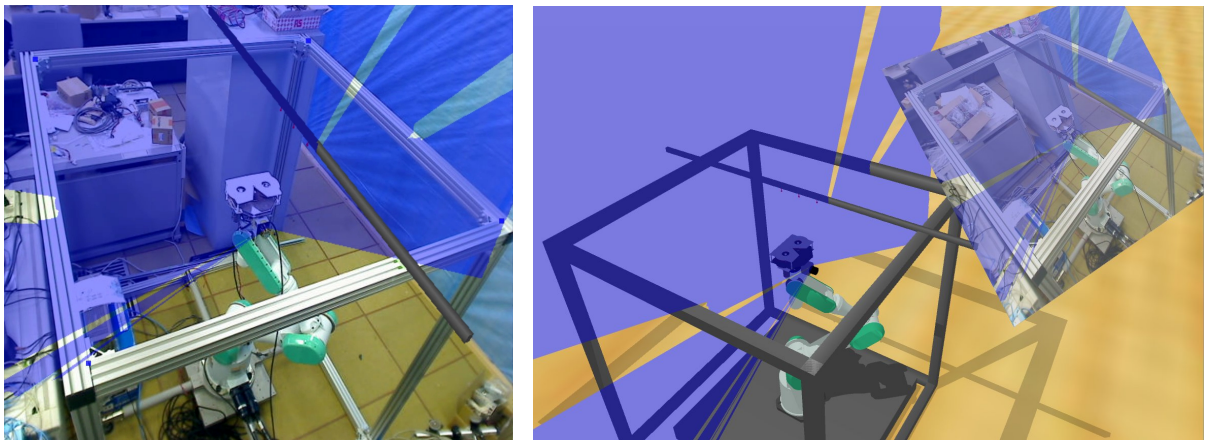


Figure 7.9: System diagram showing the RT components in the MR simulation. The components highlighted in red are MR sensors (MRLaser, and MRForceTorque) and an MR robot manipulator (MRPA10) that exchange data with the original screw remover system. The diagram is taken from the OpenRTM's graphical development tool, RTSystemEditor.

a force-torque entity in the MR simulation. They subscribe to the original (real) sensor components using the device manager in MRSim and modify the raw data with inputs from the virtual world created using Gazebo. The resulting data are then published to the connecting components. The MR robot manipulator is modelled as an actuator array entity. It detects and models any physical contacts with virtual entities then sends commands to move the real PA10 accordingly.

The ability of the MR robot simulator to handle communication with robot systems running on different frameworks eased the process of simulating OpenRTM robot systems. No modifications to the simulation tool or the software system being tested were necessary. This demonstrates that the proposed design of the MR robot simulator contributes to its reusability on a different system architecture, meeting the requirement for a standardised development environment as described in Section 3.2.2. The design of the MR robot simulator also enabled it to be immediately deployed for the screw remover project, which greatly benefited the development process by minimising the large overhead commonly required in building and setting up a simulation system.



(a) (b)

Figure 7.10: Screenshots of the MR interfaces created for the development of the screw remover system. a) The AR interface showing visualisation of laser data, virtual beam, and virtual screws in context with the real world, b) The AV interface showing the simulation in a virtual environment augmented with real world laser and camera data.

MR Interfaces

The AR and the AV interface provided are shown in Figure 7.10. In the AR interface, virtual objects are overlaid in geometric registration onto the real world scene using the markerless AR configuration (configuration one) described in Section 6.2. A fixed overhead camera is used to capture a view of the whole experimentation environment which enables the users to monitor the overall robot manipulator's movement and the screw removal process.

The AV interface provides a view in a virtual environment augmented with real world information, such as sensor data and the robot movements, in real time. A free-look camera is available in the AV interface that can be controlled to move freely within the environment for observing the simulation. Figure 7.11 shows screenshots of the screw removal operation provided by the MR interfaces.

7.2.3 Evaluation

To assess the reliability of the results generated in the MR simulation, an experiment was conducted to compare its results with those from a real experiment. Each experiment was run five times, and the trajectories taken by the end-effector of the robot manipulator recorded. The time for a single screw removal was set to be 6 seconds in the MR simulation. The average trajectories are shown in Figure 7.12. One thousand evenly spaced sample points were taken from each trajectory to calculate the averages. The results show that the average path taken by the robot in the MR simulation closely resembles the one in the real experiment. This shows the robot successfully detected these virtual objects,

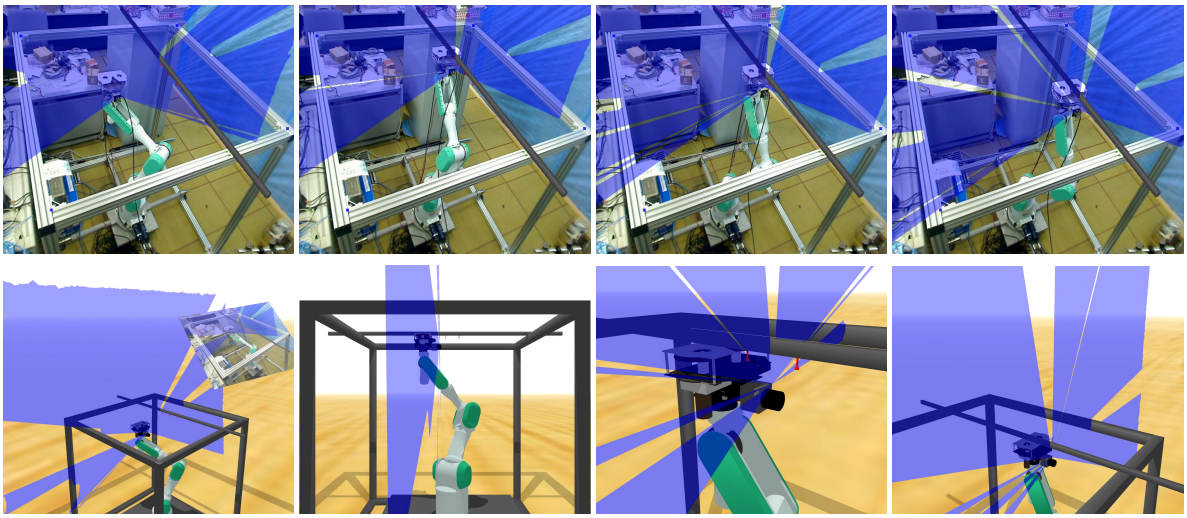


Figure 7.11: A sequence of screenshots illustrating the screw removal operation as seen through the MR interfaces. Top Row: AR interface. Bottom Row: AV interface with a free-look camera.

and the registration of the virtual beam and screws in the real world was sufficiently accurate to produce similar robot behaviours. Note that the deviation in the last segments of the trajectories between the MR simulation and the real experiment were caused by the inconsistency of the human operator when signaling the robot to stop the screw removal operation and return to the starting position.

To quantify the similarity between the two sets of trajectories, the Euclidean distances between all the raw trajectories produced in the MR simulation and those produced in the real experiment were measured. This yields a total of 5*5 RMSD values of which the average was then calculated. The results show an average RMSD of 24mm in x, 11mm in y, and 8mm in z, yielding an average RMSD of 28mm in distance. This is approximately 1.7% with respect to the average total distances travelled by the end-effector, which is 1622mm in MR simulation and 1644mm in the real experiment.

Figure 7.13 shows the trajectories over time for the five runs. As the robot's end-effector moves along the beam in the x-direction to remove screws, a step-like pattern can be observed. The results show that more variations occurred in the real experiment which the MR simulation did not produce. This behaviour was expected due to the simplified screw removal process and the fixed screw removal time. The simplified screw removal simulation resulted in some inconsistencies with the real experiment but was considered acceptable for the task since the focus was on evaluating the overall screw removal strategies, not on the custom made hardware tool. Nevertheless, future work could introduce noise to the simulation of the screw removal process to mimic real world variations if desired. Table 7.1 shows the average times for removing a single screw, and for completing the entire task. The average time for removing a single screw in the real experiment is shorter than predicted, resulting in a short task completion time compared

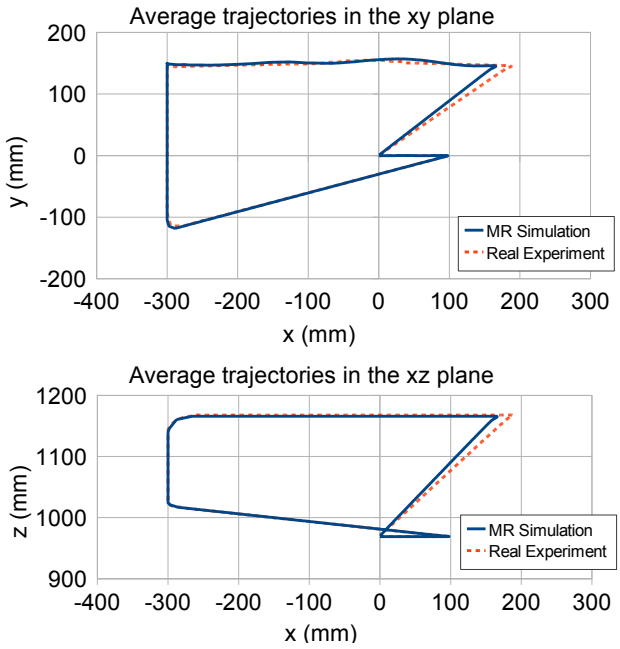


Figure 7.12: Comparison between the average trajectories of the robot's end-effector in *xy*- and *xz*-plane.

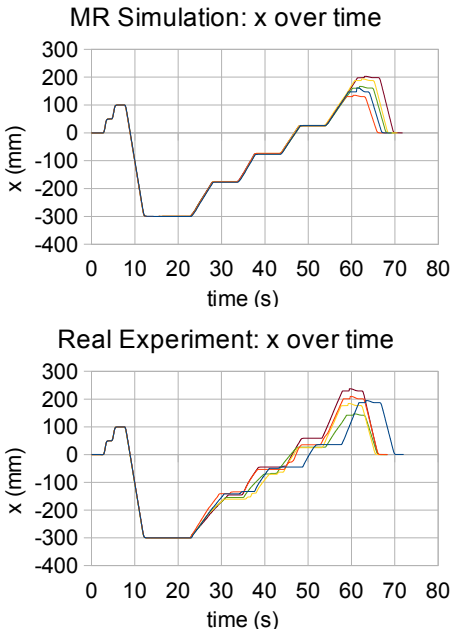


Figure 7.13: Raw trajectories of the robot's end-effector over time

Table 7.1: Average Times

	Screw Removal	Task Completion
MR Simulation	6.00 s/screw	69.97 s
Real Experiment	4.90 s/screw	68.95 s

to the MR simulation.

7.2.4 User Study

The findings from previous experiments provided evidence of the benefits of MR simulation and improvements to existing approaches, but do not reflect the user's acceptance of the technology. User studies are considered an essential part of the evaluation process, since ultimately, the fully developed MR robot simulator is intended to be used by robot operators and developers.

A within-subject user study was conducted to compare the user's experience in using MR simulation with a purely virtual simulator for performing a robotic task. Each participant carried out the task twice: once in the virtual simulation, and once in the MR simulation. To minimise any learning effects, some participants carried out the task in virtual simulation first, while some started with MR simulation.

In virtual simulation, the participants focused on a single virtual interface for carrying out the specified task. In comparison, MR simulation provides two interfaces: the AR interface and the AV interface. The participants were free to switch between the two interfaces provided for completing the task in MR simulation. The user's experience in using the two provided interfaces was also compared. Desktop videos were taken to examine the participants' actions and use of the provided interfaces when carrying out the task.

Task

The participants occupied the role of the human operator in the robot development process. They were required to operate the screw removal robot and monitor its operation to ensure the robot completed its task correctly. The participants were first told about the screw remover project and the procedure for removing screws using the PA10 robot. As outlined in Section 7.2.1, the procedure involved the user clicking on buttons on a GUI for 1) initialising the system, 2) starting the screw removal operation, 3) stopping the screw removal operation, and 4) returning the robot to the starting position.

The participants were then introduced to the MR simulator and the virtual simulator, Gazebo. They were given the opportunity to familiarise themselves with the keyboard and mouse controls of Gazebo that move a free-look camera for observing the simulation environment. Note that the same free-look camera was also available for them to use in the AV interface of MR simulation.

The operation of the screw removal task in virtual simulation was straight forward, with the participants using the GUI and the free-look camera for completing the task. In MR simulation, the participants were asked to perform an extra step that merges the real world and the virtual world. On the live video imagery captured by the camera overlooking the physical experiment environment, the participants needed to click with a mouse on the four corners of the frame that holds the ceiling beam. This calibrated the camera and registered the virtual objects into the real world for setting up the AR interface. The participants could then switch between the AR interface and the AV interface using the keyboard.

After completing the task using both simulation methods, the participants were asked to fill out a questionnaire, followed by a short interview investigating any problems that they encountered when using the two simulators.

Dependent Variables

User performance can be measured using objective metrics such as task completion time [170]. The task completion time for this screw removal task includes the time taken

to position the free-look camera before operating the robot, and any time taken by the participant to observe the simulation before confirming the completion of the operation. The task completion time excludes the time taken to set up the AR interface in MR simulation (users manually clicking on four corners for calibrating the camera), which was recorded separately.

The time spent on each visual interface in MR simulation was also recorded to investigate if there are any preferences in using the provided interfaces for this specific task.

Paired t-tests are used to identify any significant differences between the times in this repeated-measures user study. A p-value smaller than 0.05 suggests the difference is significant.

Questionnaire

A subjective questionnaire was used to collect qualitative data such as the user's thoughts and preferences in a similar manner to existing work on evaluating AR for HRI, e.g. [119]. The questionnaire comprised three sections. Section one collected the participant's demographic information. Section two measured the participant's experience in using virtual simulation and MR simulation. Section three measured the user's experience in using the AR interface and the AV interface for carrying out the specified task. A 7-point Likert scale was used throughout the questionnaire for measuring the user's experience.

The objective of questions in Section two was to gain initial insights to the user acceptance of a simulation method. This included finding out whether the user thought the simulation method was useful, efficient, and safe. In addition, the user acceptance could also be influenced by simulation realism and reliability. For example, if the simulation was perceived to be useful but its results were less reliable, this could suggest that the simulation method may not be suitable for development beyond the prototyping stages. Questions in Section two are listed below:

- Section two: Simulation Experience.
    1. This simulator is useful for debugging. (**Useful for Debugging**)
    2. This simulator is useful for evaluating prototypes. (**Useful for Prototyping**)
    3. The use of this simulator speeds up the development process. (**Speed up Development**)
    4. The use of this simulator minimises harm to the robot, the human, and the environment. (**Minimise Harm**)
    5. This simulator realistically simulates the context in which the actual robot operations will be performed. (**Realistic Simulation**)

6. The simulation results are a reliable representation of what occurs in the real world experiment. (**Reliable Results**)
7. This simulator is useful for teaching and training. (**Useful for Training**)

The objective of questions in Section three was to find out the suitability of AR and AV interfaces for robot development. The advantages of using AR visualisations for robot development as found in existing work [79] and the increased awareness of using AV in HRI [190] were evaluated to identify whether they were also applicable to help users operate and test this robot manipulator system. Questions in Section three are listed below:

- Section three: MR Interface Experience.

1. The presentation of information is intuitive to understand. (**Intuitive Display**)
2. This visual interface helps me learn about robot data. (**Aid Learning Robot Data**)
3. This visual interface helps in identifying inconsistencies between my model of the robot's view and the actual robot's view of the world. (**Identify Inconsistencies**)
4. This visual interface improves my awareness of the states and actions of objects (e.g. robot and the environment) in simulation. (**Improve State Awareness**)
5. This visual interface increases my awareness of potentially dangerous situations. (**Improve Danger Awareness**)
6. This visual interface provides flexible views of the experiment environment. (**Flexible View**)
7. This visual interface conveys spatial information between real and virtual objects. (**Provide Spatial Information**)

Two-tailed Wilcoxon Signed Rank tests are used to identify any significant differences between the users' experiences in using virtual simulation and MR simulation, and similarly, the users' experiences in using the AR interface and the AV interface.

Hypotheses

There were 6 hypotheses for this experiment.

1. Virtual simulation offers a safer environment than MR simulation for robot development because physical hardware used in MR simulation may cause damage to its surroundings.

2. MR simulation is a more accurate representation of the real world than virtual simulation because MR simulation includes real components that exhibit behaviours under the influence of real world variations, whereas virtual simulation uses simplified graphical and physics simulation models to emulate reality. There are also signs in existing work showing that hybrid simulations were used to obtain more realistic conditions for testing [220, 86]. On a similar note, findings from AR applications in laparoscopic simulations have also suggested that an AR simulator offers better realism over virtual simulations in surgical training [52].

3. MR simulation produces more reliable results than virtual simulation. Similar to the reason in hypothesis 2, more natural robot responses generated in MR simulation may lead the robot to complete its task in a closer manner to the real experiment, thus producing results that are more portable to the real world. This hypothesis is also formed based on findings from the quantitative evaluation presented in Section 7.2.3

4. The AR interface provides a more intuitive display of information than the AV interface because AV provides a view of the environment from a virtual perspective which may be less natural than the real world view in AR [52].

5. The AV interface provides a more flexible view of the environment than the AR interface because the movable camera view in AV allows users to freely navigate within the virtual environment and observe the experiment from different perspectives. This is in contrast to the fixed, limited view offered in the AR interface.

6. The AV interface increases the user's awareness of the robot status and potentially dangerous situations. The hypothesis is based on findings in existing work on AV visualisations for robotics [190], which illustrates that AV visualisations can help to describe the spatial relationship between the robot, the environment, and other task relevant information.

User Study Results

11 participants were recruited for this study (5 research engineers, and 6 technical staff). Out of the 11 participants, 10 participants had experience in computer programming and/or robot development (Computer programming: 9 participants with mean 14.10 and Standard Deviation (SD) 6.12 years of experience; Robot development: 8 participants with mean 6.83 and SD 6.85 years of experience) and 8 had used a simulation tool before for the development of computer/engineering systems.

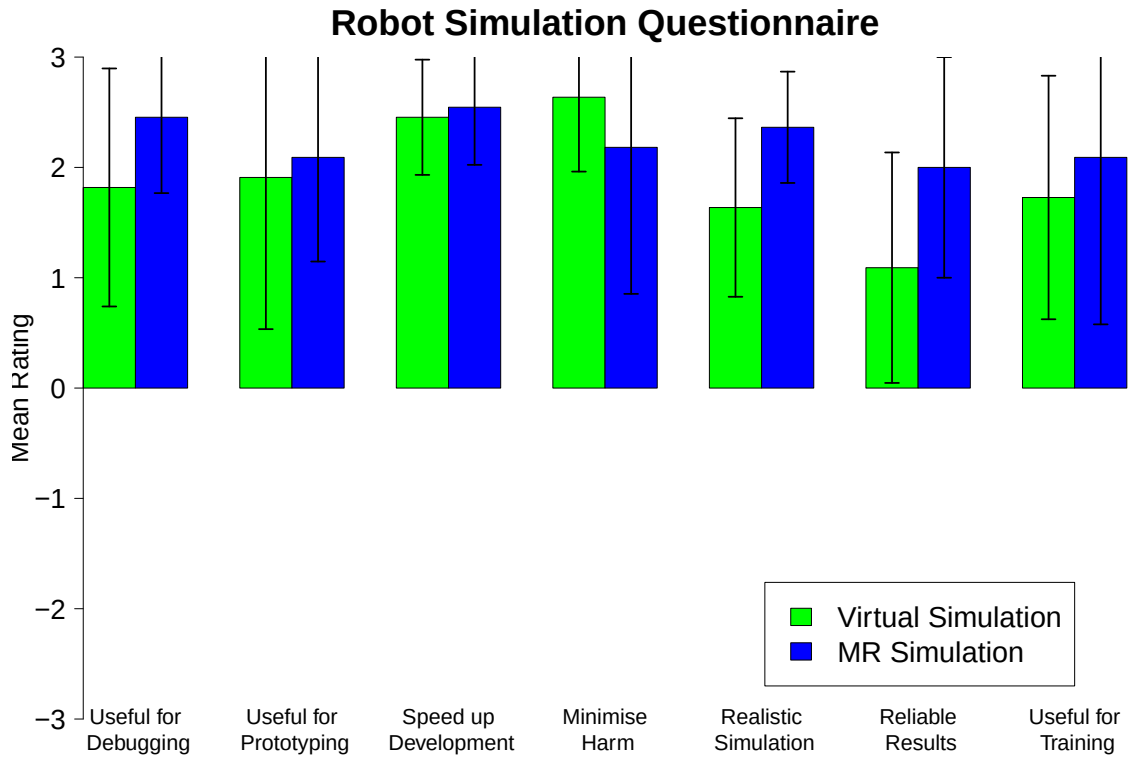


Figure 7.14: Differences between the users' experiences in using virtual simulation and MR simulation.

6 participants were randomly assigned to carry out the task in virtual simulation first, while 5 participants began with MR simulation.

Simulation Experience

All participants completed the task in virtual simulation (mean 90.73 seconds, SD 13.75) and in MR simulation (mean 102.00 seconds, SD 23.21) but the difference between the task completion times was not significant (t = -1.49, $p = 0.17$). Nevertheless, it was observed that the participants were being more cautious in MR simulation since a real robot was used, taking more time to observe the simulation environment through the interfaces provided and keeping an eye on the robot in the real environment throughout the task.

The questionnaire results on simulation experience are shown in Figure 7.14. The participants rated MR simulation significantly higher than virtual simulation for debugging (Z = -2.33, $p < 0.05$). The high rating was believed to be leveraged by the use of the AR interface in MR simulation, which helped users understand complex robot data.

The questionnaire results did not indicate that virtual simulation was significantly safer than MR simulation (Z = -0.96, $p = 0.33$), thus the finding did not support hypothesis 1. However, in the later interview it was found that 4 participants expressed concern about

the robot colliding with other real world objects in the environment, such as the ceiling beam frame, and 3 of whom reported being nervous while the screw removal operation was taking place in MR simulation.

MR simulation was rated significantly higher than virtual simulation for producing a more realistic simulation (Z = -2.06, $p < 0.05$), and more reliable results (Z = -2.46, $p < 0.05$), hence supporting hypotheses 2 and 3. The finding was particularly encouraging as it indicated that MR simulation gave users stronger confidence of accurate results, potentially allowing the simulator to be used for extended testing and evaluation of robot systems beyond the prototyping stage.

Statistical analysis did not suggest any significant differences between MR simulation and virtual simulation for prototyping (Z = -0.53, $p = 0.60$), speeding up development (Z = -0.58, $p = 0.56$), and teaching and training (Z = -0.97, $p = 0.33$).

MR Interface Experience

All participants used both the AR interface (mean 61.73 seconds, SD 26.18) and the AV interface (mean 40.27 seconds, SD 22.64) for carrying out the screw removal task, but the difference between the time spent on each interface was not significant (t = 1.65, $p = 0.13$). The mean time for setting up the AR interface before starting the task in MR simulation was found to be 14.73 seconds (SD 5.59); the extra time was an overhead for using MR simulation and needs to be minimised in the future.

The questionnaire results on MR interface experience are shown in Figure 7.15. The participants rated the AR interface significantly higher than the AV interface for intuitive display of information (Z = -2.07, $p < 0.05$), and also commented that overlaying virtual robot data in the physical world made understanding robot data and debugging intuitive, thus supporting hypothesis 4. However, there was no significant difference between the AR interface and the AV interface for learning robot data (Z = -1.93, p = 0.05), and identifying inconsistencies between the user's view and the robot's view of the world (Z = -1.63 $p = 0.10$) at the $p < 0.05$ level.

There was no significant finding indicating that the AV interface provided a more flexible view than AR interface (Z = -1.73, $p = 0.08$), thus it did not support hypothesis 5. A number of participants commented that they would prefer having the ability to move the physical camera around while using the AR interface rather than using a fixed camera. The results could differ if an onboard or movable camera was used. This needs to be further investigated.

Interestingly, a number of observations were not expected. The AR interface was rated equally important as the AV interface for improving awareness of robot states, and there was no significant difference between the AV interface and the AR interface for improving awareness of dangerous situations (Z = -1.19, $p = 0.24$). This finding did not support

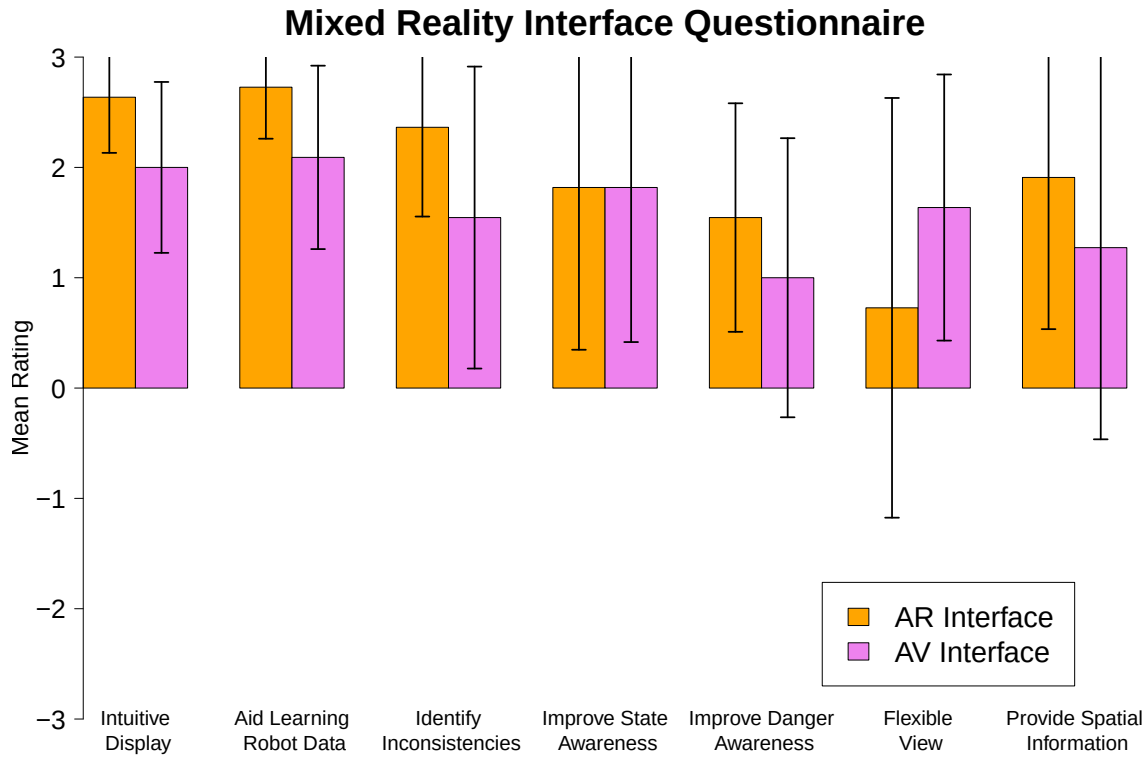


Figure 7.15: Differences between the users' experiences in using the AR interface and the AV interface.

hypothesis 6. Furthermore, the participants rated the AR interface significantly higher for providing spatial information than the AV interface (Z = -2.33, $p < 0.05$). This was a surprise as the AR interface is only a 2D display of information.

Overall, this user study obtained results that are different from existing work on using AV visualisation for improving user awareness [97, 190]. One possible reason is that the benefit of using this AV interface was not perceived to be high in a small workspace. The second possibility may be due to an unintuitive mapping of the mouse and keyboard controls to the free-look camera view. Although the users were given time to practice until they were comfortable, the later interview found that 5 participants (45%) commented on the difficulty of using keyboard and mouse inputs in the AV interface. Suggestions included using a 3D mouse or a joystick as an alternative input device. Reviewing the recorded desktop videos identified incidents that supported this finding. It was discovered that many participants did not continue to move the free-look camera after the screw removal operation started. The free-look camera was left at a fixed position either looking over the entire simulation environment or zoomed in at the three screws to be removed. Two participants became lost in the AV interface and immediately switched to the AR interface to continue monitoring the screw removal operation. This user interaction problem is not present in the original ecological interface for robot teleoperation [190] as only

a tethered camera view is used and no user inputs are necessary. While an adjustable viewing perspective is important as suggested in the original work, there are usability barriers that need to be overcome in order to exploit the full potential of the AV interface. Future work could also consider comparing the AV interface with an AR interface that allows users to control its viewpoint.

7.2.5 Summary

This case study evaluation analysed the use of MR simulation for the development of an industrial robot manipulator system. The task required of the robot was to remove screws from suspended ceiling beams. Motivated by concerns regarding safety and the considerable resource requirements in the testing process, an MR simulation involving a real robot manipulator and virtual ceiling beams and screws was created. The use of MR simulation provided a cost-effective solution to experimenting with the robot manipulator's movements and testing the screw removal process before it was ready for complete real world tests. This has helped to minimise the high demand of spare parts needed in the testing process, and prevented damage to the screw removal tool in case of failures.

Quantitative validation suggests that MR simulation may produce robot behaviours that are similar to those from the real experiment in terms of the trajectory taken by the robot to complete its task. Comparative evaluation shows an average RMSD of approximately 1.7% between the trajectories taken by the robot in the MR simulation and those in the real experiment. Moreover, user study results revealed that users were positive about MR simulation and believed the simulation results are a reliable representation of what occurs in the real world. The AR interface offered an intuitive view and helped users debug robot data. Nevertheless, it was found that it provided a limited view of the scene and a movable viewpoint should to be considered in the future. The AV interface lacked an intuitive user interaction tool for navigating in the simulation environment. More natural interaction methods need to be investigated in the future to improve the usability of the simulator.

7.3 Case Study Three: Cow Monitoring from a UAV

This case study evaluates the use of MR simulation for the development of a UAV system to be deployed in agriculture. One of the common agricultural tasks identified in the literature review is livestock and vegetation monitoring using UAVs. The high mobility of UAVs enables fast exploration of agriculture fields to collect information in remote areas that are otherwise time-consuming or difficult to access by farmers. The development of UAV systems is however difficult, especially during the testing stage due to reasons

concerning the high risk involved in the operation, site availability, weather conditions, and considerable resource requirements.

This project presents a good case study for evaluating a number of aspects of MR simulation. The nature of the robot platform in this project differs from those in the previous case studies described in this chapter, and demands stricter safety requirements during the development process. This case study evaluates the ability of the MR robot simulator to:

- provide robot developers safe and flexible ways of creating test scenarios for high risk robot operations.
- handle erratic behaviour of an aerial robot platform when mixing the two worlds to create MR interactions.
- produce robust and accurate AR visualisation under the rapid motion of the UAV.

7.3.1 Cow Monitoring Project

In discussion with staff at a local research dairy farm, it was identified that one of the main hurdles of efficient farming is the lack of support for monitoring and tracking cow statuses on the farm. The farmers are interested in information such as a cow's health status and temperature (for indicating signs of pregnancy or oestrus), which affect its productivity. However, a single farmer is typically responsible for looking after a herd size of several hundreds of cows which makes the process of information management difficult. Currently, automatic milking machines are used to record cow data during the milking process but they do not provide an on-demand solution. Farmers need to wait for the herd of cows to be milked before gaining access to the up-to-date data. The milking process may take long, including the time taken for the cow to get from the field to the machine. Putting more of such expensive machinery in the farm for the purpose of faster data collection is also not a cost-effective solution.

In response, a vision based cow monitoring aerial robot system is proposed for this task. Autonomous aerial robots can potentially improve efficiency by monitoring and collecting data of agricultural objects from the air. In particular, a VTOL aircraft is chosen for its hovering capability which is ideal for the monitoring task. As a proof of concept, the development began with a micro aerial robot system. The robot used is a quadrotor helicopter, named Hummingbird, from Ascending Technologies [3], see Figure 7.16. The platform measures 0.53m in diameter and has a maximum flight time of approximately 20 minutes and a payload of 200 grams. Hummingbird comes with an onboard controller that implements attitude stabilisation based on the onboard IMU. The autopilot system allows for GPS-based navigation but is disabled in the current stage of the development.

Figure 7.16: Hummingbird quadrotor helicopter from Ascending Technologies [3]. A lightweight camera is mounted beneath the platform.

Communication between the ground control station and the vehicle is performed over an XBee radio link. A Point Grey Firefly MV FFMV-03M2C camera with 4mm lens is mounted directly beneath the centre of the quadrotor and oriented to look downwards for the cow monitoring task. 640x480 colour images are transferred over the firewire cable to the ground station where all the computations are performed.

As the development of the project is at the prototyping stage, testing is primarily carried out in an indoor environment. Figure 7.17 shows the robot in the experimentation space. Four A1-sized textured aerial images are laid out on the laboratory floor to form a mock-up agricultural environment for testing the vision based position control component of the UAV system. Note that the experimental setup is not to scale and designed for prototyping purposes only.

Task Description

The overall goal of the project is to enable the UAV to autonomously detect a target cow on the ground using the onboard camera and hover over the target so that any data of interest can be collected by specialised instruments carried onboard the UAV. The quadrotor should try to stay hovering above the target and follow its movement for as long as it is necessary for the data collection process. To accomplish this goal, the project is divided into the following tasks:

- Object Detection: Detection (and possibly recognition) of a cow using the onboard vision sensor.
- Object Following: High level waypoint control of the robot to follow the target object.

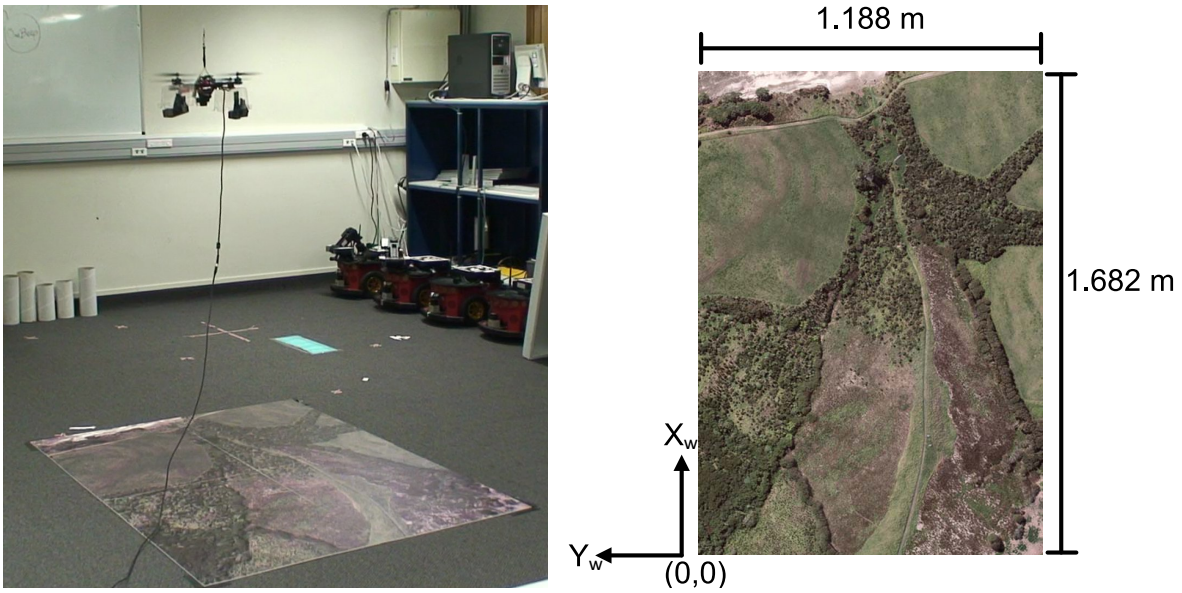


Figure 7.17: a) Indoor UAV system test setup. The UAV relies on vision for position control, and it is shown hovering over the textured aerial map on the floor. b) Dimensions of the agricultural aerial map used in the experiments. The bottom left corner defines the world origin.

- Position Control: Low level vision based position control of the robot to keep it hovering at the specified location.
- Data Collection: Collect data about the target object using specialised instruments onboard.

System Prototype

The project takes a common approach in robot development and splits up the four tasks into four components. The four components are developed in parallel which will at the end be integrated into a complete UAV system.

The design of the system must be modular. Future work will look at building on the base UAV system to extend its capability to perform other agricultural tasks such as weed detection which is another important issue in agriculture that demands robot automation.

The project is in the early stage of development. Three (out of four) prototype components have been implemented as a proof of concept. Brief descriptions of the components are given below.

- Object Detector: Processes image data and identifies target location in image coordinates (x_i, y_i). In this prototype, object detection is based on colour detection in the Hue Saturation and Value (HSV) colour space. After thresholding the input image using the specified HSV boundaries and applying filtering operations, con-

tours of potential matches can be extracted. Currently the system only tracks one target, which corresponds to the largest contour identified.

- Object Follower: Generates position commands in world coordinates (x_w, y_w, z_w, ψ_w) based on the image location of the target to keep it in the view of the robot. A linear feedback control algorithm is first used to derive position commands in the robot's coordinate system. Given that the robot's pose relative to the world can be obtained from the Position Control Component (vision based pose estimates), the local position commands can be transformed into the world coordinate system.
- Position Control: Implements Proportional–Integral–Derivative (PID) controllers for robot's x, y, z, and yaw control based on the position commands. Controller feedback is in the form of vision pose estimates. Notably, the extensible AR tracking algorithm described in Section 6.3 has been used to provide the pose estimates.

7.3.2 Mixed Reality Simulation

It is clear that testing the UAV system with a real cow in the early stages is impractical. It is too dangerous to conduct field tests as it could harm the real animal and the surrounding agricultural objects in case of failures. Moreover, we can not control the behaviour of a real cow. Forcing a real cow to move from one point to another to test the performance of the object following algorithm is not a feasible testing method. MR simulation is able to help in testing the system by simulating a virtual animated model of the target cow to be followed. It provides developers a complete control over the virtual target and enables them to efficiently create repeatable tests of various scenarios, e.g. moving pattern and speed of the cow.

To provide robot developers as much insight to real world tests as possible, a real robot is used in this MR simulation. The MR simulation is able to augment the robot's visual sensing with the virtual cow, which enables the robot developers to observe the behaviour of the actual UAV in flight as it responds to the simulated input in real time. This is believed to have a benefit over testing using pre-recorded videos.

In summary, the MR simulation created for this project consists of:

Real: UAV platform, onboard vision sensor, aerial map (indoor mock-up agricultural environment).

Virtual: cow.

Mixing the Two Worlds

The main challenge in creating the MR simulation for this project is in keeping the two worlds in a consistent state, which is necessary for generating correct and accurate MR

interactions. To facilitate interaction between the real robot and the virtual cow, the primary task is to augment the robot's vision input to reflect the presence of this virtual cow. This task is, however, difficult due to the erratic motion of the mobile UAV platform. Registration of the virtual cow must be accurate since the behaviour of the robot is directly dependent on the resulting augmented image data produced by the MR simulation. Large errors in augmentations could lead to unexpected UAV movements and potentially cause a serious crash. Robust AR tracking is essential for both accurate simulations and the safety of the experiment.

The extensible AR configuration described in Section 6.3 has been chosen for augmenting the robot's vision sensor due to PTAM's robust performance under rapid camera motions. The high mobility of the UAV platform also suggests the possibility that it is likely to continuously explore different scenes of the operating environment, and thus extensible tracking technology is necessary. In this MR simulation, the onboard camera is modelled using a camera entity, and it is attached with a behaviour that uses the pose estimates from the extensible AR system for registering the virtual target cow onto the live video imagery. This generates augmented image data that will be delivered to the Object Detector component of the UAV system.

It is noted that photorealistic rendering of virtual objects would create more realistic augmented image data. However, it adds further computational overhead to the use of the MR simulation, thus it is considered less important and omitted in this early prototyping stage of the UAV system. Efficient methods for creating photorealistic renderings may need to be investigated in the future.

The next task is to ensure the state of the real robot and its virtual representation are consistent. The real UAV platform, modelled using a position entity, requires a robust localisation algorithm (for updating the pose of its virtual representation) that is able to cope with the rapid and erratic motions of this mobile platform. To solve this problem, the extensible AR tracking technology has also been used for localising the robot. By knowing the offset of the onboard camera from the center of the UAV platform, the AR tracking pose estimates can be used to calculate the robot's position and update its virtual representation accordingly.

A system diagram of the components in this MR simulation is shown in Figure 7.18. The two MR entities created are MRCamera and MRPosition. Instead of reading images from the real vision sensor, the Object Detector processes augmented images generated by MRCamera for cow detection. Similarly, position commands are sent to MRPosition who forwards them to the Hummingbird autopilot system.

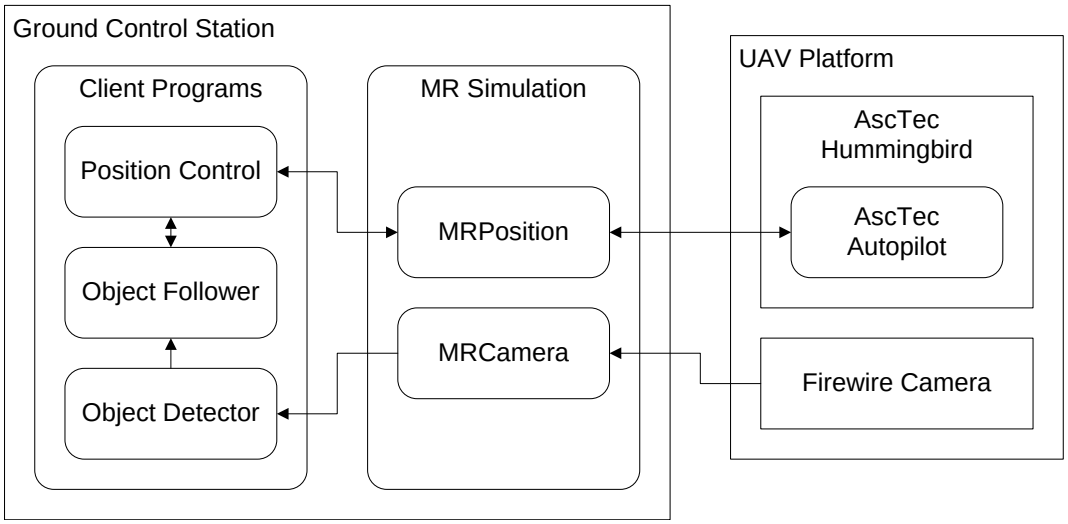


Figure 7.18: System diagram of the MR simulation for testing the implemented UAV system. The Gazebo interface and other components in the MR robot simulator are omitted in this diagram for clarity.

MR Interfaces

Figure 7.19 shows screenshots of the UAV based cow monitoring operation taken from the two MR interfaces. The AR interface provides users a view of the environment from the robot's onboard camera. Creating AR visualisation from an onboard camera has been found to be difficult in the first case study evaluation which relies on the planar region tracker of AR configuration one. Among reasons stated earlier, this case study switches to the extensible AR configuration for creating AR visualisation that minimises AR discontinuity.

Due to the high mobility of this robot, as well as findings from the screw remover case study evaluation, an AV interface consisting of a single free-look camera mode is considered insufficient for the task, since it requires users to constantly and manually control the viewpoint of the camera to follow the movement of the robot. Therefore, multiple camera modes are offered in the AV interface of this case study evaluation. In addition to the free-look camera, a tethered camera, and a fixed camera with target tracking capability are provided.

7.3.3 Evaluation

To assess the MR robot simulator's performance in simulating the prototype UAV system, an experiment was conducted that involved the robot following a moving virtual cow. The purpose is to investigate whether the augmentation in the robot's vision sensor data would be sufficiently robust and accurate to facilitate reliable robot responses. In the experiment, the virtual cow was animated to move in three different maneuver patterns

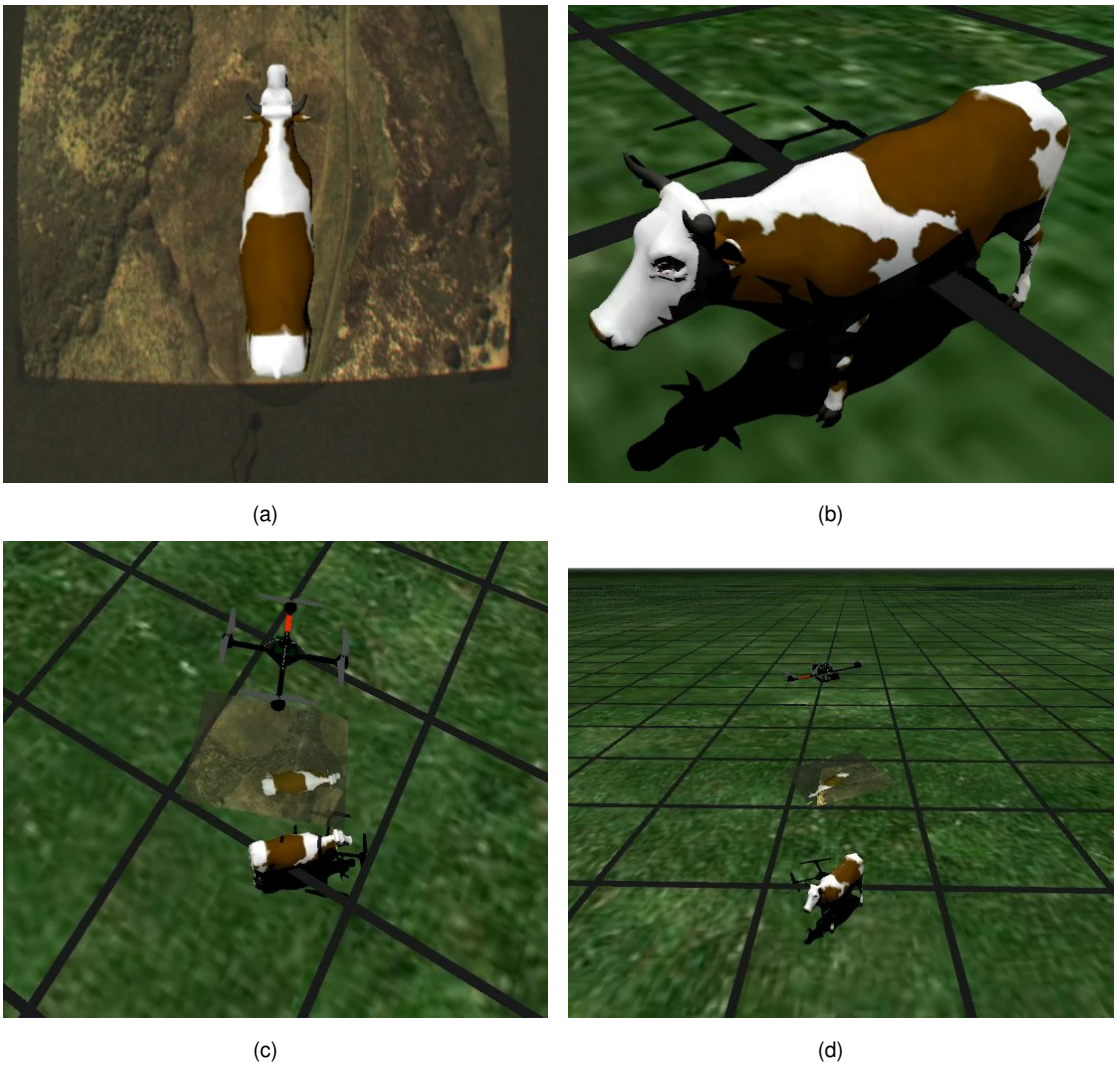


Figure 7.19: Screenshots of the MR interfaces created for the development of the prototype UAV system. a) The AR interface showing a bird's eye view of the scene from the robot's perspective; the virtual cow is seen augmented over the experimentation environment. b) The AV interface with free-look camera mode: in this example, the virtual camera is moved to focus on the target cow to be monitored in the experiment. c) The AV interface with tethered camera mode: the virtual camera follows and tracks the movement of the UAV. d) The AV interface with fixed camera mode: similar to the tethered camera mode but with a fixed virtual camera.

within the space covered by the aerial map on the floor. The pose of the robot given by the extensible AR tracker as it followed the cow was recorded. The MR robot simulator (using the extensible AR tracker based on PTAM) was deployed in Linux on an Intel(R) Xeon(R) W3530 2.8GHz CPU with 3GB of RAM and a NVIDIA Quadro FX 580 graphics card. The MR robot simulator frame rate was observed to average between 24Hz to 30Hz while running on the same computer as the UAV prototype system.

The results are shown in Figure 7.20. The virtual cow moved in 1) a straight line, 2) a cosine pattern, and 3) a sine pattern. The graphs show that the robot followed the cow

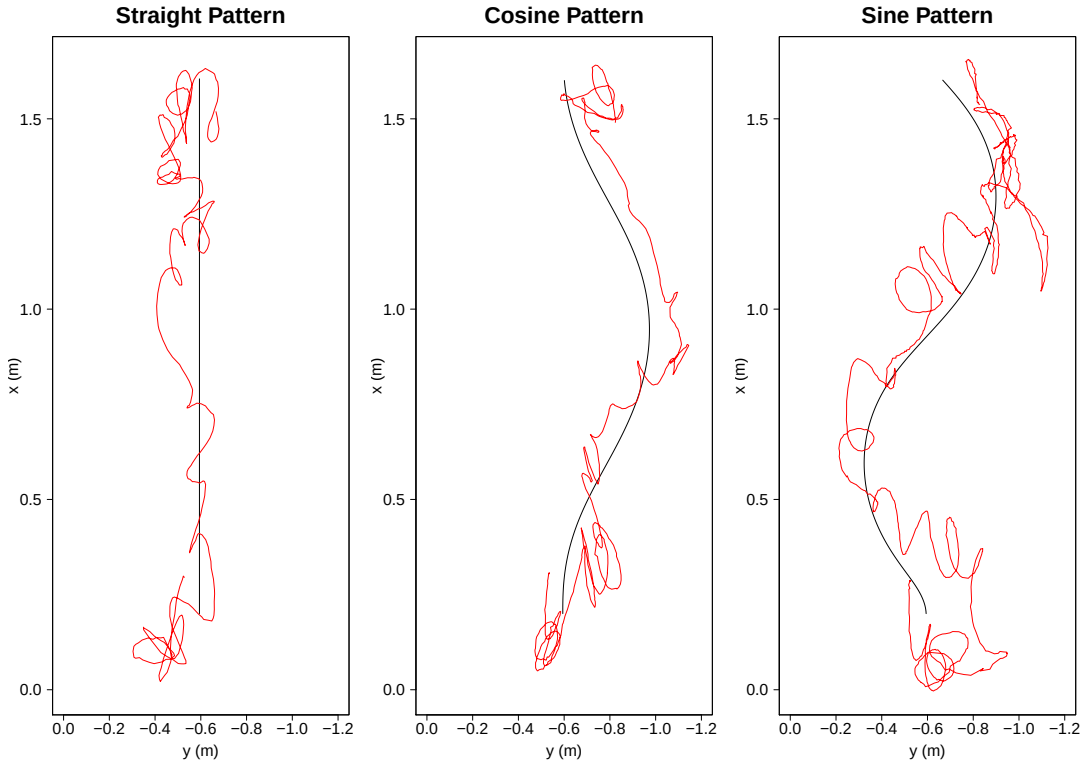


Figure 7.20: UAV flight paths (red) from following three different moving patterns of the virtual cow (black).

in each case, demonstrating successful interactions between the real robot and the virtual cow. The results also reveal to the robot developers the flight instability of this prototype UAV system, which were useful for future developments. A video, `UAVMonitoring.mp4`, illustrating the experiment where the UAV followed the virtual cow moving in the sine pattern is enclosed on the accompanying CD-ROM.

The extensible AR tracker correctly augmented the virtual cow in the view of the camera. Figure 7.21 shows screenshots of the Object Detector window, which displays the processing being carried out on the augmented vision data generated by the MR robot simulator. The virtual cow remained accurately aligned with the real world scene as it was animated to move across the aerial map.

To measure the accuracy of AR tracking, the PTAM map data and keyframes constructed were saved for later evaluation. Figure 7.22 illustrates the PTAM map that was constructed. A later experiment reinitialised tracking based on the saved data and manually measured the pose estimate errors. Starting from $x = 0$, $y = 0$, and $z = 0.6$, the camera was manually translated at increments of 0.1m along the x, y, and z axes separately. Ten measurements were taken each, and the average errors were approximately 0.02m in x, 0.03m in y, and 0.06m in z. It was found that the errors were predominantly caused by the initial offset of the PTAM map origin from the actual world origin, revealing drifts of less than 0.01m in the area covered in this experiment. However, it is expected

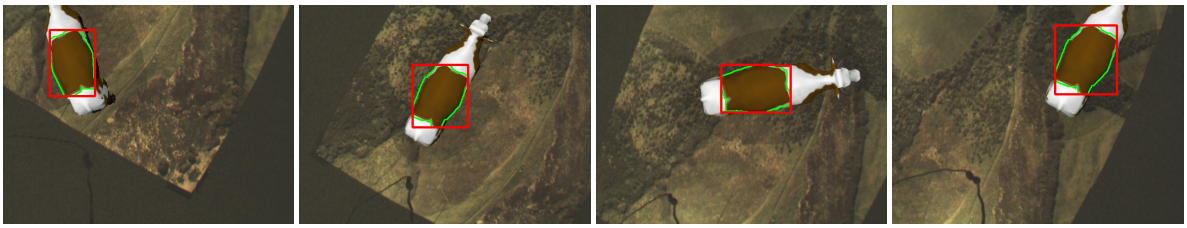


Figure 7.21: A sequence of screenshots illustrating the image processing carried out by the Object Detector. Simple colour detection was implemented as a proof of concept in this prototype component for cow detection. A bounding box is drawn in red to highlight the detected region. This sequence corresponds to the sine maneuver pattern of the cow.

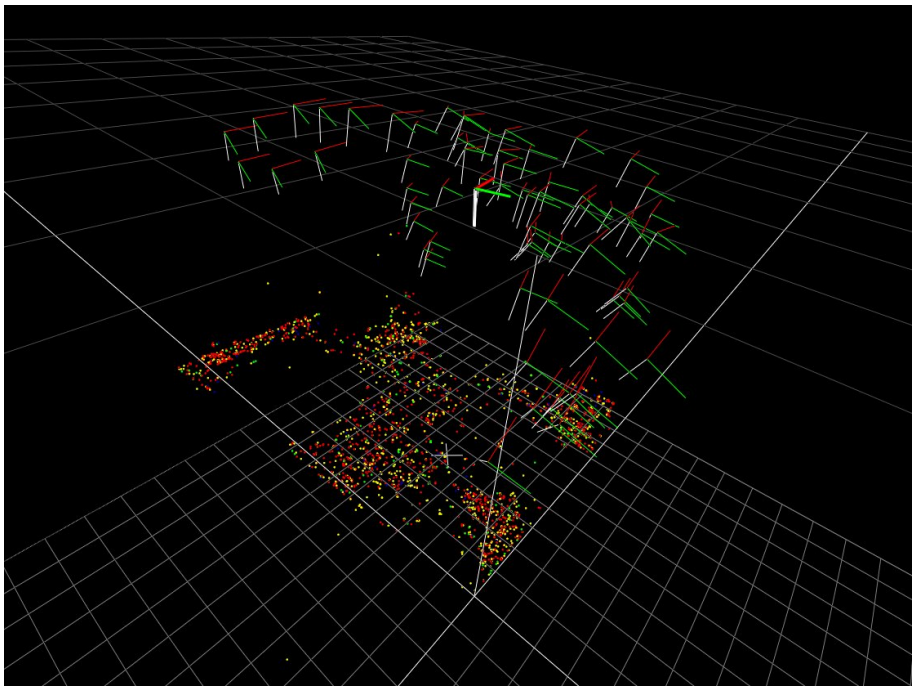


Figure 7.22: The PTAM map constructed by the extensible AR tracker in the UAV based cow monitoring operation.

that drift errors would be more apparent in larger experimentation areas.

The pose estimates of the robot provided by the extensible AR tracker, the video demonstration, and the AR imagery seen by the Object Detector window show that the extensible AR tracker was able to cope with several sudden and erratic motions of the UAV. While no significant errors were observed in the pose estimates produced during the flight, it was noticed that the extensible AR tracker picked up fewer points in scenes with large portions of grasslands due to its highly repetitive pattern. This is an expected behaviour of the underlying PTAM algorithm used. Nevertheless, in this experiment, there were sufficient textured patches surrounding this grassland region to maintain the quality of the tracking, but it must be noted that future outdoor experiments involving scenes with very few distinct features may cause the algorithm to produce poor results.

Overall, the use of the MR simulation was able to provide insights to the behaviour of the prototype UAV system for the cow monitoring task. It helped to identify areas for improvements, mainly pointing towards the need for a more robust control logic for the object following task. It is also worth pointing out that the MR simulation enabled rapid creation of the three cow maneuver scenarios. Scripts were written in a matter of minutes to create the animated movements, which were then immediately deployed in the MR simulation. This demonstrates the ability of the MR simulation to increase the efficiency of robot development, and at the same time offers the advantage of generating repeatable tests involving real components.

7.3.4 User Study

The previous user study results presented in Section 7.2.4 provided initial insights to the user's perception of the MR simulation technology and its visual interfaces. The study focused on a robot operation task, which did not involve any implementations. The implementation stage constitutes a large part of the robot development process, and it is in the interest of this research to conduct a user study to investigate how the MR robot simulator could help robot developers implement robot software components.

In this case study evaluation, a more focused user study was conducted. It targeted a user group of experienced robot developers and computer programmers who were required to write software code for controlling a robot. In comparison to the previous user study, it is an observational study in which participants were not assigned to any particular conditions. The participants were provided with a number of different simulation configurations, including MR simulations, and it is up to the participants how they use the simulations. The reason is that it is acknowledged the task given may take the participants through different stages of development, and the experimental setup may need to vary accordingly to meet their needs. Instead of restricting the participants to a single condition for the whole development process, it is more practical to let them experiment freely using the provided setups. This user study is similar to the ethnographic study presented by Collett and MacDonald [79] in their work on evaluating AR visualisations for robot development, but in comparison, formal evaluation metrics were used in this user study which allowed statistical tests to be performed to identify significance in the results.

The participants were video-taped for post-study analysis. Desktop videos were also taken to examine the participants' actions and their use of the provided simulation tools when carrying out the task.

Task

The participants occupied the role of a robot developer for the development of a small part of the UAV system described in Section 7.3.1. The primary task was to implement a small piece of C/C++ code within the Object Follower component and test their implemented system using different simulation setups. The conductor of the user study was the participant's assistant, whose primary role was the safety pilot of the experiments.

The participants were given the fully implemented Object Detector and Position Control components, and a skeleton code of the Object Follower component which they had to complete. The skeleton code is shown in Listing 7.1. The `follow()` function in the Object Follower class is missing the logic for controlling the robot's movement. The participants were told that the task was to implement the remainder of this function, by converting the target's image coordinates into position commands in world coordinates so that the robot can hover above the target and keep the target in its view.

Three development methods were provided to the participants, two of which are MR simulations. The hypotheses presented later describe how these three methods are believed to benefit the development process. The development methods are:

- **Method One**:

  Virtual simulation: This setup takes place in a completely virtual environment. It uses Gazebo as the virtual robot simulator. Virtual images are transferred to the UAV system. The simulator receives position commands from the Object Follower component and visualise the behaviour of the UAV in the virtual environment. No complex aerodynamics is integrated in the simulation. A simple dynamics model is used to mimic the behaviour of the UAV under the control of the PID controller.

- **Method Two**:

  Free-moving Camera: This setup lets users *manually* move the real UAV platform with the attached onboard vision sensor over the mock-up aerial map. However, the UAV does not respond to position commands sent by the Object Follower Component. A virtual cow is placed in the environment and augmented images are transferred to the UAV system. The users could ask the assistant to help with this task, by instructing the assistant to fly the UAV in directions needed using the remote control transmitter or to move the platform physically by hand. The pose of the UAV is reflected in the AV interface in real time.

- **Method Three**:

UAV + Camera: This setup involves the real UAV taking flight over the aerial map under the control of the Position Control component. A virtual cow is placed in the environment and augmented images are transferred to the UAV system. The UAV responds to position commands generated by the Object Follower Component. The pose of the UAV is reflected in the AV interface in real time.

The participants were free to choose and switch between the development methods for testing their implementation as the development progressed. They were asked to keep in mind the stage of development which they considered themselves at when they were using the simulations. It is assumed that the development cycle of this particular task could be divided into four stages:

1. **Stage One:** Initial implementation and validation of logic
2. **Stage Two:** Debugging and fixing errors in code
3. **Stage Three:** Evaluating performance of algorithm
4. **Stage Four:** Tuning and refining parameters

Scripts for compiling the component code, starting the UAV system, and operating the simulators were given. This was to allow the participants to focus purely on the designated task rather than spending time on setting up the development environment. Sample test programs for moving the virtual cow were also provided to the participants to help them with the testing process, and these programs can be used in all three development methods. The test programs include the straight, cosine, and sine cow maneuver patterns that were used in the previous experiment, as well as a program that lets users move the virtual cow through key presses.

Listing 7.1: The C++ skeleton source code of the Object Follower component provided to the participants.

```
1  #include "follower.h"
2  #include <iostream>
3  #include <libplayerc++/playerc++.h>
4  #include <math.h>
5
6  #define IMAGE_WIDTH   640
7  #define IMAGE_HEIGHT   480
8
9  #define MAX_CMD_X   2.2
10 #define MIN_CMD_X   -0.1
11 #define MAX_CMD_Y   0.1
```

```
12 #define MIN_CMD_Y  -1.5
13 #define MAX_CMD_Z  1.5
14 #define MIN_CMD_Z  0
15
16 using namespace PlayerCc;
17
18 Follower::Follower (PlayerCc::PlayerClient* client, PlayerCc::
       Position3dProxy* proxy) {
19   playerClient = client;
20   position3dProxy = proxy;
21
22   // default frequency
23   // the follow() function will be executed at 1/0.5 = 2Hz.
24   // change to suit the behaviour of the follow() function
25   frequency = 0.5;
26
27   followerInit = false;
28 }
29
30 Follower::~Follower(){}
31
32 int Follower::init(){
33   followerInit = true;
34   return 1;
35 }
36
37 bool Follower::isInit(){
38   return followerInit;
39 }
40
41 /*
42  * Given the target's image coordinates (pixels), send appropriate
43  * position commands in world coordinates (metres) to follow
44  * the target and keep it in the view of the camera
45  */
46 int Follower::follow(double imageCoordX, double imageCoordY, bool detected)
47 {
48   if (!detected)
49     return -1;
50
51   // get information about the current robot pose in
52   // world (absolute) coordinates
53   double robotWorldX = position3dProxy->GetXPos();
54   double robotWorldY = position3dProxy->GetYPos();
55   double robotWorldZ = position3dProxy->GetZPos();
56   double robotWorldYaw = position3dProxy->GetYaw();
```

```
57
58    /// TODO:
59    /// implement logic for generating world (absolute) position commands,
60    /// helper coordinate system transformation code is provided below for
61    /// local (relative) -> world (absolute) transformation
62
63    // determine position commands in robot's local (relative) coordinates
64    // centre of the robot is at (0,0), x is forward, and y is to the left
65    double posCmdLocalX = 0;
66    double posCmdLocalY = 0;
67
68    // simple coordinate system transformation from local (relative) to
69    // world (absolute) coordinates
70    double posCmdWorldX = robotWorldX + cos(robotWorldYaw)*posCmdLocalX - sin
          (robotWorldYaw)*posCmdLocalY;
71    double posCmdWorldY = robotWorldY + sin(robotWorldYaw)*posCmdLocalX + cos
          (robotWorldYaw)*posCmdLocalY;
72    double posCmdWorldZ = 1.1; // fly at fixed height?
73    double posCmdWorldYaw = 0; // keep yaw (in radians) constant?
74
75    // sendCommand takes world (absolute) positions, not local (relative)
76    sendCommand ( posCmdWorldX, posCmdWorldY, posCmdWorldZ, posCmdWorldYaw);
77
78    return 1;
79  }
80
81  /*
82   * check to see if the position commands are within the allowed
83   * experimentation space then send the commands to the robot
84   */
85  void Follower::sendCommand(double cmdX, double cmdY, double cmdZ, double
        cmdYaw){
86    if (cmdX > MAX_CMD_X)
87      cmdX = MAX_CMD_X;
88    if (cmdX < MIN_CMD_X)
89      cmdX = MIN_CMD_X;
90    if (cmdY > MAX_CMD_Y)
91      cmdY = MAX_CMD_Y;
92    if (cmdY < MIN_CMD_Y)
93      cmdY = MIN_CMD_Y;
94    if (cmdZ > MAX_CMD_Z)
95      cmdZ = MAX_CMD_Z;
96    if (cmdZ < MIN_CMD_Z)
97      cmdZ = MIN_CMD_Z;
98
99    position3dProxy->GoTo(cmdX, cmdY, cmdZ, 0, 0, cmdYaw);
```

```
100 }
101
102 void Follower::setExecutionFrequency(double freq){
103   frequency = freq;
104 }
105
106 double Follower::getExecutionFrequency(){
107   return frequency;
108 }
```

Procedure

The cow monitoring project was first explained to the participants, and the design of the UAV system was shown. Before starting the task, a quick demonstration was also given that walked them through the procedures for running the three simulation configurations. They were given the chance to see the virtual environment in Method One, the augmented view of the scene in Method Two, and the flight behaviour of the UAV system under the control of the Position Control component in Method Three. After making these observations, they were asked to commence the implementation and testing of the Object Follower component.

When a participant indicated task completion, the working of the system was verified in a test run that tested whether the UAV could correctly follow the virtual cow moving in the sine maneuver pattern. If the UAV failed to follow the cow, the participant was given a chance to refine their code and conduct further testing. A second and final test run was held upon indication by the participant. The participants were given a maximum of two and a half hours for the task.

After completing the task, the participants were asked to fill a questionnaire, followed by a short interview investigating any problems that they encountered during the development.

Questionnaire

The questionnaire comprised four sections. Section one collected the participant's demographic information. Section two collected information on the participant's time distribution in using the provided development methods over the four stages of the development cycle. Section three measured the participant's experience in using the different development methods. A 7-point Likert scale was used throughout this section for measuring the user's experience. The last section collected information on the participant's preferences in using the provided development methods over the different stages of development.

The questions in Section two are listed below. While there were some overlaps in

terms of general questions on safety with the previous users study questionnaire, they were still considered to be important given the high risks involved in this development task. In addition, questions specific to this user study were asked in the questionnaire to identify whether a method is more suitable than another for this robot development task.

- Section two: Development Method Experience:

    1. This method helped to keep negative consequences of object following failures to minimal (if any). (**Minimise Neg Consequences**)
    2. This method was safe for the developer. (**Safe for Developer**)
    3. This method has a short learning curve. (**Short Learning Curve**)
    4. The use of this method resulted in fewer mistakes in the development process. (**Fewer Mistakes**)
    5. This method was effective in terms of producing good quality results and advancing the development progress. (**Effective Development**)
    6. This method gave me confidence that the robot could achieve the goal in this task. (**Increase Confidence**)
    7. This method helped in understanding the flight behaviour of this particular aerial robot system. (**Aid Understanding**)
    8. This method was useful for tuning control parameters of the object following algorithm. (**Useful for Tuning**)
    9. This method was useful for experimenting with different flight maneuvers for object following (e.g. different altitudes, trailing distances). (**Useful for Experimentation**)

Hypotheses

The hypotheses were:

1. The participants would vary the level of virtualisation of the simulation as the development progresses, adding more real world components into the experimental setup as necessary. The incremental development approach was found to be common in the development of UAV systems as described in Section 3.1.2. In addition, Method One and Method Three are believed to be essential because the simulated and real UAVs in these methods respond to position commands generated by the control algorithm, while Method Two benefits users who take a more cautious and systematic approach to testing.

2. Method One is useful for initial implementation (Stage One) and debugging errors (Stage Two). It may also be useful for experimenting with different object following strategies (Stage Three) because experiments can be conducted safely and rapidly in virtual simulation to help users explore their ideas without the cost of running real world tests.

3. Method One is safer and easier to use compared to other methods because it does not include the overhead of setting up and preparing the real UAV for test flights. However, it provides the users lower confidence of the actual behaviour of the UAV system in comparison to Method Three.

4. Method Two is useful for helping users understand the robot's field of view, and for validating the position commands by comparing them against the real world experimentation area before commencing test flights (Stage One and Stage Two). This is important for ensuring safety since flying the UAV outside the area would cause the vision based position controller to fail and the UAV to crash.

5. Method Two may also be used as a safer alternative for tuning and refining the control parameters of the algorithm (Stage Four) because there may be uncertainties and risks when directly deploying these changes to the real UAV platform in Method Three. Method Two may help users validate the new position commands with respect to the physical environment seen through the UAV's onboard camera.

6. Method Three is heavily required in the later stage of development for evaluating performance (Stage Three) and tuning and refining parameters (Stage Four). This is because the method is able to generate UAV behaviours in response to the user's control algorithm, and the users may also believe that the results produced are more reliable.

7. Method Three is effective and results in fewer mistakes, because a responsive UAV system enables users to observe the actual UAV behaviour under the control of their algorithm. This may help to provide a clearer understanding and expectation of the robot behaviour, as well as the changes that need to be made to accomplish the given task. However, the risk of test flights is higher and this method may also require a longer learning curve.

User Study Results

The user study was advertised to researchers and students in the academic field as well as computer scientists and engineers in the profession. 10 participants were recruited for this study (4 academic researchers, 5 students at the postgraduate level, 1 software engineer),

	Complete Task?	Time (min)	Methods Used	Monitoring Approach	UAV Crash?	Controller Type
P1	Yes	132	All	Both	No	Proportional
P2	Yes	150	M1,M3	Interfaces	Yes	Proportional
P3	No	N/A	M1,M3	Interfaces	No	Proportional
P4	Yes	74	All	Interfaces	No	Rule-based
P5	Yes	102	M1,M3	Both	Yes	Fuzzy
P6	Yes	83	M1,M3	Both	No	Proportional
P7	Yes	149	M1,M3	Both	No	Proportional
P8	Yes	116	All	Both	No	Proportional
P9	Yes	111	M1,M3	Interfaces	No	Proportional
P10	Yes	150	All	Both	No	Proportional

Table 7.2: User's performance data and other observations made. Monitoring Approach describes how each user monitors the operation of the robot during the development; possible options: MR Interfaces, Real world, and Both. UAV Crash corresponds to any system failure that led to a UAV crash when using Method Three. Controller Type describes the algorithm implemented by the participant in the Object Follower component.

all of whom were experienced computer programmers (mean 8.95 years of experience, SD 6.82) with Computer Science or Engineering background. 7 participants had experience in robot development (mean 3.93 years of experience, SD 2.05). However, none of the participants had experience in developing aerial robot systems.

There was no significant difference in programming experiences between the academic researchers and postgraduate students (t = -0.52, $p = 0.62$). It would be interesting for future work to recruit a larger pool of robot programmers and identify if the level of programming experience would have an effect on their development approach.

Table 7.2 summarises each participant's performance as well as observations made by the conductor. P1, P2, P3, etc. are participant codes, and M1, M2, M3 are the three development methods.

Participant's Development Approach

9 participants successfully completed the task within the given time. The remaining participant, P3, reported to have struggled to understand the relationship between the different coordinate systems, which the development methods and the MR interfaces were unable to help the user in this respect.

More than half of the participants (6 participants) did not choose to use Method Two. When later questioned about the reason, the responses were mainly because the participants believed a) the method was not useful for this particular task since the focus is in developing control algorithms but the UAV does not respond to commands (4 comments), and b) the method was redundant and the same benefit of this method could

be obtained using Method One and/or Method Three (2 comments).

Interestingly, different opinions were collected from 2 participants who decided to use Method Two. P1 in particular made use of Method Two in each of the four stages of development. P1 commented that it was necessary to see the real experimentation environment to understand the robot's field of view when determining the starting control parameters. The non-responsiveness of the robot in this method was considered beneficial to P1 who was then able to focus on the debugging output while the robot stayed in a particular position relative to the target object. P8 used Method Two for the same reason. The other 2 participants used Method Two primarily for verifying position commands through debugging printouts as an extra safety step before moving on to using Method Three. The findings supported hypothesis 4 to only some extent since only a minor proportion of the participants used this method. A useful comment collected was that the benefit of using Method Two could be leveraged if the simulation tool provided readily available graphical aids to help in visualising the position commands in the MR interfaces.

Observations on the order of development methods used found that 9 participants began with Method One. P8 was the only exception, who decided to use Method Two to collect all the parameters and implement them in the algorithm before testing it using Method One. All participants used Method Three as the last development method. The results suggest that the participants slowly transitioned from virtual simulation to MR simulation (with the real UAV) as the development progressed. This finding supported hypothesis 1.

The average percentage of time that the participants spent in each development method at each stage is shown in Table 7.3. The large portion of time spent on tuning and refining control parameters of the algorithm using Method Three (34.5%) was expected, given the unstable behaviour of the real UAV platform and the simple physics model used in the virtual simulator to model this behaviour. Observations found that using virtual simulation is still important in the later stages of development. It was observed that the majority of the participants (8 participants) constantly switched back to virtual simulation for validating the correctness of the algorithm upon small changes made to the code, and for experimenting with different object following strategies. P8 pointed out that the time for using Method Three was precious since it required more time to set up and also consumed battery resources, and therefore, Method One was a cheaper and faster alternative option for testing.

The participants' recommendations collected in Section four of the questionnaire further confirmed the finding on the use of the development methods at each stage of the development cycle. Table 7.4 shows the user preferences if they were to develop a similar robot program in the future. Method One was the most preferred method for initial implementation and debugging, while Method Three was voted to be more suitable for

Stage	M1	M2	M3	Subtotal
Initial implementation and validation of logic	**16.5%**	1.0%	3.5%	21.0%
Debugging and fixing errors in code	**7.8%**	0.6%	6.1%	14.5%
Evaluating performance of algorithm	4.5%	3.0%	**14.5%**	22.0%
Tuning and refining parameters	6.5%	1.5%	**34.5%**	**42.5%**
Total				100%

Table 7.3: Time distribution in using the three development methods (Method One: M1, Method Two: M2, Method Three: M3)

Stage	M1	M2	M3
Initial implementation and validation of logic	**10**	1	0
Debugging and fixing errors in code	**9**	2	2
Evaluating performance of algorithm	6	2	**9**
Tuning and refining parameters	2	2	**9**

Table 7.4: Users' recommended development methods for approaching each stage of the development cycle.

evaluating performance and tuning parameters. The results supported hypothesis 6 and the first part of hypothesis 2. Note that more than half of the participants also recommended to use Method One for evaluating performance of the algorithm, conforming to the observations and reasons described earlier. Out of the 4 participants that used Method Two, half of them recommended the method for the last three stages of development. However, the finding was not significant to draw conclusion on the appropriate use of Method Two in this user study.

On top of the above findings, it is believed that variations between the participants also have an influence on the way they approach the task and the development methods they decided to use. It may be necessary to conduct a user study with a larger pool of users in the future to verify these findings.

Monitoring UAV Operations

Despite the availability of the monitoring interfaces and the presence of the user study conductor, UAV crashes were unavoidable when using Method Three due to the platform's fast and unpredictable movements. They were mainly caused by the high instability and untuned control behaviour of the robot during the object following task which led the robot to fly outside the experimentation area and resulted in failure of the vision based Position Control component. Nevertheless, safety strings set up in the laboratory prevented the UAV to cause any serious damage to the surrounding environment. The two crashes resulted in replacement of a number of propellers and one motor.

Reviewing the recorded desktop videos identified that all of the participants relied on

the first person AR view for evaluating the robot's performance, either by focusing on the MR robot simulator's AR interface or the Object Detector's window which displayed the same view. This enabled them to evaluate whether the robot was able to achieve the goal of the task, i.e. keeping the target object in the view, while ensuring the aerial map remained in the view for the Position Control component to function correctly. It was observed that for participants who used the AV interface (4 participants), they mainly chose the free-look camera and/or the tethered camera mode. However, 3 participants commented that through the interfaces they could see something was wrong in the UAV system but reacting in time to prevent a crash was another problem and required experience. Again, this suggested visualising position commands in the simulation environment could help with the monitoring task by enabling users to foresee planned waypoint positions and take actions to prevent system failures. Another comment was the need to provide a more detailed virtual environment reflecting all real world objects in the AV interface so that the users could better understand how it corresponds to the actual physical experimentation space. Note that the tradeoff for implementing this suggestion is the increased amount of graphical modelling effort required.

Development Method Experience

The questionnaire results on the development method experience are shown in Figure 7.23. Method One and Method Three were rated by all the participants, and Method Two was rated by the 4 participants who used this development method. As Method Two used a robot that did not respond to commands, it had very different characteristics from the other two conditions. For this reason, statistical tests were only performed to identify significant differences between Method One and Method Three. However, user ratings on Method Two also helped to provide an indication of how the method fits the relating hypotheses. Method One and Method Three data were compared using Two-tailed Wilcoxon Signed Rank tests.

In comparison to the previous user study involving a slow-moving robot manipulator (see Section 7.2.4), using the real UAV platform in Method Three had a greater impact on the safety of the experiment. The participants rated Method One significantly higher than Method Three for helping to keep negative consequences of failures to minimal ($Z = -2.20$, $p < 0.05$), and being safe for the developer ($Z = -2.87$, $p < 0.05$). This was expected as stated in hypothesis 3. Nevertheless, the graph shows that the average ratings for Method Three in terms of safety were still overall positive.

The results did not indicate that Method One had a significantly shorter learning curve than Method Three ($Z = -1.54$, $p = 0.12$). However, it should be noted that Method Three required more preparation steps, such as initialising AR and starting an extra Position Control process, which took more effort for the users to get the simulation started and

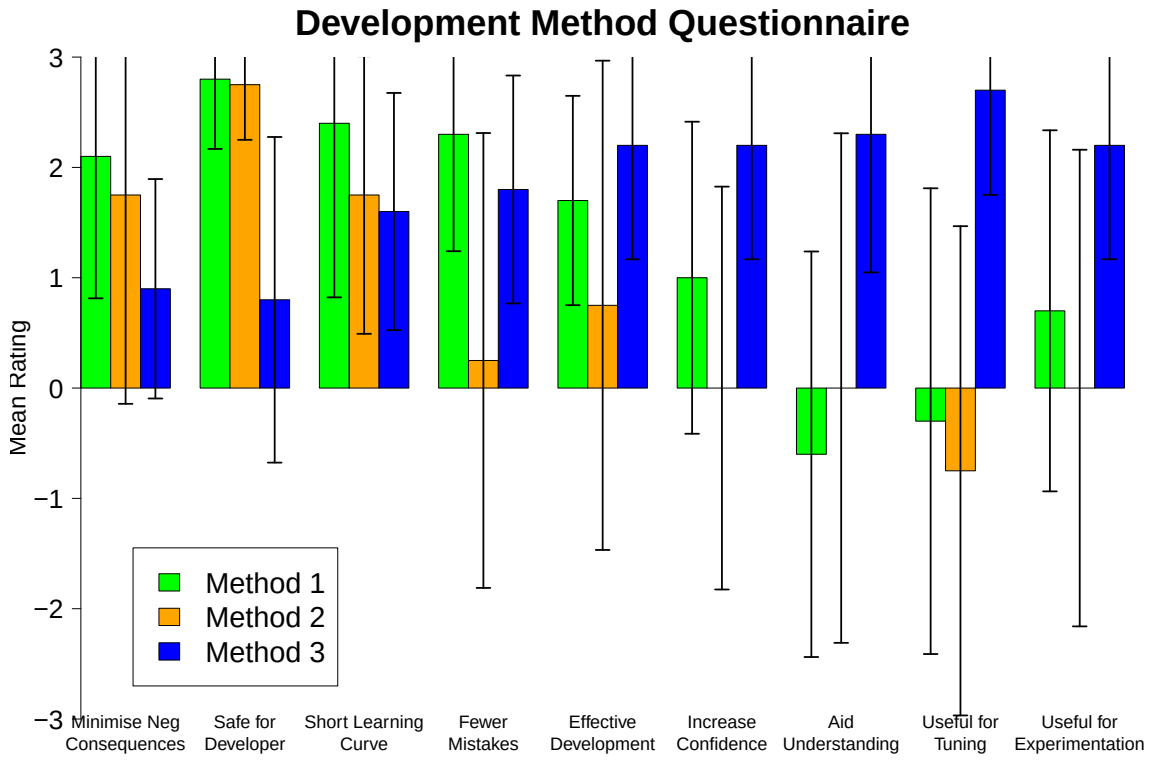


Figure 7.23: Users' experiences in using the three development methods.

the UAV in flight for testing.

The results did not indicate that Method Three was significantly more effective than Method One (Z = -1.00, $p = 0.32$). Similarly, there was no significant difference between Method Three and Method One for giving users greater confidence (Z = -1.56, $p = 0.12$), or making fewer mistakes in the development process (Z = -0.91, $p = 0.36$). Consequently, the results did not support hypothesis 7. However, observations and post-study video analysis identified that an unexpected hardware influence on the UAV behaviour in test flights in Method Three may have led some participants to create mistakes during development. It was observed that after transitioning from Method One (and Method Two for participants who used this development method) to Method Three, the majority of the participants found their algorithm did not produce reliable robot performance; the UAV mainly flew at an x-offset behind the target and eventually lost the target after a short period. Initially, the participants thought there were errors in their algorithm logic, and mistakenly began to make incorrect changes to their code before performing additional testing using Method Three. However, the true cause for this UAV behaviour was due to the effect the camera cable had on the x-movement of the UAV. This effect was not simulated in Method One which caused the participants to be confused on the actual reason for the unexpected UAV behaviour. After spending time experimenting and

studying the behaviour of the real UAV using Method Three, 7 out of the 9 participants who completed the task were able to isolate this problem and modified their code to compensate for the hardware effect. The other 2 participants chose to modify the control parameters to either fly at a higher altitude for a larger view of the scene, or at greater speeds (by giving distant position commands) to complete the task. In the later interview, 5 participants reported to have experienced frustrations during this time frame.

The participants rated Method Three significantly higher than Method One for understanding the UAV flight behaviour as expected ($Z = -2.57$, $p < 0.05$) since it involved the real UAV platform in simulation. The finding could change if a high fidelity FDM was integrated into the virtual simulation. This needs to be investigated in the future.

Method Three was rated significantly higher than Method One for tuning control parameters ($Z = -2.81$, $p < 0.05$). The findings agreed with the development method time distribution data analysis and the user recommendations, suggesting Method Three was more suitable for the later stages of the development; this further supported hypothesis 6.

Statistical analysis did not suggest that Method Three was more useful than Method One for experimenting with different flight maneuvers ($Z = -1.80$, $p = 0.07$), though the result was approaching significance. While both average ratings were positive, there was high variance in Method One ratings, thus the results could not fully support the second part of hypothesis 2.

Method Two was found to be unsuitable for tuning, which rejected hypothesis 5. P1 and P8 commented later in the interview that Method Two was useful for *identifying* (not tuning) initial control parameters, which were later tuned using Method Three.

7.3.5 Summary

This case study evaluation focused on creating an MR simulation for the development of a prototype UAV system. The MR simulation used the extensible AR tracking technique for augmenting the robot's onboard vision data and has been shown to provide registration of virtual objects in the robot's view for the cow monitoring task. The use of the MR simulation enabled repeatable tests to be efficiently created for evaluating the performance of the UAV system. It helped to provide insights to the UAV flight behaviour and its performance in completing the task, which suggested improvements to the control components of the UAV system.

The user study conducted involved computer programmers implementing the Object Following component of the UAV system and testing the system using three development methods: 1) virtual simulation, 2) MR simulation with a real camera (on a manually controlled UAV), and 3) MR simulation with a real UAV and camera. Overall, the results reveal that Method One and Method Three were essential to the development process. The two methods were also complementary, each contributing to different stages of the

development cycle. Despite the simplicity in the design of the virtual simulation, it served as an important starting point for the users, and also a safe alternative for testing their algorithms. The MR simulation contributed significantly to the later stages of the development and helped users understand the UAV flight behaviour and tune its control parameters for better robot performance.

As the development of the UAV system is still at the prototyping stage, there are issues to consider before transferring the results to the final case where the UAV is used on a real farm. Work is required to optimise the control algorithms of the UAV system to operate onboard so that it can process camera images directly, thus removing the long cable to the PC. Moreover, additional testing must be carried out to evaluate the UAV system navigating in a larger experimentation area. MR simulation can again be used for this testing, by simulating virtual cows, and potentially obstacles such as trees and power lines, in environments such as a gymnasium or even a large scale UAV testbed [175].

7.4 Discussions

Through the case study evaluations, the MR robot simulator has been shown to be a robust working system after being used by various robot operators and developers for the development of different robot systems. MR simulations have been demonstrated to benefit various aspects of the robot development process, particularly providing a solution to the three main issues stated at the beginning of this chapter:

1) Reliability of Simulation Results: The first two case studies quantitatively validated the MR simulations by comparing the robot behaviour in simulations with real world tests. In both cases, the robot trajectories generated by the MR simulations have been found to produce an error of less than 2.0% with respect to those in the real experiments. The user study in case study two has also identified significant findings indicating that the users were positive about MR simulation in terms of its realism and the reliability of the simulation results. The last case study focused on prototyping an agricultural UAV system where comparisons with real world tests were not yet possible. Nevertheless, it was shown that the use of MR simulation was able to provide valuable insights to the robot behaviour, which helped to identify deficiencies in the UAV system before advancing to real world tests in an actual agricultural environment. These insights include the flight instability resulting from the linear control of the Object Follower component, and the undesired effects of the cable on the flight dynamics of the UAV system.

2) Reusable Solution: This chapter has demonstrated that the MR robot simulator is reusable in three different applications. The MR robot simulator has been used to simulate a ground robot in a search and rescue operation, a robot manipulator system for an industrial building interior demolition task, and a UAV system for an agricultural

monitoring task. The MR robot simulator takes advantage of the physics engine and existing sensor model available in the underlying simulation platform Gazebo, as well as open source kinematics libraries in building the MR simulations described in this chapter. The robot software frameworks also played a part in connecting the MR robot simulator with different robot platforms and other hardware resources. It is believed that the MR robot simulator would continue to broaden its application domains as these open source communities grow larger day by day.

3) Improvement to Safety: The use of MR simulations has addressed concerns regarding safety in the development process to some extent. Exploratory trials in the first case study have observed that the use of MR simulations prevented actual damage to the real robot in the search and rescue scenario when it collided with a small virtual object on the ground which the laser was not able to detect. The benefit of this use of MR simulation would magnify as later in the future it is used to test the obstacle avoidance component on the aerial robot. Similarly, simulating a virtual target for testing the UAV system helped to minimise the consequence of failures, which would otherwise have been severe if physical targets were used in the experiments. However, it must be noted that the degree to which the use of MR simulation can improve safety is also dependent on the task and the real components included in the simulation. The user study results have indicated that the inclusion of the real UAV system in the MR simulation had a significant impact on safety, but this was not found to be an issue in the MR simulation involving the industrial robot manipulator system. Additionally, it was observed that while the MR interfaces helped users in monitoring the UAV behaviour, the users could not foresee future events. Work is still required in designing additional visual aids in the interfaces that are able to actively alert users of potentially hazardous situations.

Other Insights from User Studies: User study results have provided initial insights to the comparison between the AR interface and the AV interface for robot development. It was found that the fixed, exocentric AR view was significantly more intuitive than the AV view with a free-look camera for the screw remover operation task. However, this may be due to usability issues of the AV interface control tools. The findings suggest that improvements to the interaction method may be necessary to enable the users to control the viewpoint in a more natural and intuitive way.

Other insights have also been drawn from the observational user study. The benefit of MR simulations is most prominent after an initial implementation has been developed. It was shown in the last user study that an MR simulation involving the real robot was complementary to virtual simulation, and enabled the users to comfortably transition to a more physical environment. Building the virtual simulation and the MR simulation on the same simulation platform also allowed users to easily switch from one method to another without learning to use separate controls and interfaces, which facilitated a

smoother development process.

8

Conclusions and Future Work

8.1 Conclusions

This thesis identifies open problems faced throughout the robot development process, specifically focusing on development methods and environments that are used. The analysis of the aerial robot development cycle in Section 3.1 has revealed three important issues that can not yet be effectively solved using existing approaches: 1) the lack of smooth and reliable transition from simulation to reality, 2) the various custom solutions and tools that exist which have minimal reusability and interoperability between robot systems and applications, and most importantly, 3) the considerable time, cost, and safety requirements in real world tests remain an issue in the development of complex and expensive robot systems. The analysis has led to a set of requirements for improving robot development, which suggested a common and standardised simulation environment capable of providing users a more flexible and controlled way of experimentation. Visualisations for helping users understand and monitor the safety of experiments were also considered important.

The solution proposed in this thesis is MR simulation. The concept of merging the real and the virtual worlds opens up a whole new dimension for robot simulations. Based on the HIL simulation paradigm, it enables the users to design simulations involving real and virtual objects that co-exist and interact in real time. MR simulations enable users to:

- replace dangerous components in the experimental setup with safe and virtual counterparts,

- cost-effectively simulate resources that are expensive or unavailable and observe them seamlessly interact with real components,

- flexibly vary the amount of real and virtual components over the development process for more reliable results,

- efficiently create different complex test scenarios and environment configurations (of real and virtual components) for experimentation.

These benefits facilitate an incremental development cycle where software and hardware for different components can be developed and tested before a physical prototype is ready.

One of the main contributions of this research is a generic MR framework. The most novel elements of the MR framework are an entity modelling method that takes into account an object's degree of physical and functional virtualisation, and a behaviour-based interaction scheme for creating interactions between real and virtual objects. The core framework has been purposely designed to be generic to enable standardised implementation of MR systems. This presents opportunities for extensions and enhancements in future by other domain experts in the MR community. The core framework, which integrates the two novel elements, provides formalisation of a) the static representation of the environment with the use of scene graphs, and b) the dynamic interaction between entities in the environment formulated in terms of state transitions and constraints. An extension of the framework to robot simulation has been demonstrated. Compared to existing hybrid simulation methods which predominantly focus on enabling robots to sense virtual objects, the MR framework creates richer interactions in simulations. In addition to interactions through sensory augmentations of robots, the novel interaction scheme introduced in this work is able to model physical interactions between arbitrary real and virtual entities in the environment. This enables simulation of robot tasks involving collisions and manipulations, which were not possible in existing hybrid simulation tools. The MR framework has been validated in case study evaluations which demonstrated successful real-virtual interactions between the robot and the environment; these include a real robot colliding with virtual obstacles, physical contacts between a real robot manipulator and virtual target objects, and interactions with virtual objects through real laser, force-moment, and vision sensors. A set of guidelines have also been provided to help users design MR simulations with an appropriate degree of virtualisation and realism to meet their task requirements.

Another contribution to the robotics community is an MR robot simulator which has been implemented based on the MR framework. The MR robot simulator is unique as it builds on a general purpose virtual robot simulator to support simulations of a wider range of robot systems. The MR robot simulator has been demonstrated to be a robust working system throughout the case study evaluations. A notable aspect of the implementation is in its design to accommodate different robot software frameworks. The thesis shares the same goal as open source robot software frameworks such as Player and OpenRTM to promote standardisation, better reusability of software code, and interoperability between heterogeneous components in robot development. Therefore, the MR robot simulator integrates these robot software frameworks in its design to support simulations for a wider range of robot systems. This is especially motivated by the outstanding issue in the aerial robotics community where sharing of technologies is difficult since custom in-house solutions are still the preferred choice in various research groups.

An important advantage of applying MR to robot development is the benefit of its AR and AV visualisation techniques. Two visual interfaces have been designed to accompany the MR robot simulator, namely the AR and AV interfaces. The AR interface provides users an intuitive view of the MR simulation environment by visualising complex robot and task relevant data in context with the real world in real time. The AV interface has been proposed to compensate for the limitations of AR, by providing flexible views of the simulation environment where the AR interface can not cover. This helps to improve the user's awareness of simulation events. Their use and benefits have also been demonstrated in the case study evaluations of the MR robot simulator.

This research has placed a strong focus on applying AR visualisations to robotics. Not only does this help in creating suitable AR visualisations for MR simulations, but the work can provide insights and new solutions to future research in the area of AR for robotics in general. The requirements analysis has identified the need for an efficient and robust tracking technique to create continuous and reliable AR as well as an extensible tracking solution to deploy the system in potentially large environments. As the result, two AR configurations have been proposed that integrate natural feature tracking techniques. Compared to existing marker based solutions commonly employed in AR systems in robotics, the two AR configurations have the advantage of allowing AR visualisations to be created in unprepared environments where robots commonly operate in today. The first AR configuration is designed to be simple, providing a good balance between speed and accuracy for AR in small environments or from cameras with small changing viewpoints, while the second AR configuration is computationally more expensive but extensible to explore unknown scenes.

An important issue identified in the literature review is the lack of validation of robot simulators, especially HIL/hybrid simulation tools. A significant effort has been put into

the evaluation of the MR robot simulator. MR simulations have been investigated for use in the development of three robot systems in three different applications. The first case study assessed the use of MR simulation for a robot search and rescue scenario. It evaluated the MR robot simulator as a whole, demonstrating the ability of the MR robot simulator to accurately mix virtual obstacles into the real physical setup for testing the navigation component of a wheeled ground robot. Quantitative evaluation has found that MR simulation yields closer results to the real experiment in comparison to virtual simulation. The evaluation has also demonstrated the combined use of the AR and AV interfaces to provide real time visual feedback to the user, highlighting possible causes of damage to the robot.

The second case study deployed the MR robot simulator for testing an industrial robot manipulator system that is used for removing screws in the building interior renovation process. The evaluation has shown that the MR simulation created was able to minimise resource requirements during development while facilitating a reliable transfer of results to the real experiment. User study results have also indicated the MR robot simulator's improvement over virtual simulation in terms of simulation realism and reliability of results, and confirmed the benefits of the AR interface. A number of improvements were also identified. It was found that a number of users expressed the need for a movable viewpoint in the AR interface, and a more natural interaction method in the AV interface.

The last case study created an MR simulation for the development of an agricultural UAV system. The evaluation has shown that MR simulations enable different scenarios to be created to test the performance of the UAV system in a cow monitoring task. The extensible AR tracking technique has also been demonstrated to successfully cope with the erratic motion of the UAV platform and augment virtual information in the robot's vision data for creating MR interactions. User study results have indicated a significant contribution of the MR robot simulator to the evaluation and tuning of robot systems in the later stages of development. The findings have provided an insight to the complementary nature of virtual simulation and MR simulation during robot development, and suggested the MR robot simulator facilitated a smooth development process.

In summary, MR simulations can improve the robot development process by providing a more standardised and reusable simulation environment for the incremental development of both simple and complex robot systems in different applications. The open source nature of the implemented simulation tool promotes better sharing of knowledge and technologies from simulation users, designers, and developers across different fields of robotics. With improved interoperability, MR simulations also encourage collaborations between robot developers from different groups, enabling them to create and test robot systems in parallel without depending on the actual hardware or software components. The work leads to a more efficient and cost-effective robot development process.

8.2 Future Work

The work presented in this thesis suggests a number of areas for improvements and further research. They are organised into different levels as follows.

8.2.1 Concept

The MR framework serves as the basis of all MR robot simulations shown in this thesis and have been demonstrated to be promising. Nevertheless, the framework needs to be extended to take into account time- and safety-critical properties of robot operations in its modelling. The current MR framework makes interactions happen but relies on the implementation to ensure that they happen within strict time and safety requirements. Extension of the framework to model and enforce these properties is considered important for more standardised implementations of MR systems.

The core MR framework, having been designed to be generic, now requires effort in validating the extent to which it is able to meet the different needs of users in the MR community. To answer this question, an exhaustive survey and requirements analysis may be necessary, but it is beyond the scope of this research. Future work in applying and validating the framework in other fields of MR is encouraged.

8.2.2 Implementation

A number of features for building an enhanced development environment were discussed in Section 3.3 but one was not implemented during this research – a robot-specific integrated development environment. This feature was given a lower priority but the idea proposes a valuable extension of this research to integrate with the wealth of work in the area of robot programming.

This thesis has also investigated the associations between data types and visual representations for effective visualisations. However, the implemented MR robot simulator provides a limited number of data visualisation options to visualise robot data. Work is necessary to implement and compare different combinations of data visual representations for assisting robot development, such as the visualisation designs for robot development proposed in [157] and [79], and identify their impacts on the user's experience in using the MR robot simulator.

The types of sensor and actuator based interactions implemented were also limited by the models available in the underlying simulation platform. Sensors such as thermal and sound are not yet supported. A wider range of sensor and actuator models need to be implemented so that the MR robot simulator can be used and evaluated on its ability to simulate robot systems relying on these interfaces for interaction.

8.2.3 Evaluation

The evaluation on the MR robot simulator has yet to fully consider issues related to the installation and preparation of the MR simulations. Creating MR simulations requires physics and graphical modelling, and setting up the simulation environment, all of which influence the acceptance of the technology in the industry. Work is necessary to evaluate the actual gain of using the MR robot simulator over the cost of installing the system and creating MR simulations.

More importantly, an in-depth, long term user study is necessary. In the user studies presented in this thesis, each user trial was held for a maximum of a few hours. It is difficult to draw conclusive remarks based on a study of such short periods. The robot development process is typically long that could span weeks or months, and large projects could take years. A long term user study, potentially a ethnographic study, would help to verify the contributions of the MR robot simulator to the different stages of the robot development cycle. For example, a long term study could investigate how users would design, build, and run MR simulations themselves based on given requirements analysis, then identify the ease and difficulties faced throughout the process.

8.2.4 Usability

Accurate AR visualisation in dynamic environments remains an unresolved problem. While the AR interface of the MR robot simulator integrates state-of-art AR tracking technologies, the performance degrades when they are deployed in dynamic environments with considerable noise and lighting changes. Discontinuities of AR visualisation due to tracking failures deteriorate the simulation experience of the user, especially during important simulation events. More robust tracking solutions are needed. This is a known issue which is actively being addressed in the robotics, computer vision, and AR communities.

A more intuitive user interaction method is necessary to improve the usability of the MR robot simulator. The mouse and keyboard interfaces were found to be difficult to use to navigate in the simulation environment, and more natural user interaction metaphors were desired. Future work may be required to investigate between the impact of different interaction paradigms on the usability of the MR robot simulator, from conventional desktop configurations to immersive environments such as with the use of HMDs, wearable input devices, or projector displays. It may also be necessary to determine whether the preferred interaction paradigm varies between MR simulations of different robot tasks.

A
Videos

A number of videos have been included on a CD-ROM accompanying this thesis. These videos have been referenced throughout the thesis, and they help to illustrate the visualisations created, and demonstrate the results achieved. The videos are listed below along with the sections that they were referenced in.

- `AVInterfaceDemo.mp4` – Section 5.4.1
- `ObstacleAvoid_RealRobot.mp4` – Section 5.5.1
- `ObstacleAvoid_VirtualRobot.mp4` – Section 5.5.1
- `Collision_RealRobot.mp4` – Section 5.5.1
- `ARConf1Recovery.mp4` – Section 6.2.4
- `ARConf2Results.mp4` – Section 6.3.3
- `ScrewRemover.mp4` – Section 7.2.2
- `UAVMonitoring.mp4` – Section 7.3.3

B

Evaluation of Existing Robot Simulators

Shown on the following pages is a table containing a list of robot simulators and the criteria for evaluation. The critera are chosen by considering the essential features outlined in Section 3.3, as well as common features offered in robot simulation tools, and requirements for integrating MR. The table has been put together by collecting information from published papers, official websites, manuals, and also the author's experience in using the simulation tool (if the tool is publically available).

Note that many of the robot simulators are actively being developed, and thus information may change from the time when the evaluation was conducted. The table was updated on January, 2010.

Keys:
O - Yes
X - No
? - Information not found
1 - Lowest, 5 - Highest

	Simulator	URL	OS			Programming Language	Documentation (1~5)	Source	Community Support (1~5)	Built-in IDE
			Windows	Linux	Mac					
	Mobile Robot Simulator:									
1	Dave's Robotic Operating System	http://dros.org/	X	O	?	C/C++, Perl	2	O	1	X
2	Biorobotic Simulation Program	http://www.life.uiuc.edu/delcomyn/RSSimSoftware.html	X	O	?	Config++, C++	1	?	1	X
3	Carmen's Simulator	http://carmen.sourceforge.net/	X	O	?	C	2	O	2	X
4	Darwin2K	http://darwin2k.sourceforge.net/	X	O	?	C++	2	O	2	X
5	EyeSim	http://robotics.ee.uwa.edu.au/eyebot/doc/sim/sim.html	O	O	?	C	1	?	1	X
6	Gazebo	http://playerstage.sourceforge.net/index.php?src=gazebo	X	O	O	C/C++, Python	3	O	4	X
7	Marilou Robotics Studio	http://www.marilou-roboticsstudio.com/	O	X	X	C/C++, CLI, C#, J#, VB#	3	X	3	O
8	Microsoft Robotics Studio's Simulator	http://msdn.microsoft.com/en-us/robotics/default.aspx	O	X	X	VB.Net, C#, JScript, Python	4	X	3	O
9	Mobile Robots MobileSim	http://robots.mobilerobots.com/wiki/MobileSim	O	O	O	C++	2	O	2	X
10	MobotSim	http://www.mobotsoft.com/	O	X	X	BASIC	1	X	1	O
11	OpenHRP	http://www.openrtp.jp/openhrp3/	O	O	O	C/C++	3	O	2	O
12	OpenRAVE	http://openrave.programmingvision.com/	O	O	O	C/C++, Python, Matlab/Octave	4	O	4	O
13	OpenSim	http://opensimulator.sourceforge.net/	X	O	?	C++	2	O	1	X
14	RoboCup Soccer Simulator	http://sourceforge.net/apps/mediawiki/sserver/	O	O	O	C++	2	O	3	X
15	RoboWorks	http://www.newtonium.com/	O	X	X	VB	1	X	1	X
16	Rossum's Playhouse (RP1)	http://rossum.sourceforge.net/sim/index.html	O	O	?	C/C++, Java	2	O	3	X
17	Simbad	http://simbad.sourceforge.net/	O	O	O	Java, Python	2	O	3	X
18	SimRobot	http://www.informatik.uni-bremen.de/simrobot/	O	O	O	C/C++	2	O	2	O
19	Stage	http://playerstage.sourceforge.net/index.php?src=stage	X	O	O	C/C++, Python	4	O	4	X
20	UCHILSIM	http://www.robocup.cl/uchilsim/	O	X	X	C/C++	1	X	1	X
21	USARSim	http://usarsim.sourceforge.net/	O	O	?	C/C++	4	O	4	X
22	Webots	http://www.cyberbotics.com/products/webots/index.html	O	O	O	C/C++, Java	4	X	4	O
23	Yobotics!	http://yobotics.com/simulation/simulation.htm	O	O	O	Java	2	X	1	O
	Industrial Robot Simulator:									
24	Camelot's Ropsim	http://www.camelot.dk/	O	X	X	Vendor Native Language	?	?	?	O
25	COSIMIR®	?	O	X	X	?	?	X	?	O
26	EASY-ROB + Famous Robot®	http://www.easy-rob.com/en/easy-rob.html	O	X	X	C/C++	?	?	?	O
27	RobotWorks	http://www.robotworks-eu.com/	O	X	X	?	1	X	1	O
	Flight Simulator:									
28	AeroSim Blockset	http://www.u-dynamics.com/aerosim/default.htm	O	O	X	MatLab/SIMULINK	2	?	?	X
29	FlightGear + TerraGear	http://www.flightgear.org	O	O	O	C/C++	4	O	4	X
30	Microsoft Flight Simulator X	http://www.microsoft.com/games/flightsimulatorx/	O	X	X	-	3	X	4	?
31	RT Dynamics	http://www.rtdynamics.com/	O	O	X	C/C++	3	?	?	X
32	X Plane	http://www.x-plane.com/	O	O	O	-	4	X	4	X

Figure B.1: Evaluation of Robot Simulators, Page 1.

	Extendibility		User Interface		Simulation												Features			
	Plug-in Support	Modifiability (1~5)	GUI	Interactive	Graphics				Physics		Networking	Modelling			Audio		Simulation Models			
					2D	3D	OpenGL	DirectX	Kinematics	Dynamics		Loading	Texturing	Scripting	2D	3D	Wheeled	Legged	Flying	Manipulator
1	O	2	O	O	O	X	X	X	X	X	O	X	X	X	X	X	O	?	x	?
2	?	?	O	O	X	O	O	X	O	O	?	?	?	O	?	?	X	O	X	O
3	O	2	O	O	O	X	X	X	X	X	O	X	X	O	X	X	O	?	X	X
4	O	2	O	O	X	O	O	X	O	O	X	X	X	O	X	X	O	O	?	O
5	?	?	O	O	O	O	O	X	X	O	?	?	?	O	X	?	O	O	X	?
6	O	5	O	O	X	O	O	X	X	O	O	O	O	O	X	X	O	O	?	O
7	O	1	O	O	X	O	?	O	X	O	O	O	O	?	?	?	O	O	?	O
8	O	1	O	O	X	O	X	O	X	O	O	O	O	O	?	?	O	O	?	?
9	O	2	O	O	O	X	X	X	X	X	O	X	X	O	X	X	O	?	X	X
10	?	1	O	O	O	X	X	X	X	X	?	?	?	?	X	X	O	X	X	X
11	O	2	O	O	X	O	O	X	O	O	O	O	?	O	X	X	O	O	?	O
12	O	5	O	O	X	O	O	X	O	O	O	O	?	O	X	X	O	O	?	O
13	O	3	O	O	X	O	O	X	O	O	X	?	?	O	X	X	O	?	X	O
14	?	2	?	?	?	?	?	?	X	?	O	?	?	?	?	?	?	?	?	X
15	X	1	O	O	X	O	O	X	X	X	O	X	X	O	X	X	X	X	X	X
16	O	2	O	O	O	X	X	X	X	X	O	X	X	X	X	X	O	?	X	X
17	O	1	O	O	X	O	X	X	X	X	X	X	X	X	X	X	O	X	X	X
18	?	1	O	O	O	O	O	X	O	O	?	?	?	O	X	X	O	O	X	?
19	O	5	O	O	O	X	O	X	X	X	O	O	O	O	X	X	O	X	X	X
20	O	1	O	O	X	O	O	X	X	O	?	O	O	O	?	?	X	O	X	X
21	O	4	O	O	X	O	O	O	X	O	O	O	O	O	O	?	O	O	O	O
22	O	1	O	O	O	O	O	X	O	O	?	O	O	O	X	X	O	O	O	O
23	?	?	O	O	X	O	X	X	X	O	?	?	O	?	?	?	O	O	X	O
24	?	?	O	O	X	O	?	?	O	?	?	O	?	?	?	?	X	X	X	O
25	?	?	O	O	X	O	O	?	O	?	O	O	?	?	?	?	?	?	X	O
26	X	1	O	O	X	O	O	X	O	O	?	O	?	?	?	?	?	?	?	O
27	?	?	O	O	X	O	?	?	O	X	?	?	?	?	?	?	X	X	X	O
28	?	?	X	X	X	X	X	X	X	O	X	X	X	O	X	X	X	X	O	X
29	O	3	O	O	X	O	O	X	X	O	O	O	O	O	O	?	X	X	O	X
30	X	1	O	O	X	O	X	O	X	O	O	O	O	O	O	?	X	X	O	X
31	O	?	O	O	X	?	?	?	X	O	X	X	X	O	?	?	X	X	O	X
32	O	1	O	O	X	O	O	O	X	O	O	O	O	O	O	?	X	X	O	X

Figure B.2: Evaluation of Robot Simulators, Page 2.

	Features											Environment Simulation	Multi Robot Support	Code Portable to Real Robot	Mixed Reality Requirements		Pricing
	Sensor Libraries									Actuator Libraries	Sensor/Actuator Noise Model				Online Visualisation	Augmented Reality	
	Sonar	Laser	IR	Odomety	Inertial	GPS	Vision	Light	Tactile								
1	O	O	O	?	?	X	X	X	O	?	X	O	?	?	?	X	Free, Open Source
2	?	?	?	?	?	?	X	X	X	O	O	X	?	O	?	X	Free
3	?	O	?	O	?	?	?	?	?	?	?	O	O	O	?	X	Free, Open Source
4	X	X	X	?	X	X	X	X	O	O	?	O	?	X	?	X	Free, Open Source
5	X	X	O	O	X	X	O	X	O	O	O	O	O	O	?	X	Free
6	X	O	O	O	O	X	O	X	O	O	X	O	O	O	O	X	Free, Open Source
7	O	O	O	O	?	O	O	O	O	O	?	O	O	O	?	X	Free/1493+ Euro
8	O	O	O	O	?	O	O	O	O	O	?	O	O	O	?	X	Free/$399USD
9	O	O	?	O	?	?	O	?	?	?	?	O	O	O	?	X	Open Source
10	O	?	?	?	X	X	X	X	X	?	?	O	O	?	?	X	$19-$30USD
11	X	X	X	?	?	X	O	X	O	O	?	O	O	O	O	X	Free, Open Source
12	X	O	X	?	?	?	O	X	?	O	?	O	O	O	O	X	Free, Open Source
13	X	X	X	X	X	X	?	X	X	?	?	O	O	?	?	X	Free, Open Source
14	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	X	Open Source
15	X	X	X	X	X	X	X	X	X	X	X	X	?	?	?	X	$750USD
16	X	X	O	?	X	X	X	X	O	X	X	O	O	O	X	X	Free, Open Source
17	X	X	O	X	X	X	O	X	O	X	X	O	O	X	?	X	Free, Open Source
18	X	O	O	?	X	X	O	X	O	O	X	O	O	X	?	X	Free
19	O	O	O	O	O	X	X	X	O	O	X	O	O	O	?	X	Free, Open Source
20	?	?	?	?	?	X	O	X	?	O	?	O	O	O	?	X	Free (Demo)
21	O	O	O	O	O	O	O	X	O	O	O	O	O	O	O	X	Open Source
22	O	O	O	O	O	O	O	O	O	O	O	O	O	O	?	X	$400, $2700, $4100
23	X	X	X	X	X	X	X	X	X	O	?	O	?	?	X	X	$995USD
24	X	X	X	X	X	X	X	X	X	?	?	?	?	O	X	X	?
25	X	X	X	X	X	X	X	X	?	?	X	O	O	O	?	X	?
26	?	?	?	?	?	?	?	?	?	?	?	?	O	O	X	X	$1150, $4600, $7475USD
27	X	X	X	X	X	X	X	X	X	O	?	?	?	O	X	X	1000+ EURO
28	X	X	X	X	O	O	X	X	X	X	X	X	X	?	X	X	Free
29	X	X	X	?	O	O	X	X	X	O	?	O	X	?	O	X	Open Source
30	X	X	X	?	O	O	X	X	X	?	?	O	X	?	O	X	$50USD
31	X	X	X	X	?	?	X	X	X	X	X	X	O	?	X	X	?
32	X	X	X	?	O	O	X	X	X	?	?	O	X	?	O	X	$29/$500/$1000USD

Figure B.3: Evaluation of Robot Simulators, Page 3.

C

UML Diagrams of MRSim

This appendix presents UML diagrams to illustrate the implementation of MRSim.

C.1 MRSim Class Structure

This class diagram illustrates an example of how classes can be extended from the core components of MRSim. It features behaviour, data, and device classes related to a position entity.

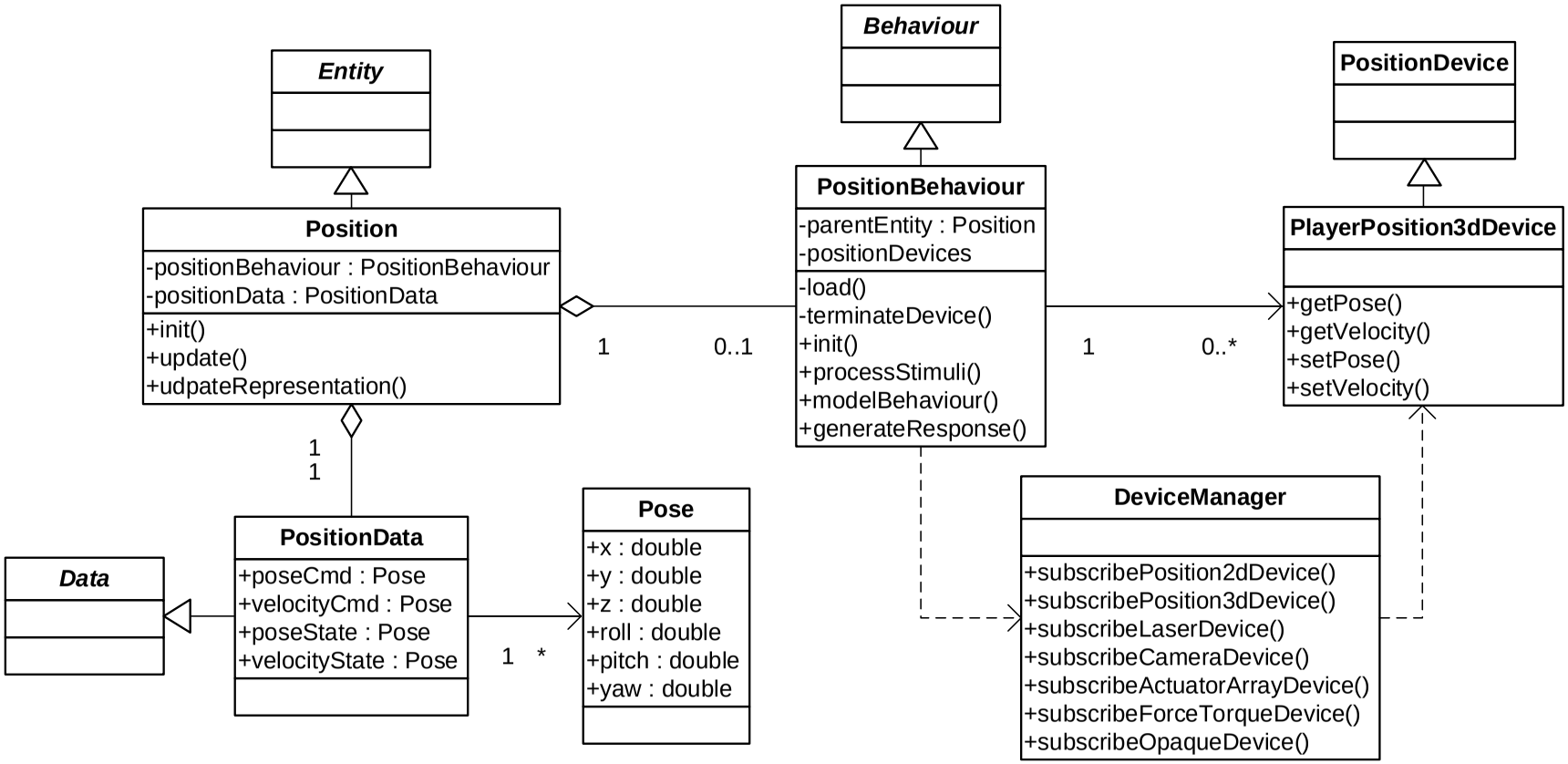


Figure C.1: UML class diagram showing the structure of MRSim, using the position entity as an example.

C.2 Example Initialisation

This sequence diagram illustrates how MRSim is initialised. It shows how the world loads an entity, its behaviour, and associated devices. In this example, a real position entity is loaded in initialisation.

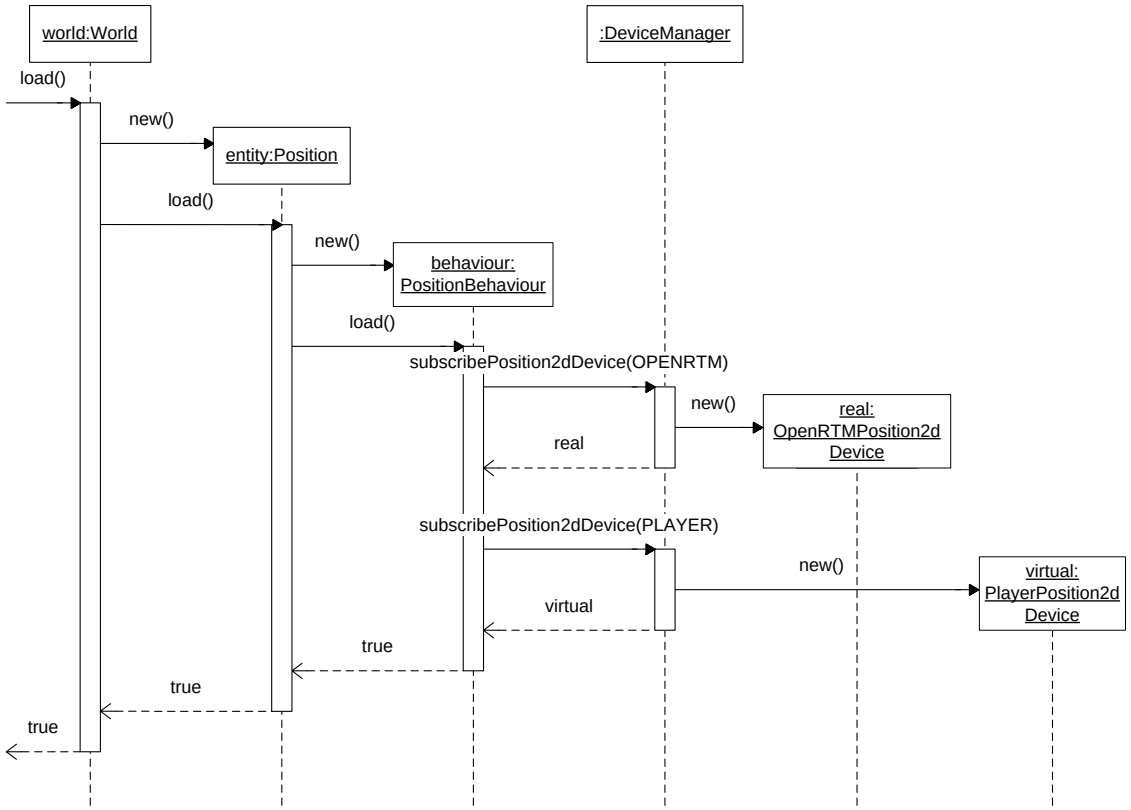


Figure C.2: UML sequence diagram showing how a real position entity, its behaviour, and devices are loaded.

C.3 Behaviour Sequence Diagrams

Sequence diagrams have been created for the position, laser, and rigid behaviours to illustrate how interactions are carried out. There are two sequence diagrams for each behaviour; one describes the behavhiour of a real entity, the other describes the behaviour of a virtual entity.

C.3.1 Position Behaviour

Shown on the following pages

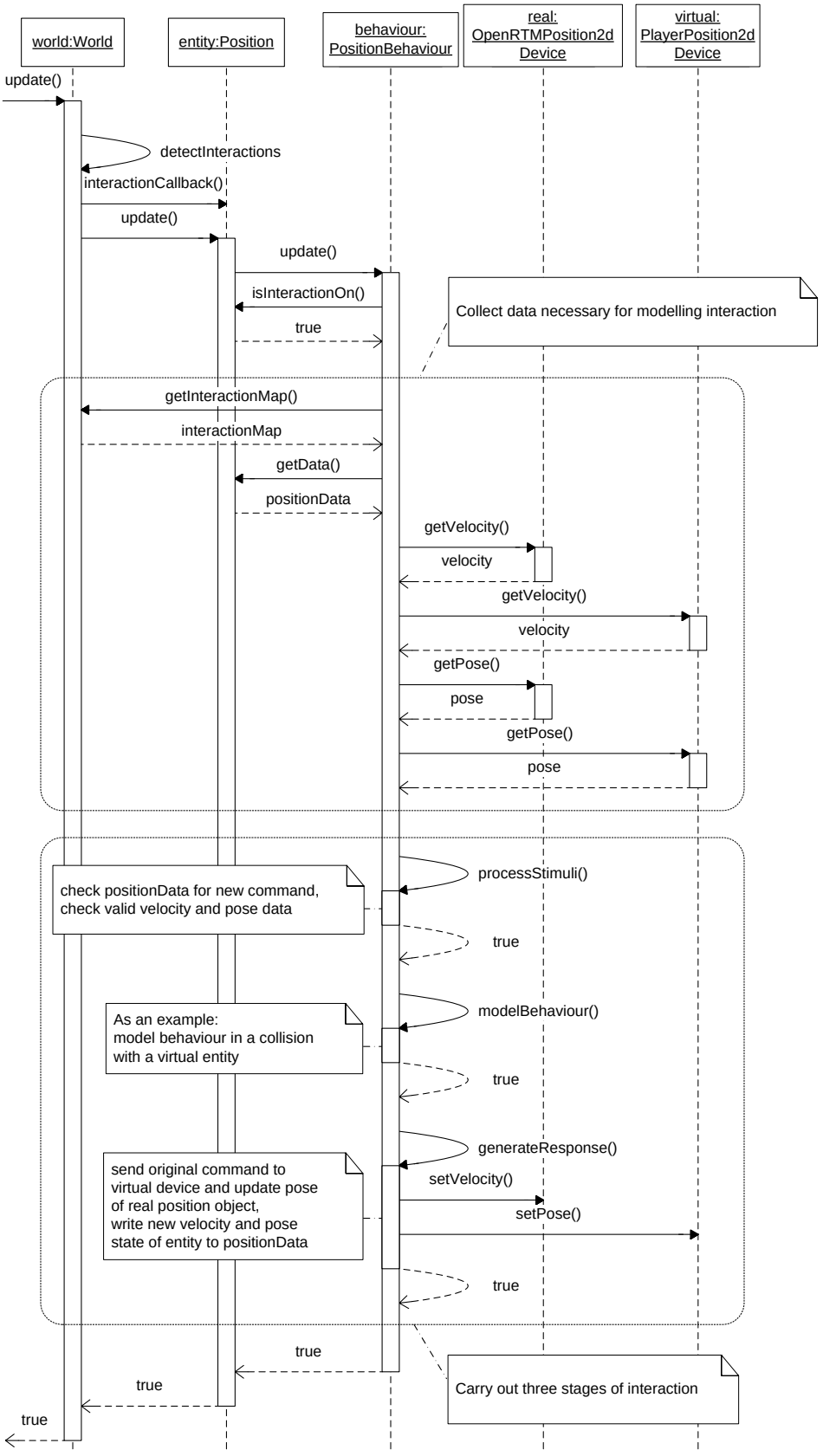


Figure C.3: UML sequence diagram of the position behaviour attached to a **real position** entity.

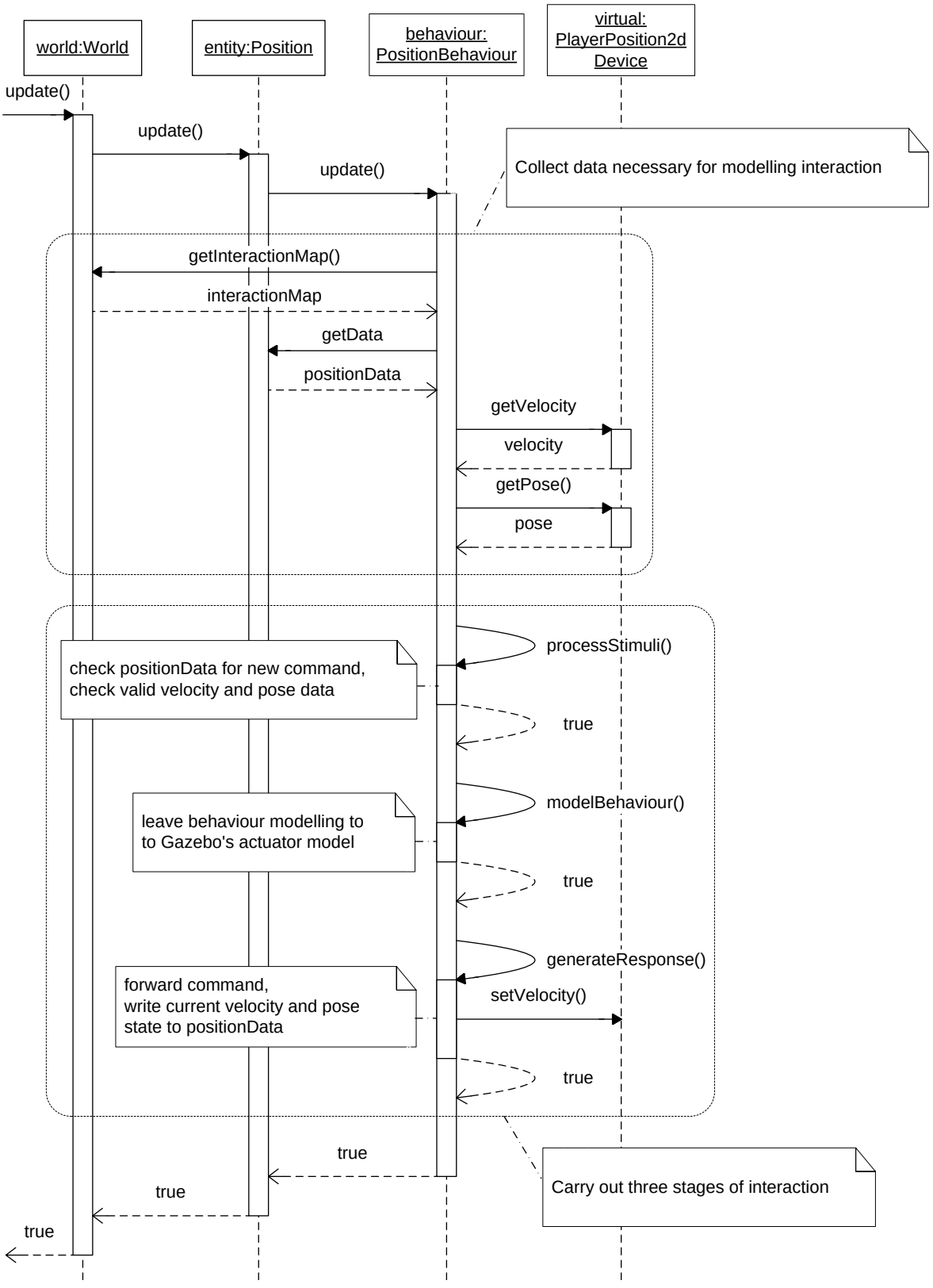


Figure C.4: UML sequence diagram of the position behaviour attached to a **virtual position** entity.

C.3.2 Laser Behaviour

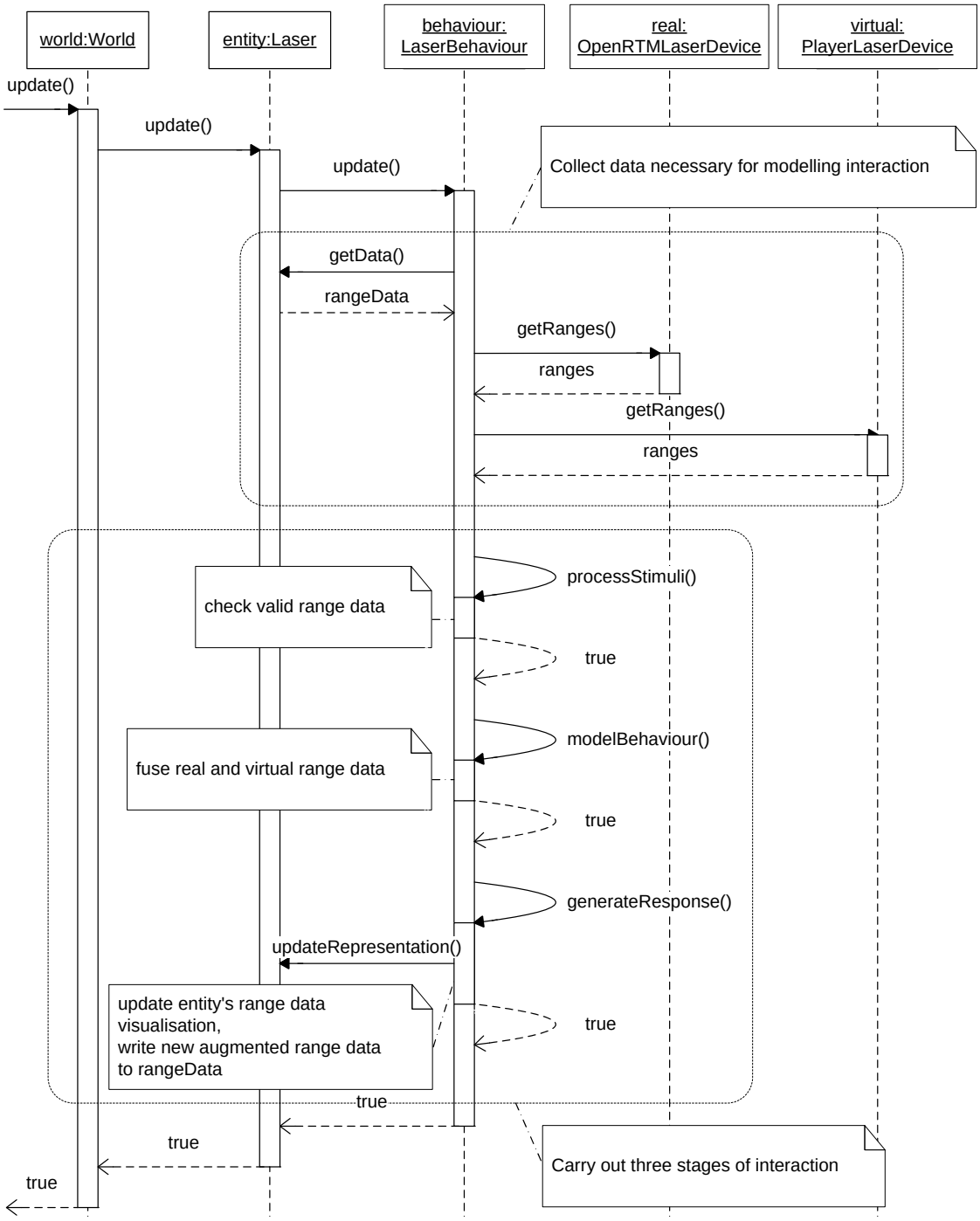


Figure C.5: UML sequence diagram of the laser behaviour attached to a **real laser** entity.

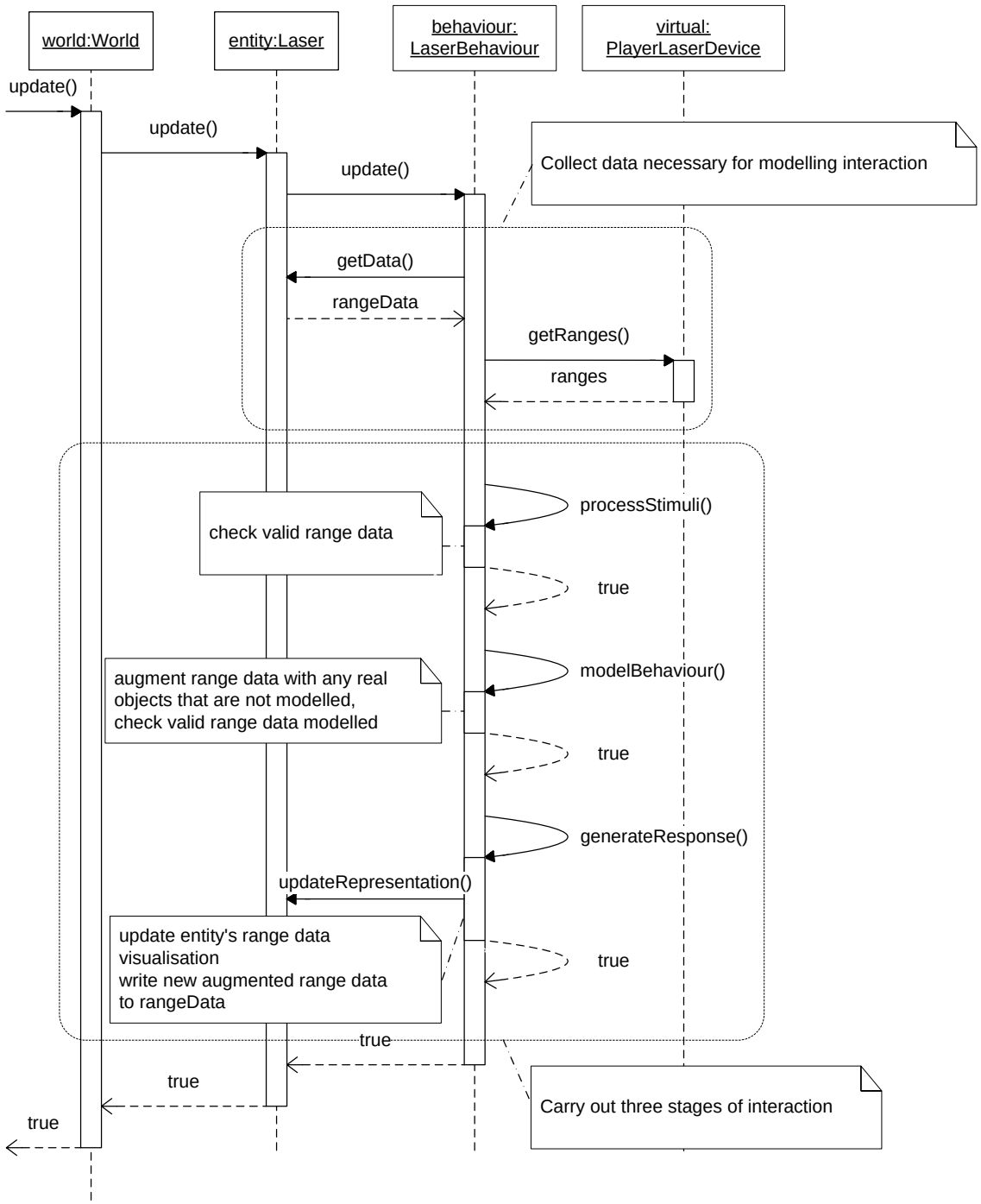


Figure C.6: UML sequence diagram of the laser behaviour attached to a **virtual laser** entity.

C.3.3 Rigid Behaviour

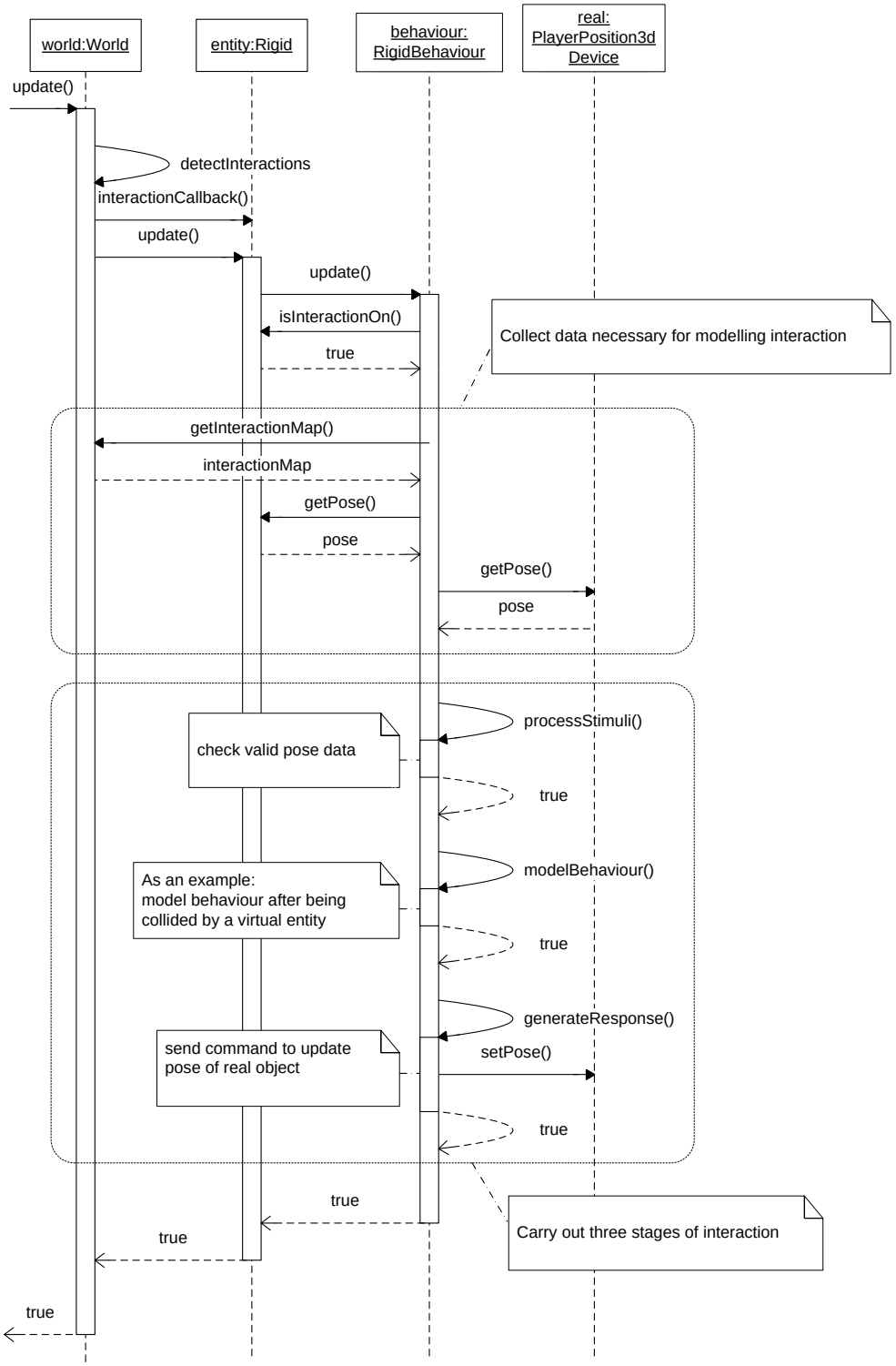


Figure C.7: UML sequence diagram of the rigid behaviour attached to a **real rigid** entity.

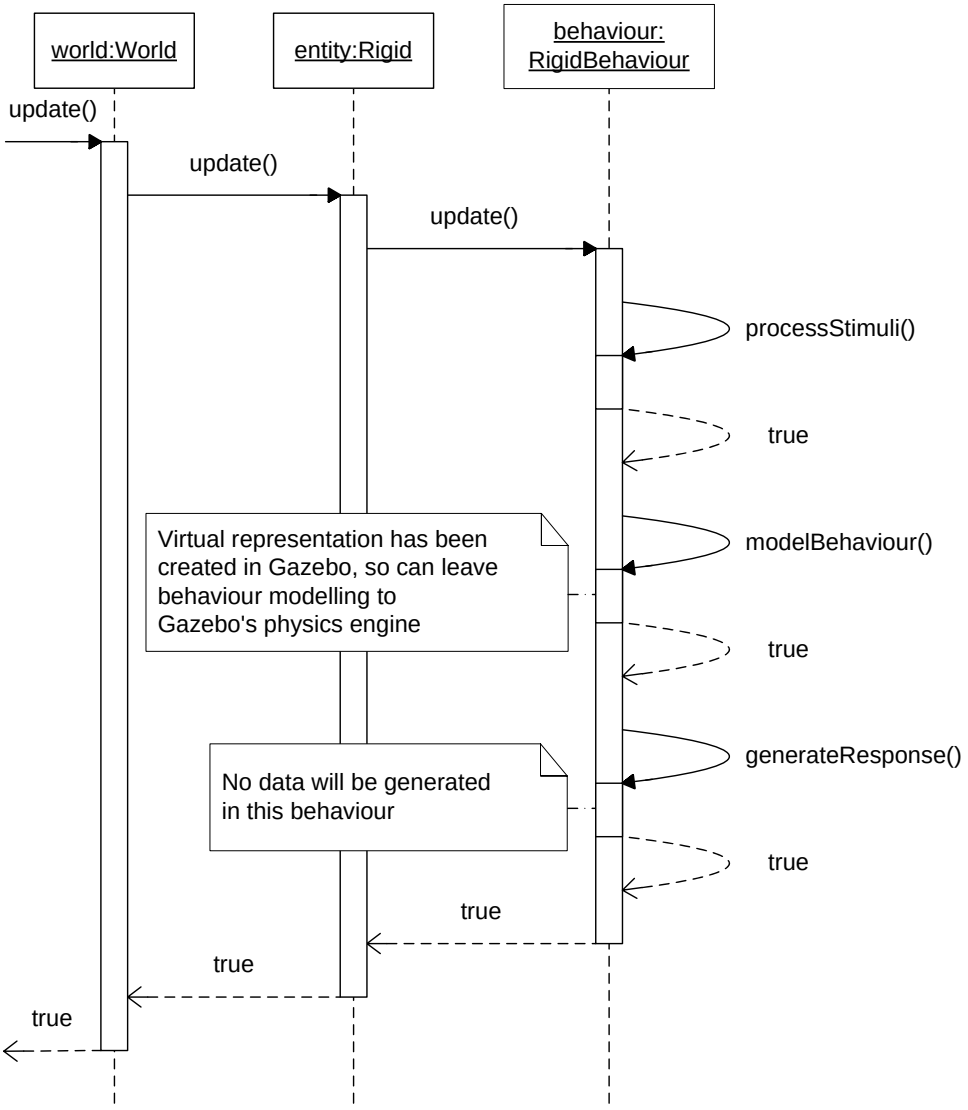


Figure C.8: UML sequence diagram of the rigid behaviour attached to a **virtual rigid** entity.

Bibliography

[1] *The ABB group - automations and power technologies.* http://www.abb.com/.

[2] *AMIRE - Authoring Mixed Reality.* http://www.amire.net/.

[3] *Ascending Technologies.* http://www.asctec.de/.

[4] *Bullet Physics Library.* http://www.bulletphysics.com/.

[5] *CAE.* http://www.cae.com/.

[6] *Carmen robot navigation toolkit.* http://carmen.sourceforge.net/.

[7] *JSBSim - an open source, platform-independent, flight dynamics model in C++.* www.jsbsim.org.

[8] *metaio GmbH.* http://www.metaio.com/.

[9] *OpenHRP - Open Architecture Humanoid Robotics Platform.* http://www.openrtp.jp/openhrp3/en/index.html.

[10] *Robotics domain task force.* http://robotics.omg.org/.

[11] *The rossum project.* http://rossum.sourceforge.net/.

[12] *Unmanned dynamics.* http://www.u-dynamics.com/.

[13] *Virtual collaboration arena.* http://www.virca.hu/.

[14] *Aeronautics. Yamaha Motor*, 2007. http://www.yamaha-motor.co.jp/global/about/business/sky/index.html.

[15] *Issues paper - unmanned aeiral vehicles*, tech. report, Civil Aviation Authority of New Zealand, January 22 2007.

[16] *MXR Software Development Kit*, February 2008. http://sourceforge.net/projects/mxrtoolkit/.

[17] *Statistics New Zealand*, January 2008. http://www.stats.govt.nz/.

[18] *Ageia PhysX*, 2009. http://www.ageia.com/.

[19] *OGRE 3D : Object-oriented graphics rendering engine.*, 2009. http://www.ogre3d.org.

[20] *The Player/Stage project*, 2009. http://playerstage.sf.net/.

[21] *The Object Management Group (OMG)*, October 2009. http://www.omg.org/.

[22] F. Aghili and J.-C. Piedboeaeuf, *Emulation of robots interacting with environment*, IEEE/ASME Transactions on Mechatronics, 11 (2006), pp. 35 – 46.

[23] J. Aleotti and S. Caselli, *Robot grasp synthesis from virtual demonstration and topology-preserving environment reconstruction*, in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, USA, Oct 29 - Nov 2 2007, pp. 2692–2697.

[24] M. Alighanbari, L. Bertuccelli, and J. How, *A robust approach to the UAV task assignment problem*, in Proceedings of the 45th IEEE Conference on Decision and Control, 2006, pp. 5935–5940.

[25] A. Ames, E. Hinman-Sweeney, and J. Sizemore, *Automated generation of weld path trajectories*, in Proceedings of the 6th IEEE International Symposium on Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing, E. Hinman-Sweeney, ed., 2005, pp. 182–187.

[26] P. Amstutz and A. Fagg, *Real time visualization of robot state with mobile virtual reality*, in Proceedings of the IEEE International Conference on Robotics and Automation, vol. 1, 2002, pp. 241–247.

[27] N. Andersen, I. Braithwaite, M. Blanke, and T. Sorensen, *Combining a novel computer vision sensor with a cleaning robot to achieve autonomous pig house cleaning*, in Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference, 12-15 Dec. 2005, pp. 8331–8336.

[28] J. Anderson and J. Baltes, *A mixed reality approach to undergraduate robotics education*, in Proceedings of AAAI-07 (Robot Exhibition Papers), R. Holte and A. Howe, eds., Vancouver, Canada, July 2007, AAAI Press, pp. 1979–1980.

[29] N. Ando, T. Suehiro, and T. Kotoku, *A software platform for component based RT-system development: OpenRTM-aist*, in Proceedings of the 1st International Conference on Simulation, Modeling, and Programming for Autonomous Robots, Berlin, Heidelberg, 2008, Springer-Verlag, pp. 87–98.

[30] ANYKODE, *Marilou robotics studio.*, January 2008. http://www.anykode.com/index.php.

[31] S. Arima, S. Shibusawa, N. Kondo, and J. Yamashita, *Traceability based on multi-operation robot; information from spraying, harvesting and grading operation robot*, in Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics, vol. 2, 20-24 July 2003, pp. 1204–1209.

[32] R. Arkin, *Behavior-based robotics*, MIT press, 1998.

[33] N. Asakawa and Y. Takeuchi, *Teachingless spray-painting of sculptured surface by an industrial robot*, in Proceedings of the IEEE International Conference on Robotics and Automation, Y. Takeuchi, ed., vol. 3, 1997, pp. 1875–1879.

[34] A. Ayyagari, J. Harrang, and S. Ray, *Airborne information and reconnaissance network*, in Conference Proceeding of the IEEE Military Communications Conference, vol. 1, 21-24 Oct. 1996, pp. 230–234.

[35] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, *Recent advances in augmented reality*, IEEE Computer Graphics and Applications, 21 (2001), pp. 34–47.

[36] R. Azuma and G. Bishop, *Improving static and dynamic registration in an optical see-through HMD*, in Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, New York, NY, USA, 1994, ACM, pp. 197–204.

[37] R. T. Azuma, *A survey of augmented reality*, Presence: Teleoperators and Virtual Environments, 6 (1997), pp. 355–385.

[38] ——, *Mixed Reality: Merging Real and Virtual Worlds*, Springer-Verlag, Berlin, 1999, ch. 21, pp. 379–390.

[39] M. Bajura and U. Neumann, *Dynamic registration correction in video-based augmented reality systems*, IEEE Computer Graphics and Applications, 15 (1995), pp. 52–60.

[40] B. Balaguer and S. Carpin, *Where am i? a simulated GPS sensor for outdoor robotic applications*, in Proceedings of the International Conference on Simulation, Modeling, and Programming for Autonomous Robots, Springer, 2008, pp. 222–233.

[41] J. Banks, *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, Wiley-Interscience, 1998.

[42] I. Barakonyi, T. Psik, and D. Schmalstieg, *Agents that talk and hit back: animated agents in augmented reality*, in Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality, 2004, pp. 141–150.

[43] M. Bauer, B. Bruegge, G. Klinker, A. MacWilliams, T. Reicher, S. Riss, C. Sandor, and M. Wagner, *Design of a Component–Based Augmented Reality Framework*, in Proceedings of the IEEE and ACM International Symposium on Augmented Reality. ISAR'01, 2001, pp. 45–54.

[44] H. Bay, T. Tuytelaars, and L. V. Gool, *SURF: Speeded up robust features*, in Proceedings of the 9th European Conference on Computer Vision, May 2006, pp. 404–417.

[45] R. Behringer, J. Park, and V. Sundareswaran, *Model-based visual tracking for outdoor augmented reality applications*, in Proceedings of the International Symposium on Mixed and Augmented Reality, Darmstadt, Germany, 2002, pp. 277–322.

[46] J. Beis and D. Lowe, *Shape indexing using approximate nearest-neighbour search in high-dimensional spaces*, in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, D. Lowe, ed., Puerto Rico, 1997, pp. 1000–1006.

[47] S. Benford, C. Greenhalgh, G. Reynard, C. Brown, and B. Koleva, *Understanding and constructing shared spaces with mixed-reality boundaries*, ACM Transactions on Computer-Human Interaction, 5 (1998), pp. 185–223.

[48] G. Biggs, T. Kotoku, and T. Tanikawa, *Ceiling beam screw removal using a robotic manipulator*, in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, USA, October 11–15 2009.

[49] G. Biggs and B. MacDonald, *A pragmatic approach to dimensional analysis for mobile robotic programming*, Autonomous Robots, 25 (2008), pp. 405–419. 10.1007/s10514-008-9103-x.

[50] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, *Vision based MAV navigation in unknown and unstructured environments*, in Proceedings of the IEEE International Conference on Robotics and Automation, 2010, pp. 21–28.

[51] S. Bortoff, *The University of Toronto RC helicopter: a test bed for nonlinear control*, in Proceedings of the IEEE International Conference on Control Applications, vol. 1, Hawaii, USA, 1999, pp. 333–338.

[52] S. Botden, S. Buzink, M. Schijven, and J. Jakimowicz, *Augmented versus virtual reality laparoscopic simulation: What is the difference?*, World Journal of Surgery, 31 (2007), pp. 764–772.

[53] D. Brageul, S. Vukanovic, and B. MacDonald, *An intuitive interface for programming by demonstration*, in Proceedings of the IEEE International Conference on Robotics and Automation, Pasadena, California, USA., 2008, pp. 3570–3575.

[54] V. Brujic-Okretic, J.-Y. Guillemaut, L. Hitchin, M. Michielen, and G. Parker, *Remote vehicle manoeuvring using augmented reality*, in International Conference on Visual Information Engineering, University of Surrey, Guildford, UK, 2003, pp. 186–189.

[55] M. Bryson, M. Johnson-Roberson, and S. Sukkarieh, *Airborne smoothing and mapping using vision and inertial sensors*, in Proceedings of the IEEE International Conference on Robotics and Automation, Kobe, Japan, May 12–17 2009, pp. 2037–2042.

[56] D. Bulanon, T. Kataoka, H. Okamoto, and S. Hata, *Development of a real-time machine vision system for the apple harvesting robot*, in SICE Annual Conference, vol. 1, 4-6 Aug. 2004, pp. 595–598.

[57] G. Burdea, *Invited review: the synergy between virtual reality and robotics*, IEEE Transactions on Robotics and Automation, 15 (1999), pp. 400–410.

[58] S. Burigat and L. Chittaro, *Navigation in 3D virtual environments: Effects of user experience and location-pointing navigation aids*, International Journal of Human-Computer Studies, 65 (2007), pp. 945–958.

[59] G. Cai, B. Chen, K. Peng, M. Dong, and T. Lee, *Modeling and control system design for a UAV helicopter*, in Proceedings of the 14th Mediterranean Conference on Control and Automation, June 2006, pp. 1–6.

[60] M. Calonder, V. Lepetit, K. Konolige, J. Bowman, P. Mihelich, and P. Fua, *Compact signatures for high-speed interest point description and matching*, in IEEE International Conference on Computer Vision, Kyoto, Japan, 2009.

[61] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, *USARSim: a robot simulator for research and education*, in Proceedings of the IEEE International Conference on Robotics and Automation, Roma, April 10-14 2007, pp. 1400–1405.

[62] S. Carpin, Y. N. T. Stoyanov, M. Lewis, and J. Wang, *Quantitative assessments of USARSim accuracy*, in Proceedings of the Workshop on Performance Metrics for Intelligent Systems, 2006.

[63] R. O. Castle, G. Klein, and D. W. Murray, *Video-rate localization in multiple maps for wearable augmented reality*, in Proceedings of the 12th IEEE International Symposium on Wearable Computers, Pittsburgh PA, September 28–October 1 2008, pp. 15–22.

[64] M. Cavazza, F. Charles, S. J. M. O. Martin, X. Marichal, and A. Nandi, *Multimodal acting in mixed reality interactive storytelling*, IEEE Multimedia, 11 (2004), pp. 30–39.

[65] B. Cervin, C. Mills, and B. Wuensche, *A 3d interface for an unmanned aerial vehicle*, in Proceedings of the International Conference on Image and Vision Computing New Zealand, 2004.

[66] E. Chalmers, *Monocular and binocular cues in the perception of size and distance*, The American Journal of Psychology, 65 (1952), pp. 415–423.

[67] A. Chapman and S. Sukkarieh, *A protocol for decentralized multi-vehicle mapping with limited communication connectivity*, in Proceedings of the IEEE International Conference on Robotics and Automation, Kobe, Japan, May 12–17 2009, pp. 357–362.

[68] D. Chekhlov, M. Pupilli, W. Mayol, and A. Calway, *Robust real-time visual SLAM using scale prediction and exemplar based feature description*, in IEEE International Conference on Computer Vision and Pattern Recognition, June 2007.

[69] C. Chen, *Information Visualiszation: Beyond the Horizon*, Springer-Verlag London Limited, 2 ed., 2004.

[70] H. Chen, W. Sheng, N. Xi, M. Song, and Y. Chen, *Automated robot trajectory planning for spray painting of free-form surfaces in automotive manufacturing*, in Proceedings of the IEEE International Conference on Robotics and Automation, W. Sheng, ed., vol. 1, 2002, pp. 450–455.

[71] H. Chen, O. Wulf, and B. Wagner, *Object detection for a mobile robot using mixed reality*, in Interactive Technologies and Sociotechnical Systems, 2006, pp. 466–475.

[72] I. Chen, B. MacDonald, B. Wünsche, G. Biggs, and T. Kotoku, *A simulation environment for OpenRTM-aist*, in Proceedings of the IEEE International Symposium on System Integration, Tokyo, Japan, November 29 2009, pp. 113–117.

[73] I. Y.-H. Chen, B. MacDonald, and B. Wünsche, *Markerless augmented reality for robots in unprepared environments*, in Proceedings of the Australasian Conference on Robotics and Automation, Canberra, Australia, December 3–5 2008.

[74] I. Y.-H. Chen, B. MacDonald, and B. Wünsche, *Mixed reality simulation for mobile robots*, in Proceedings of the IEEE International Conference on Robotics and Automation, Kobe, Japan, May 12–17 2009, pp. 232–237.

[75] I. Y.-H. Chen, B. MacDonald, B. Wünsche, G. Biggs, and T. Kotoku, *Analysing mixed reality simulation for industrial applications: A case study in the development of a robotic screw remover system*, in Proceedings of the Second International Conference on Simulation, Modeling and Programming for Autonomous Robots, Darmstadt, Germany, November 15–18 2010, pp. 350–361.

[76] I. Y.-H. Chen, B. A. MacDonald, and B. C. Wünsche, *Designing a mixed reality framework for enriching interactions in robot simulation*, in Proceedings of the International Conference on Computer Graphics Theory and Applications, Angers, France, May 17–21 2010, pp. 331–338.

[77] R. Chen, *Vision-based cow tracking for a uav. phd proposal*, tech. report, Department of Electrical and Computer Engineering, University of Auckland., January 2007.

[78] C. Chlestil, E. Leitgeb, N. Schmitt, S. Muhammad, K. Zettl, and W. Rehm, *Reliable optical wireless links within UAV swarms*, in Proceedings of the International Conference on Transparent Optical Networks, vol. 4, 18-22 June 2006, pp. 39–42.

[79] T. Collett and B. MacDonald, *An augmented reality debugging system for mobile robot software engineers*, Journal of Software Engineering for Robotics, 1 (2009), pp. 18–32.

[80] T. H. J. Collett, *Augmented Reality Visualisation for Robot Developers*, PhD thesis, Department of Electrical and Comuter Engineering, The University of Auckland, 2006. Available Online: http://hdl.handle.net/2292/1510.

[81] A. Comport, E. Marchand, M. Pressigout, and F. Chaumette, *Real-time markerless tracking for augmented reality: the virtual visual servoing framework*, Transactions on Visualization and Computer Graphics, 12 (2006), pp. 615–628.

[82] K. L. B. Cook, *The silent force multiplier: The history and role of UAVs in warfare*, in Proceedings of the IEEE Aerospace Conference,, 2007, pp. 1–7.

[83] F. Cosmi, C. Fabro, P. Susmel, and G. Zoppello, *Automation in dairy farms: a robotic milking system*, in Proceedings of the 8th International Conference on Advanced Robotics, 7-9 July 1997, pp. 33–37.

[84] Cyberbotics, *Webots*, January 2009. http://www.cyberbotics.com/.

[85] M. Daily, Y. Cho, K. Martin, and D. Payton, *World embedded interfaces for human-robot interaction*, in Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 2003., Y. Cho, ed., 2003, p. 6.

[86] B. Davis, P. Patron, and D. Lane, *An augmented reality architecture for the creation of hardware-in-the-loop & hybrid simulation test scenarios for unmanned underwater vehicles*, in OCEANS, 2007, pp. 1–6.

[87] A. Davison, *Real-time simultaneous localisation and mapping with a single camera*, in Proceedings of the 9th IEEE International Conference on Computer Vision, 2003, pp. 1403–1410.

[88] 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, 26 (2007), pp. 1052–1067.

[89] J. de Carufel, E. Martin, and J.-C. Piedboeuf, *Control strategies for hardware-in-the-loop simulation of flexible space robots*, IEE Proceedings Control Theory and Applications, 147 (2000), pp. 569 –579.

[90] K. de Koning and J. Rodenburg, *Automatic milking: State of the art in europe and north america*, in Automatic milking: a better understanding. Conference Proceedings, Lelystad, Netherlands, March 2004.

[91] A. G. de Sa and G. Zachmann., *Virtual reality as a tool for verification of assembly and maintenance processes*, Computers & Graphics, 23 (1999), pp. 389–403.

[92] R. Diankov and J. Kuffner, *OpenRAVE: A planning architecture for autonomous robotics*, Tech. Report CMU-RI-TR-08-34, Robotics Institute, Carnegie Mellon University, Pittsburgh, 2008.

[93] A. Dietrich, M. Schulze, S. Zug, and J. Kaiser, *Visualization of robot's awareness and perception*, in Proceedings of the First International Workshop on Digital Engineering, ACM, 2010, pp. 38–44.

[94] A. Dix, J. Finlay, G. Adowd, and R. Beale., *Human Computer Interaction*, Prentice Hall, 2004.

[95] F. Dovis and F. Sellone, *Increasing the capacity of existing terrestrial outdoor radio mobile systems by means of UAV-HALE platforms*, in Proceedings of the 51st IEEE Vehicular Technology Conference, vol. 3, Tokyo, Japan, 2000, pp. 2365–2369.

[96] M. Dragone, T. Holz, and G. O'Hare, *Using mixed reality agents as social interfaces for robots*, in Proceedings of the 16th IEEE International Symposium on Robot and Human interactive Communication, T. Holz, ed., 2007, pp. 1161–1166.

[97] J. Drury, J. Richer, N. Rackliffe, and M. Goodrich, *Comparing situation awareness for two unmanned aerial vehicle human interface approaches*, in Proceedings IEEE International Workshop on Safety, Security and Rescue Robotics, MD, USA, August 2006.

[98] J. Drury, J. Scholtz, and H. Yanco, *Awareness in human-robot interactions*, in Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol. 1, October 2003, pp. 912–918.

[99] J. L. Drury, L. Riek, and N. Rackliffe, *A decomposition of uav-related situation awareness*, in Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction, New York, NY, USA, 2006, ACM, pp. 88–94.

[100] J. Dufrene, W.R., *AI techniques in uninhabited aerial vehicle flight*, IEEE Aerospace and Electronic Systems Magazine, 19 (2004), pp. 8–12.

[101] A. Dünser, R. Grasset, and M. Billinghurst, *A survey of evaluation techniques used in augmented reality studies*, in ACM SIGGRAPH ASIA 2008 courses, SIGGRAPH Asia '08, New York, NY, USA, 2008, ACM, pp. 5:1–5:27.

[102] R. Enns and J. Si, *Helicopter trimming and tracking control using direct neural dynamic programming*, IEEE Transactions on Neural Networks, 14 (2003), pp. 929–939.

[103] Epic Games, *Unreal Technology*, 2009. http://www.unrealtechnology.com/.

[104] B. Espiau, F. Chaumette, and P. Rives, *A new approach to visual servoing in robotics*, vol. 8, Springer, 1992, pp. 313–326.

[105] F. Ferland, F. Pomerleau, C. T. Le Dinh, and F. Michaud, *Egocentric and exocentric teleoperation interface using real-time 3d video projection*, in Proceedings of the 4th ACM/IEEE international conference on human robot interaction, New York, USA, 2009, pp. 37–44.

[106] M. Fiala, *ARTag revision 1, a fiducial marker system using digital techniques*, National Research Council Publication, 47419 (2004).

[107] M. Fischler and R. Bolles, *Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography*, Communications of the ACM, 24 (1981), pp. 381–395.

[108] FlightGear, *Flightgear flight simulator.*, January 2008. http://www.flightgear.org/.

[109] T. Fong, C. Thorpe, and C. Baur, *Advanced interfaces for vehicle teleoperation: Collaborative control, sensor fusion displays, and remote driving tools*, vol. 11, Springer, 2001, pp. 77–85.

[110] E. Freund and D. Pensky, *COSIMIR factory: extending the use of manufacturing simulations*, in Proceedings of the IEEE International Conference on Robotics and Automation, vol. 3, 2002, pp. 2805–2810.

[111] E. Freund and J. Rossmann, *Projective virtual reality: bridging the gap between virtual reality and robotics*, IEEE Transactions on Robotics and Automation, 15 (1999), pp. 411–422.

[112] R. Garcia and L. Barnes, *Multi-UAV simulator utilizing X-Plane*, Journal of Intelligent and Robotic Systems, 57 (2010), pp. 393–406.

[113] V. Gavrilets, I. Martinos, B. Mettler, and E. Feron, *Flight test and simulation results for an autonomous aerobatic helicopter*, in Proceedings of the 21st Digital Avionics Systems Conference, I. Martinos, ed., vol. 2, 2002, pp. 8C3–1–8C3–6.

[114] B. P. Gerkey, R. T. Vaughan, and A. Howard, *The player/stage project: Tools for multi-robot and distributed sensor systems*, in Proceedings of the International Conference on Advanced Robotics (ICAR 2003), June 30–July 3 2003, pp. 317–323.

[115] J. J. Gibson, *The Ecological Approach to Visual Perception*, MA: Houghton Mifflin, Boston, 1979.

[116] A. Göktoğan and S. Sukkarieh, *An Augmented Reality System for Multi-UAV Missions*, in Proceedings of the Simulation Conference and Exhibition, SimTect, Sydney, Australia, May 9–12 2005.

[117] K. Gotou, T. Fujiura, Y. Nishiura, H. Ikeda, and M. Dohi, *3-D vision system of tomato production robot*, in Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics, vol. 2, 20-24 July 2003, pp. 1210–1215.

[118] S. Green, M. Billinghurst, X. Chen, and G. Chase, *Human-robot collaboration: A literature review and augmented reality approach in design*, International Journal of Advanced Robotic Systems, 5 (2008), pp. 1–18.

[119] S. Green, J. Chase, X. Chen, and M. Billinghurst, *Evaluating the augmented reality human-robot collaboration system*, International Journal of Intelligent Systems Technologies and Applications, 8 (2010), pp. 130–143.

[120] I. Grierson and J. Gammon, *The use of aerial imaging in kangaroo management*, in Proceedings of the IEEE International Geoscience and Remote Sensing Symposium, J. Gammon, ed., vol. 4, 2000, pp. 1446–1448.

[121] D. Gu, G. Pei, H. Ly, M. Gerla, B. Zhang, and X. Hong, *UAV aided intelligent routing for ad-hoc wireless network in single-area theater*, in Proceedings of the IEEE Wireless Communications and Networking Conference, vol. 3, 2000, pp. 1220–1225.

[122] H-SIM, *Simdrone UAV simulator*. http://www.h-sim.com/.

[123] C. D. Hansen and C. R. Johnson, eds., *The Visualisation Handbook*, Elsevier Butterworth-Heinemann, 2005.

[124] F. Harmon, A. Frank, and S. Joshi, *Application of a CMAC neural network to the control of a parallel hybrid-electric propulsion system for a small unmanned aerial vehicle*, in Proceedings of the IEEE International Joint Conference on Neural Networks, vol. 1, 31 July-4 Aug. 2005, pp. 355–360.

[125] C. Harris and M. Stephens, *A combined corner and edge detector*, in Fourth Alvey Vision Conference, 1988, pp. 147–151.

[126] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2 ed., 2004.

[127] Y. Hashimoto, H. Murase, T. Morimoto, and T. Torii, *Intelligent systems for agriculture in Japan*, IEEE Control Systems Magazine, 21 (2001), pp. 71–85.

[128] S. R. Herwitz, L. F. Johnson, S. E. Dunagan, R. G. Higgins, D. V. Sullivan, J. Zheng, B. M. Lobitz, J. G. Leung, B. A. Gallmeyer, M. Aoyagi, R. E. Slye, and J. A. Brass, *Imaging from an unmanned aerial vehicle: agricultural surveillance and decision support*, Computers and Electronics in Agriculture, 44 (2004), pp. 49–61.

[129] B. Hine, P. Hontalas, T. Fong, L. Piguet, E. Nygren, and A. Kline, *VEVI: A virtual environment teleoperations interface for planetary exploration*, SAE transactions, 104 (1996), pp. 615–628.

[130] W. Hoff, K. Nguyen, and T. Lyon, *Computer vision-based registration techniques for augmented reality*, Proceedings of Intelligent Robots and Computer Vision XV, SPIE, 2904 (1996), pp. 538–548.

[131] T. Hollerer, S. Feiner, T. Terauchi, G. Rashid, and D. Hallaway, *Exploring MARS: developing indoor and outdoor user interfaces to a mobile augmented reality system*, Computers and Graphics, 23 (1999), pp. 779–785.

[132] T. Holz, M. Dragone, and G. O'Hare, *Where robots and virtual agents meet*, International Journal of Social Robotics, 1 (2008), pp. 83–93.

[133] C. E. Hughes, C. B. Stapleton, D. E. Hughes, and E. M. Smith, *Mixed reality in education, entertainment, and training*, IEEE Computer Graphics and Applications, 25 (2005), pp. 24–30.

[134] N. Ibrahim and N. Noor, *Navigation technique in 3D information visualisation*, in IEEE Region 10 Conference TENCON., vol. 2, 2004, pp. 379–382.

[135] R. Isermann, J. Schaffnit, and S. Sinsel, *Hardware-in-the-loop simulation for the design and testing of engine-control systems*, Control Engineering Practice, 7 (1999), pp. 643 – 653.

[136] N. Ito, *Agricultural robots in Japan*, in Proceedings of the IEEE International Workshop on Intelligent Robots and Systems. Towards a New Frontier of Applications, vol. 1, 1990, pp. 249–253.

[137] N. Jakobi, *Evolutionary robotics and the radical envelope-of-noise hypothesis*, Adaptive behavior, 6 (1997), p. 325.

[138] N. Jakobi, P. Husbands, and I. Harvey, *Noise and the reality gap: The use of simulation in evolutionary robotics*, in Advances in Artificial Life, F. Morán, A. Moreno, J. Merelo, and P. Chacón, eds., vol. 929 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 1995, pp. 704–720.

[139] E. Johnson and S. Mishra, *Flight Simulation for the Development of an Experimental UAV*, in Proceedings of the AIAA modeling and simulation technologies conference, 2002.

[140] G. P. JonesIV, L. G. Pearlstine, and H. F. Percival, *An assessment of small unmanned aerial vehicles for wildlife research*, in Wildlife Society Bulletin, vol. 34, October 2006, pp. 750–758.

[141] Y. Kameda, T. Takemasa, and Y. Ohta, *Outdoor see-through vision utilizing surveillance cameras*, in Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality, 2004, pp. 151–160.

[142] S. Karim, C. Heinze, and S. Dunn, *Agent-based mission management for a UAV*, in Proceedings of the Conference on Intelligent Sensors, Sensor Networks and Information Processing, 2004, pp. 481–486.

[143] H. Kato and M. Billinghurst, *Marker tracking and HMD calibration for a video-based augmented reality conferencing system*, in Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality, 1999, pp. 85–94.

[144] Y. Ke and R. Sukthankar, *PCA-SIFT: a more distinctive representation for local image descriptors*, in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, 2004, pp. II–506–II–513.

[145] 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*, International Journal of Robotics Research, 30 (2011), pp. 384–404.

[146] C.-S. Kim, C.-S. Kim, K.-S. Hong, H. Y.-S. Han, S.-H. Kim, and S.-C. Kwon, *PC-based off-line programming using VRML for welding robots in shipbuilding*, in Proceedings of the IEEE Conference on Robotics, Automation and Mechatronics, K.-S. Hong, ed., vol. 2, 2004, pp. 949–954 vol.2.

[147] J. P. C. Kleijnen, *Verification and validation of simulation models*, European Journal of Operational Research, 82 (1995), pp. 145–162.

[148] G. Klein and D. Murray, *Parallel tracking and mapping for small AR workspaces*, in Proceedings of the Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, November 2007, pp. 225–234.

[149] G. Klein and D. Murray, *Improving the agility of keyframe-based SLAM*, in European Conference on Computer Vision, Springer, 2008.

[150] ——, *Parallel tracking and mapping on a camera phone*, in Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality, Orlando, FL, USA, 2009, pp. 83–86.

[151] J. Klippenstein and H. Zhang, *Quantitative evaluation of feature extractors for visual SLAM*, in Fourth Canadian Conference on Computer and Robot Vision, 2007, pp. 157–164.

[152] G. H. Klungel, B. A. Slaghuis, and H. Hogeveen, *The effect of the introduction of automatic milking systems on milk quality*, J. Dairy Sci., 83 (2000), pp. 1998–2003.

[153] K. Kobayashi, K. Nishiwaki, S. Uchiyama, H. Yamamoto, and S. Kagami, *Viewing and reviewing how humanoids sensed, planned and behaved with mixed reality technology*, in Proceedings of the IEEE-RAS 7th International Conference on Humanoid Robots, 2006.

[154] K. Kobayashi, K. Nishiwaki, S. Uchiyama, H. Yamamoto, S. Kagami, and T. Kanade, *Overlay what humanoid robot perceives and thinks to the real-world by mixed reality system*, in Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, 2007, pp. 275–276.

[155] N. Koenig and A. Howard, *Design and use paradigms for Gazebo, an open-source multi-robot simulator*, in Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, September 28–October 2 2004, pp. 2149–2154.

[156] N. Kondo, *Fruit grading robot*, in Proceedings. IEEE/ASME International Conference on Advanced Intelligent Mechatronics, vol. 2, 20-24 July 2003, pp. 1366–1371.

[157] A. Kozlov, B. MacDonald, and B. Wünsche, *Covariance Visualisations for Simultaneous Localisation and Mapping*, in Proceedings of the Australian Conference on Robotics and Automation, Sydney, Australia, December 2–4 2009.

[158] Y. Kuroda, K. Aramaki, and T. Ura, *AUV test using real/virtual synthetic world*, in IEEE Symposium on Autonomous Underwater Vehicle Technology, Monterey, USA, 1996, pp. 365–372.

[159] A. Lecuyer, C.Megard, J.-M. Burkhardt, T. Lim, S. Coquillart, P. Coiffet, , and L. Graux., *The effect of haptic, visual and auditory feedback on an insertion task on a 2-screen workbench.*, in Proceedings of the Immersive Projection Technology Symposium, 2002.

[160] J. Ledin, *Hardware-in-the-loop simulation*, Embedded System Program, 12 (1999), pp. 42–60.

[161] M. Lefebvre, S. Gil, M. Glassey, C. Baur, and T. Pun, *3D computer vision for agrotics: the potato operation, an overview*, in Proceedings of the 11th IAPR International Conference on Pattern Recognition. Conference A: Computer Vision and Applications., vol. 1, 30 August–3 September 1992, pp. 207–210.

[162] M. Liu, G. Egan, and Y. Ge, *Identification of attitude flight dynamics for an unconventional UAV*, in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, pp. 3243–3248.

[163] M. A. Livingston, L. J. Rosenblum, S. J. Julier, D. Brown, Y. Baillot, J. E. S. II, J. L. Gabbard, and D. Hix, *An augmented reality system for military operations in urban terrain*, in Proceedings of the Interservice / Industry Training, Simulation, & Education Conference, Orlando, December 2-5 2002, pp. 1–8.

[164] J. Looser, R. Grasset, H. Seichter, and M. Billinghurst, *OSGART-A pragmatic approach to MR*, in Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality, 2006, pp. 22–25.

[165] M. Lourakis and A. Argyros, *Vision-based camera motion recovery for augmented reality*, in Proceedings of the Computer Graphics International, Washington, DC, USA, 2004, IEEE Computer Society, pp. 569–576.

[166] D. Lowe, *Object recognition from local scale-invariant features*, in Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 2, 1999, pp. 1150–1157.

[167] M. Lundell, J. Tang, and K. Nygard, *Fuzzy petri net for UAV decision making*, in Proceedings of the International Symposium on Collaborative Technologies and Systems, 15-20 May 2005, pp. 347–352.

[168] B. MacDonald, D. Yuen, S. Wong, E. Woo, R. Gronlund, T. Collett, F. Trépanier, and G. Biggs, *Robot programming environments*, in Proceedings of the 10th Electronics New Zealand Conference, Hamilton, 2003, Citeseer, pp. 1–2.

[169] J. Mackinlay, *Opportunities for information visualization*, IEEE Computer Graphics and Applications, 20 (2000), pp. 22–23.

[170] J. Maida, C. Bowen, and J. Pace, *Improving robotic operator performance using augmented reality*, in Human Factors and Ergonomics Society Annual Meeting Proceedings, vol. 51, Human Factors and Ergonomics Society, 2007, pp. 1635–1639.

[171] E. Marchand and F. Chaumette, *Virtual visual servoing: a framework for real-time augmented reality*, in Proceedings of EUROGRAPHICS'02, vol. 21, Saarebrcken, Germany, 2002, pp. 289– 298.

[172] A. Martin and M. Emami, *An architecture for robotic hardware-in-the-loop simulation*, in Proceedings of the IEEE International Conference on Mechatronics and Automation, IEEE, 2006, pp. 2162–2167.

[173] I. A. McManus, D. G. Greer, and R. A. Walker, *UAV avionics "hardware in the loop" simulator*, in Proceedings of the 10th Australian International Aerospace Congress, Brisbane, Queensland, Australia., 2003.

[174] A. Meyer, *X-Plane*, January 2008. http://www.x-plane.com/.

[175] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, *The GRASP multiple micro-UAV testbed*, IEEE Robotics & Automation Magazine, 17 (2010), pp. 56–65.

[176] Microsoft, *Microsoft flight simulator.*, January 2008. http://www.microsoft.com/games/pc/flightsimulator.aspx.

[177] ——, *Microsoft robotics studio*, 2009. http://msdn2.microsoft.com/en-us/robotics/default.aspx.

[178] K. Mikolajczyk and C. Schmid, *A performance evaluation of local descriptors*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 27 (2005), pp. 1615–1630.

[179] P. Milgram and H. Colquhoun, *A taxonomy of real and virtual world display integration*, Mixed Reality-Merging Real and Virtual Worlds, (1999), pp. 5–28.

[180] P. Milgram and F. Kishino, *A taxonomy of mixed reality visual display*, in IEICE Transactions on Information Systems, vol. E77-D, December 1994, pp. 1321–1329.

[181] P. Milgram, A. Rastogi, and J. Grodski, *Telerobotic control using augmented reality*, in Proceedings of the 4th IEEE International Workshop on Robot and Human Communication, Tokyo, 1995, pp. 21–29.

[182] A. Monferrer and D. Bonyuet, *Cooperative robot teleoperation through virtual reality interfaces*, in Proceedings of the Sixth International Conference on Information Visualisation, 2002, pp. 243–248.

[183] M. Monta, N. Kondo, and Y. Shibano, *Agricultural robot in grape production system*, in Proceedings of the IEEE International Conference on Robotics and Automation, vol. 3, 21-27 May 1995, pp. 2504–2509.

[184] P. Moreels and P. Perona, *Evaluation of features detectors and descriptors based on 3D objects*, in Proceedings of the Tenth IEEE International Conference on Computer Vision, vol. 1, 2005, pp. 800–807.

[185] National Institute of Advanced Industrial Science and Technology (AIST), *RT-Middleware: OpenRTM-aist*, 2009. http://www.openrtm.org/.

[186] A. Nawrat and K. Kozak, *Development of a system for simulation and control of an unmanned aerial vehicle*, Advanced Robotics, 23 (2009), pp. 257–268.

[187] U. Neumann and S. You, *Natural feature tracking for augmented reality*, IEEE Transactions on Multimedia, 1 (1999), pp. 53–64.

[188] L. Nguyen, M. Bualat, L. Edwards, L. Flueckiger, C. Neveu, K. Schwehr, M. Wagner, and E. Zbinden, *Virtual reality interfaces for visualization and control of remote vehicles*, Autonomous Robots, 11 (2001), pp. 59–68.

[189] T. H. D. Nguyen, T. C. T. Qui, K. Xu, A. D. Cheok, S. L. Teo, Z. Zhou, A. Mallawaarachchi, S. P. Lee, W. Liu, H. S. Teo, L. N. Thang, Y. Li, and H. Kato, *Real-time 3d human capture system for mixed-reality art and entertainment*, IEEE Transactions on Visualization and Computer Graphics, 11 (2005), pp. 706–721.

[190] C. Nielsen, M. Goodrich, and R. Ricks, *Ecological interfaces for improving mobile robot teleoperation*, IEEE Transactions on Robotics, 23 (2007), pp. 927–941.

[191] W. Niland, *The migration of a collaborative UAV testbed into the flames simulation environment*, in Proceedings of the Winter Simulation Conference, 2006, pp. 1266–1272.

[192] K. Nishiwaki, K. Kobayashi, S. Uchiyama, H. Yamamoto, and S. Kagami, *Mixed reality environment for autonomous robot development*, in Proceedings of the IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, May 19–23 2008, pp. 2211–2212.

[193] G. O'Hare, B. Duffy, J. Bradley, and A. Martin, *Agent chameleons: Moving minds from robots to digital information spaces*, 2003, pp. 18–21.

[194] M. Ollis and A. Stentz, *Vision-based perception for an automated harvester*, in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, 7-11 Sept. 1997, pp. 1838–1844.

[195] S. K. Ong, J. W. S. Chong, and A. Y. C. Nee, *Methodologies for immersive robot programming in an augmented reality environment*, in Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia, New York, NY, USA, 2006, ACM, pp. 237–244.

[196] S. K. Ong, M. L. Yuan, and A. Y. C. Nee, *Tracking points using projective reconstruction for augmented reality*, in Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia, New York, NY, USA, 2005, ACM Press, pp. 421–424.

[197] J. M. Ortiz and M. Olivares, *A vision based navigation system for an agricultural field robot*, in Proceedings of the IEEE 3rd Latin American Robotics Symposium, Oct. 2006, pp. 106–114.

[198] J. Owens, *Workspace - a microcomputer-based industrial robot simulator and off-line programming system*, in IEE Colloquium on Next Steps for Industrial Robotics, 1994, pp. 4/1–4/4.

[199] M. Özuysal, M. Calonder, V. Lepetit, and P. Fua, *Fast keypoint recognition using random ferns*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 32 (2009), pp. 448–461.

[200] M. Özuysal, P. Fua, and V. Lepetit, *Fast keypoint recognition in ten lines of code*, in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Minneapolis, Minnesota, USA, 18–23 June 2007, pp. 1–8.

[201] Z. Papp, M. Dorrepaal, A. Thean, and M. van Elk, *High fidelity real-time simulator with mixed real and virtual sensors*, in Proceedings of the IEEE International Conference on Robotics and Automation, 2005, pp. 4526–4531.

[202] C.-H. Park, K.-T. Park, and D.-G. Gweon, *Development of industrial dual arm robot for precision assembly of mechanical parts for automobiles*, in International Joint Conference SICE-ICASE, K.-T. Park, ed., 2006, pp. 3059–3062.

[203] K. Pentenrieder, C. Bade, F. Doil, and P. Meier, *Augmented reality-based factory planning - an application tailored to industrial needs*, in Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, 2007, pp. 31–42.

[204] D. Perera and K. Blashki, *A comparative study of computational value in visual cues used in positioning 3D objects*, in Proceedings of the 12th International Multi-Media Modelling Conference, K. Blashki, ed., 2006, pp. 464–468.

[205] T. Pettersen, J. Pretlove, C. Skourup, T. Engedal, and T. Lokstad, *Augmented reality for programming industrial robots*, in Proceedings of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003, pp. 319–320.

[206] W. Piekarski and B. Thomas, *Tinmith-metro: New outdoor techniques for creating city models with an augmented reality wearable computer*, in Proceedings of the Fifth International Symposium on Wearable Computers, IEEE, 2001, pp. 31–38.

[207] W. Piekarski and B. H. Thomas, *An object-oriented software architecture for 3D mixed reality applications*, in Proceedings of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality, October 7–10 2003, pp. 247–256.

[208] M. Pinkney, D. Hampel, and S. DiPierro, *Unmanned aerial vehicle (UAV) communications relay*, in Proceedings of the IEEE Military Communications Conference, D. Hampel, ed., vol. 1, 1996, pp. 47–51.

[209] J. Platonov, H. Heibel, P. Meier, and B. Grollmann, *A mobile markerless ar system for maintenance and repair*, in IEEE/ACM International Symposium on Mixed and Augmented Reality, 2006, pp. 105–108.

[210] E. Price and G. Egan, *Real-time UAV visualization using a flight simulator*, in AIAC12, 16-22 March 2007.

[211] I. Price and G. Lamont, *GA directed self-organized search and attack UAV swarms*, in Proceedings of the Winter Simulation Conference, 3-6 Dec. 2006, pp. 1307–1315.

[212] M. Pupilli and A. Calway, *Real-time camera tracking using a particle filter*, in Proceedings of the British Machine Vision Conference, 2005, pp. 519–528.

[213] ——, *Real-time visual SLAM with resilience to erratic motion*, in IEEE Computer Vision and Pattern Recognition, 2006.

[214] S. Rasmussen, J. Mitchell, P. Chandler, C. Schumacher, and A. Smith, *Introduction to the MultiUAV2 Simulation and Its Application to Cooperative Control Research (I)*, in Proceedings of the American Control Conference, vol. 7, 2005, pp. 4490–4501.

[215] S. Rasmussen, J. Mitchell, C. Schulz, C. Schumacher, and P. Chandler, *A multiple UAV simulation for researchers*, in Proceedings of the AIAA Modeling and Simulation Technologies Conference, Austin, Texas, 2003.

[216] M. Recce, J. Taylor, A. Plebe, and G. Tropiano, *Vision and neural control for an orange harvesting robot*, in Proceedings of the International Workshop on Neural Networks for Identification, Control, Robotics, and Signal/Image Processing, 21-23 Aug. 1996, pp. 467–475.

[217] G. Reitmayr and T. Drummond, *Going out: robust model-based tracking for outdoor augmented reality*, in Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality, IEEE Computer Society, 2006, pp. 109–118.

[218] T.-M. Rhyne, *Does the difference between information and scientific visualization really matter?*, IEEE Computer Graphics and Applications, 23 (2003), pp. 6–8.

[219] M. Ribo, P. Lang, H. Ganster, M. Brandner, C. Stock, and A. Pinz, *Hybrid tracking for outdoor augmented reality applications*, IEEE Computer Graphics and Applications, 22 (2002), pp. 54–63.

[220] P. Ridao, E. Batlle, D. Ribas, and M. Neptune, *Neptune: a HIL simulator for multiple UUVs*, IEEE TECHNO-OCEAN, 4 (2004), pp. 524–531.

[221] E. Rinalducci and K. Uliano, *Visual cues and potential problems in flight simulation*, in Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, K. Uliano, ed., vol. 2, 1992, pp. 1153–1158.

[222] RobotWorks, *Robotworks.*, January 2008. http://robotworks-eu.com/products/RBWabout.htm.

[223] B. Rogers and M. Graham, *Motion parallax as an independent cue for depth perception*, Perception, 8 (1979), pp. 125–34.

[224] Y. Rogers, M. Scaife, S. Gabrielli, H. Smith, and E. Harris, *A Conceptual Framework for Mixed Reality Environments: Designing Novel Learning Activities for Young Children*, Presence: Teleoperators & Virtual Environments, 11 (2002), pp. 677–686.

[225] M. Rosenthal, A. State, J. Lee, G. Hirota, J. Ackerman, K. Keller, E. D. Pisano, M. Jiroutek, K. Muller, and H. Fuchs, *Augmented reality guidance for needle biopsies: A randomized, controlled trial in phantoms*, in MICCAI '01: Proceedings of the 4th International Conference on Medical Image Computing and Computer-Assisted Intervention, London, UK, 2001, Springer-Verlag, pp. 240–248.

[226] E. Rosten and T. Drummond, *Machine learning for high-speed corner detection*, in European Conference on Computer Vision, vol. 1, May 2006, pp. 430–443.

[227] A. Ryan, M. Zennaro, A. Howell, R. Sengupta, and J. Hedrick, *An overview of emerging results in cooperative UAV control*, in Proceedings of the 43rd IEEE Conference on Decision and Control, vol. 1, 14-17 Dec. 2004, pp. 602–607.

[228] S. Sakai, K. Osuka, T. Maekawa, and M. Umeda, *Active vision of a heavy material handling agricultural robot using robust control: a case study for initial cost problem*, in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2-6 Aug. 2005, pp. 572–578.

[229] R. Sargent, *Verification and validation of simulation models*, in Proceedings of the 40th conference on Winter simulation, Winter Simulation Conference, 2008, pp. 157–169.

[230] K. Satoh, M. Anabuki, H. Yamamoto, and H. Tamura, *A hybrid registration method for outdoor augmented reality*, in Proceedings of the IEEE and ACM International Symposium onAugmented Reality, 2001, pp. 67–76.

[231] B. Schachter, *Computer image generation for flight simulation*, IEEE Computer Graphics and Applications, 1 (1981), pp. 29–68.

[232] I. Schiller and J. Draper, *Mission adaptable autonomous vehicles*, in IEEE Conference on Neural Networks for Ocean Engineering, 15-17 August 1991, pp. 143–150.

[233] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavari, L. Encarnacao, M. Gervautz, and W. Purgathofer, *The Studierstube Augmented Reality Project*, Presence: Teleoperators & Virtual Environments, 11 (2002), pp. 33–54.

[234] J. Scholtz, J. Young, J. Drury, and H. Yanco, *Evaluation of human-robot interaction awareness in search and rescue*, in Proceedings of the IEEE International Conference on Robotics and Automation, vol. 3, IEEE, 2004, pp. 2327–2332.

[235] G. Schweighofer and A. Pinz, *Robust pose estimation from a planar target*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28 (2006), pp. 2024–2030.

[236] D. Seward and A. Garman, *The software development process for an intelligent robot*, Computing & Control Engineering Journal, 7 (1996), pp. 86–92.

[237] J. Shi and C. Tomasi, *Good features to track*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1994, pp. 593–600.

[238] M. Shoji, K. Miura, and A. Konno, *U-Tsu-Shi-O-Mi: the virtual humanoid you can reach*, in ACM SIGGRAPH emergence technologies, New York, NY, USA, 2006, ACM, p. 34.

[239] G. Simon and M.-O. Berger, *Pose estimation for planar structures*, IEEE Computer Graphics and Applications, 22 (2002), pp. 46–53.

[240] G. Simon, A. Fitzgibbon, and A. Zisserman, *Markerless tracking using planar structures in the scene*, in Proceedings of the IEEE and ACM International Symposium on Augmented Reality, 2000, pp. 120–128.

[241] F. Sistler, *Robotics and intelligent machines in agriculture*, IEEE Journal of Robotics and Automation, 3 (1987), pp. 3–6.

[242] F. Sistler, *Grading agricultural products with machine vision*, in Proceedings of the IEEE International Workshop on Intelligent Robots and Systems., vol. 1, 3–6 July 1990, pp. 255–261.

[243] I. Skrypnyk and D. Lowe, *Scene modelling, recognition and tracking with invariant image features*, in Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality, 2004, pp. 110–119.

[244] J. Smith and T. Nguyen, *Fuzzy logic based resource manager for a team of UAVs*, in Annual meeting of the North American Fuzzy Information Processing Society, 2006, pp. 463–470.

[245] R. Smith, *Open Dynamics Engine*, 2009. http://www.ode.org/.

[246] J. Sreng, A. Lecuyer, C. Megard, and C. Andriot, *Using visual cues of contact to improve interactive manipulation of virtual objects in industrial assembly/maintenance simulations*, Transactions on Visualization and Computer Graphics, 12 (2006), pp. 1013–1020.

[247] C. Stapleton and C. E. Hughes, *Believing is seeing: cultivating radical media innovations*, IEEE Computer Graphics and Applications, 26 (2006), pp. 88–93.

[248] M. Stilman, P. Michel, J. Chestnutt, K. Nishiwaki, S. Kagami, and J. Kuffner, *Augmented reality for robot development and experimentation*, Tech. Report CMU-RI-TR-05-55, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, November 2005.

[249] H. Strasdat, J. Montiel, and A. Davison, *Real-time monocular SLAM: Why filter?*, in Proceedings of the IEEE International Conference on Robotics and Automation, 2010, pp. 2657–2664.

[250] M. Sugeno, M. Sugeno, I. Hirano, S. Nakamura, and S. Kotsu, *Development of an intelligent unmanned helicopter*, in Proceedings of IEEE International Conference on Fuzzy Systems, I. Hirano, ed., vol. 5, 1995, pp. 33–34.

[251] M. Sugimoto, G. Kagotani, H. Nii, N. Shiroma, M. Inami, and F. Matsuno, *Time follower's vision: a teleoperation interface with past images*, IEEE Computer Graphics and Applications, 25 (2005), pp. 54–63.

[252] R. Sugiura, N. Noguchi, and K. Ishii, *Remote-sensing technology for vegetation monitoring using an unmanned helicopter*, in Biosystems Engineering, vol. 90, April 2005, pp. 369–379.

[253] J. Sullivan, *Revolution or evolution? the rise of the UAVs*, in Proceedings of the International Symposium on Technology and Society. Weapons and Wires: Prevention and Safety in a Time of Fear., 2005, pp. 94–101.

[254] V. Sundareswaran and R. Behringer, *Visual servoing-based augmented reality*, in Proceedings of the International Workshop on Augmented Reality. IWAR '98, Natick, MA, USA, 1999, A. K. Peters, Ltd., pp. 193–200.

[255] J. Swan and J. Gabbard, *Survey of user-based experimentation in augmented reality*, in Proceedings of 1st International Conference on Virtual Reality, Las Vegas, Nevada, 2005, pp. 1–9.

[256] R. Takahashi, H. Ise, A. Konno, M. Uchiyama, and D. Sato, *Hybrid simulation of a dual-arm space robot colliding with a floating object*, in Proceedings of the IEEE International Conference on Robotics and Automation, 2008, pp. 1201–1206.

[257] Y. Takahashi, J. Ogawa, and K. Saeki, *Automatic tomato picking robot system with human interface using image processing*, in The 27th Annual Conference of the IEEE Industrial Electronics Society, vol. 1, 29 Nov.-2 Dec. 2001, pp. 433–438.

[258] H. Tamura, H. Yamamoto, and A. Katayama, *Mixed reality: Future dreams seen at the border between real and virtual worlds*, IEEE Computer Graphics and Applications, 21 (2001), pp. 64–70.

[259] H. Tanner, A. Jadbabaie, and G. Pappas, *Stable flocking of mobile agents, part I: fixed topology*, in Proceedings of the 42nd IEEE Conference on Decision and Control, A. Jadbabaie, ed., vol. 2, 2003, pp. 2010–2015.

[260] ——, *Stable flocking of mobile agents part II: dynamic topology*, in Proceedings of the 42nd IEEE Conference on Decision and Control, A. Jadbabaie, ed., vol. 2, 2003, pp. 2016–2021.

[261] The RoboCup Federation, *RoboCup*, 2009. http://www.robocup.org/.

[262] B. Thomas, B. Close, J. Donoghue, J. Squires, P. D. Bondi, M. Morris, and W. Piekarski, *ARQuake: an outdoor/indoor augmented reality first person application*, in Proceedings of the Fourth International Symposium on Wearable Computers, October 16–17 2000, pp. 139–146.

[263] C. Tomasi and T. Kanade, *Detection and tracking of point features*, Tech. Report CMU-CS-91-132, Carnegie Mellon University, 1991.

[264] T. Trinh and J. Kuchar, *Study of visual cues for unmanned aerial vehicle waypoint allocation*, in Proceedings of the 18th Digital Avionics Systems Conference, J. Kuchar, ed., vol. 1, 1999, pp. 4.D.5–1–4.D.5–8.

[265] A. Tsourdos, *A formal model approach for the analysis and validation of the cooperative path planning of a UAV team*, in Proceedings of the IEE Seminar on Autonomous Agents in Control, 2005, pp. 67–73.

[266] S. Uchiyama, K. Takemoto, K. Satoh, H. Yamamoto, and H. Tamura, *MR Platform: a basic body on which mixed reality applications are built*, in Proceedings of the International Symposium on Mixed and Augmented Reality, Darmstadt, Germany, September 30–October 1 2002, pp. 246–.

[267] R. Vaughan, *Massively multi-robot simulation in stage*, Swarm Intelligence, 2 (2008), pp. 189–208.

[268] N. G. Vinson, *Design guidelines for landmarks to support navigation in virtual environments*, in Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit, Pittsburgh, Pennsylvania, United States, 1999, ACM, pp. 278–285.

[269] V. Vlahakis, J. Karigiannis, M. Tsotros, N. Ioannidis, and D. Stricker, *Personalized augmented reality touring of archaeological sites with wearable and mobile computers*, in Proceedings of the Sixth International Symposium on Wearable Computers, 2002, pp. 15–22.

[270] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, *Pose tracking from natural features on mobile phones*, in Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, Cambridge, UK, 2008, IEEE Computer Society, pp. 125–134.

[271] D. Wagner and D. Schmalstieg, *ARToolKitPlus for pose tracking on mobile devices*, in Computer Vision Winter Workshop, February 6-8 2007.

[272] J. Wang, M. Lewis, and J. Gennari, *Usar: A game based simulation for teleoperation*, in Proceedings of the 47th Annual Meeting of the Human Factors and Ergonomics Society, Oct. 13-17 2003.

[273] J. Wang, M. Lewis, S. Hughes, and M. Koes, *Validating usarsim for use in hri research*, in Proceedings of the 49th Human Factors and Ergonomics Society Annual Meeting, 2005, p. 457V461.

[274] C. Warrington, G. Roth, and E. Dubois, *Markerless augmented reality for cubic panorama sequences*, in IEEE/ACM International Symposium on Mixed and Augmented Reality, 2006, pp. 255–256.

[275] F. L. Wendt, *SIFT based augmented reality*, master's thesis, University of Applied Sciences in Stuttgart, February 2007.

[276] B. Williams, G. Klein, and I. Reid, *Real-time SLAM relocalisation*, in Proceedings of the International Conference on Computer Vision, 2007, pp. 1–8.

[277] K. C. Wong, *Survey of regional developments: Civil applications*, in UAV Australia Conference, February 8 2001, pp. 8–16.

[278] Z. Wu, H. Kumar, and A. Davari, *Performance evaluation of OFDM transmission in UAV wireless communication*, in Proceedings of the Thirty-Seventh Southeastern Symposium on System Theory, 20-22 March 2005, pp. 6–10.

[279] Y. Yamada, S. Nagamatsu, and Y. Sato, *Development of multi-arm robots for automobile assembly*, in Proceedings of the IEEE International Conference on Robotics and Automation, S. Nagamatsu, ed., vol. 3, 1995, pp. 2224–2229.

[280] A. Yao and A. Calway, *Robust estimation of 3-D camera motion for uncalibrated augmented reality*, Tech. Report CSTR-02-001, Department of Computer Science, University of Bristol, March 2002.

[281] S. You and U. Neumann, *Fusion of vision and gyro tracking for robust augmented reality registration*, in IEEE Virtual Reality, 2001, pp. 71–78.

[282] S. You, U. Neumann, and R. Azuma, *Orientation tracking for outdoor augmented reality registration*, IEEE Computer Graphics and Applications, 19 (1999), pp. 36–42.

[283] J. Young, E. Sharlin, and J. Boyd, *Implementing bubblegrams: The use of haar-like features for human-robot interaction*, in IEEE International Conference on Automation Science and Engineering, E. Sharlin, ed., 2006, pp. 298–303.

[284] M. Yuan, S. Ong, and A. Nee, *Registration using natural features for augmented reality systems*, IEEE Transactions on Visualization and Computer Graphics, 12 (2006), pp. 569–580.

[285] J. Zauner, M. Haller, and A. Brandl, *Authoring of a mixed reality assembly instructor for hierarchical structures*, in Proceedings of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality, October 7-10 2003, pp. 237–246.

[286] S. Zelinski, T. Koo, and S. Sastry, *Optimization-based formation reconfiguration planning for autonomous vehicles*, in Proceedings of the IEEE International Conference on Robotics and Automation, T. Koo, ed., vol. 3, 2003, pp. 3758–3763.

[287] P. Zhan, K. Yu, and A. Swindlehurst, *Wireless relay communications using ann unmanned aerial vehicle*, in Proceedings of the IEEE 7th Workshop on Signal Processing Advances in Wireless Communications, K. Yu, ed., 2006, pp. 1–5.

[288] J. Zhao, J. Tow, and J. Katupitiya, *On-tree fruit recognition using texture properties and color data*, in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2-6 Aug. 2005, pp. 263–268.