Journal of Software Engineering for Robotics 2(1), September 2011, 40-54
ISSN: 2035-3928

A Flexible Mixed Reality Simulation Framework for Software Development in Robotics

Ian Yen-Hung CHEN[1] Bruce A. MACDONALD[1] Burkhard C. WÜNSCHE[2]
[1] Department of Electrical and Computer Engineering, University of Auckland, New Zealand
[2] Department of Computer Science, University of Auckland, New Zealand

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 cycle. This paper identifies requirements for improving the robot development process, focusing on facilitating a reliable transition process from simulation to real world tests. Building on the Hardware-in-the-Loop (HIL) simulation paradigm, we propose to apply the concept of Mixed Reality (MR) to robot simulation for creating a testbed where real and virtual objects are capable of interacting with one another. This paper contributes a conceptual MR framework for providing a generic simulation environment for experimentation. Using our conceptual framework, we implement an MR robot simulator that integrates widely accepted robotic software frameworks with standardised data interfaces to increase interoperability. Case study evaluations demonstrate the reuse of the MR robot simulator in three different applications, illustrating how it could help to create realistic test environments, and address development cost and efficiency issues.

Index Terms—Mixed Reality, Robot Simulation, Software Development.

1 Introduction

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 tasks expected of robots have grown more complicated, and are situated in complex and unpredictable environments shared with humans. This has led to a more challenging robot development process. Experiments are conducted under different environments and validated using a multitude of scenarios and inputs in order to gather reliable results before deploying the robot in the actual operation. Moreover, high risk robot operations, such as underwater, aerial, and space applications, typically require substantial resources and human support to ensure safety. The considerable cost and time for creating well-designed tests and for meeting safety requirements often pose challenges to rapid development of robot systems.

While virtual simulations can help to reduce cost and time during robot software development, the real world contains noise and variations that are difficult to replicate in simulation. Complex environments or systems are expensive to model, thus simulations often rely on approximations and assumptions in order to make the problem mathematically and computationally tractable [1]. Robot software engineers are faced with uncertainties when bringing simulation results to the real world. In high risk applications, the consequence of failures can be significant, thus it is crucial to facilitate safe and reliable transfer of results to reality.

One approach to bridge the gap between simulation and real world tests is hybrid simulation. Hybrid simulation is based on the Hardware-in-the-Loop (HIL) simulation paradigm. It is a form of real time simulation that combines simulation models and physical hardware in experimentation to offer a cost-effective alternative to real world tests. The method has been commonly used in applications involving the development of complex dynamic systems, such as aerial robots, offering a solution to testing robot systems in an incremental fashion [2]. Implementing an HIL simulation is often expensive, thus in order to be viable the simulation's benefits must exceed its development cost [3]. Consequently HIL simulations are most commonly found in application fields where the consequence of test failures are severe, for example in testing underwater robot obstacle avoidance algorithms [4], developing control strategies for aerial robots [5], and simulating collisions in

Regular paper – *Manuscript received March 1, 2011; revised August 31, 2011.*

- *Ian Chen was funded by the University of Auckland Doctoral Scholarship.*

space applications [6].

The goal of our work focuses on providing a flexible and reusable approach to robot experimentation, with the intention of allowing hybrid simulations to be applied to a wider range of robot domains. To achieve this goal, we explore the application of Mixed Reality (MR) to the development of robot systems. MR merges the real world and the virtual world to form a single coherent environment. This paper proposes that the concept of MR can be applied to robot simulation, creating a new simulation method, named MR simulation, that acts as a stepping stone towards testing in the real world. MR simulations offer increased flexibility that will contribute to the robot software development process. Various complex test scenarios involving real and virtual entities, both robotic devices and environmental objects, can be created for evaluating algorithms such as planning and control strategies before system integration. This also enables concurrent development of robot components (both software and hardware), which helps to speed up the software development cycle and allows problems to be identified early.

MR simulation stems from the HIL simulation paradigm and shares many advantages with existing HIL and hybrid simulation approaches. In comparison, MR simulation stresses a higher degree of coherency between the existence of real and virtual objects by placing an emphasis on realising interactions between them, so that in simulations, any virtual entities are able to physically interact with real entities, and vise versa. More importantly, we have a strong interest in providing a reusable solution. While there are signs of providing more modular HIL simulation methods for robot development [7], [8], work is still necessary to allow them to be reused for developing robot systems across different applications. Generic HIL robot simulators are less common; this is in contrast to various general purpose virtual robot simulators that are available today, including Player project's Stage [9] and Gazebo [10], USARSim [11], and OpenRAVE [12].

Section 2 describes related work. Section 3 discusses requirements for improving the robot experimentation environment. Section 4 presents a conceptual MR framework as part of the overall solution. Section 5 demonstrates the application of the MR framework to robot simulation. Section 6 describes an implementation of the framework. Section 7 evaluates the implemented system and presents results. Section 8 discusses issues encountered during the evaluation process, and Section 9 proposes guidelines for creating MR simulations.

2 Related Work

There is a growing interest in applying MR to robotics, primarily in the form of visualisations. Specifically, MR encompasses Augmented Reality (AR) and Augmented Virtuality (AV), both of which have been used to help humans understand robots or assist them in accomplishing a task. AR overlays virtual objects onto a view of the real world. AR has been demonstrated to help convey robot information and improve human-robot interaction, e.g., [13], [14], aid teaching of tasks in programming by demonstration, e.g., [15], [16], [17], and assist robot teleoperation, e.g., [18], [19], [20]. In contrast to AR, AV augments a virtual environment with real world information. AV has more commonly been applied for visualisation in robot teleoperation tasks to increase the operator's situation awareness, e.g., [21].

Our work focuses on the application of MR to robot software development. Previous work in this field has concentrated on using AR visualisations to help robot developers debug and program robot systems. Collett and MacDonald [22] propose an intelligent debugging space that uses AR visualisation for overlaying robot data in context with the real world. The 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. A similar technique is used by Kozlov *et al.* [23] to help robot programmers debug SLAM algorithms. Ong *et al.* [24] create an AR environment for immersive programming of robots. A developer is able to view the results of its algorithm replicated on a virtual robot in the real world.

It can be seen that AR offers unique benefits to robot software development. Most research limits the application of AR/MR to visualisation tasks and does not consider physical interactions between real and virtual objects, which can be useful in robot applications. An example is the RoboCup MR League [25], where teams of thumb-size robots engage in soccer matches on a virtual simulated soccer field. However, the concept of real-virtual interaction has rarely been explored in robotics, despite its potential to aid simulations tasks.

There exists little research in robotics that exploits the use of MR for more than visualisation. Stilman *et al.* [26] 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 setup uses an array of cameras that track objects in the experimentation environment to form a representation of the real world where virtual elements can be inserted for testing planning and control algorithms. Kobayashi *et al.* [27] and Nishiwaki *et al.* [28] later named this an "MR environment," and apply AR visualisations techniques for extensive debugging of motion planning and vision based recognition algorithms. 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.

Kuroda *et al.* [4] 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 [29]. Inspired by AR, the work proposes a classification that considers hybrid simulations an advancement of conventional HIL

simulations, where both simulated and real robots/sensors can be used in simulation. This classification has been extended by Davis *et al.* [30] to explicitly define HIL simulation and hybrid simulation as a form AV and AR respectively. An AR framework is proposed for simulating AUV operations. While the framework allows real and virtual components to be used interchangeably in simulation, interactions between them are limited to only sensing components, i.e., augmenting real sensory readings with virtual input. This makes it difficult to reuse the framework for simulation in other applications involving tasks such as collisions, and object manipulations.

Göktoğan and Sukkarieh [31] also propose an AR framework for experimenting cooperative control of a team of Unmanned Aerial Vehicles (UAV) in target detection tasks. The AR framework overcomes resource limitations 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.

The review identified that existing AR or MR based HIL simulators are primarily application-specific. There appear to be no formalised or generalised methods for constructing simulations that facilitate interactions between real and virtual objects. A software model for creating standardised implementations of simulation systems is also missing. Moreover, there is a lack of effort in validating these simulation tools.

3 Requirements Analysis

To address the issues identified in Section 2, it is necessary to design solutions that are reusable and customisable for different robots and applications. In addition to solving specific problems of existing HIL/hybrid simulation approaches, we must keep in mind the overall goal and consider in our solution more effective ways to transition from simulation to deployment. We propose requirements for implementing a flexible and enhanced environment for robot experimentation.

Unified Testing of Robot Components: In HIL simulation, robot developers can develop an algorithm and incrementally experiment and test the different robot designs, using simulated components as necessary, before the actual hardware/software components such as sensors or an arm are available. An important requirement for achieving such incremental development is that the simulated components must be able to interact with the rest of the system so that the robot could correctly function as a whole [26]. The integration should be achieved with minimal or no changes to both real and simulated components of the robot system.

Robot-Environment Interactions: Existing work on HIL simulation such as [32], [6] has indicated the importance of simulating the robot's interaction with the environment. The robot environment often plays a major role on the influence of robot behaviour in robot operations and should be included into the loop of the experimental design. 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. The real robot should perceive and interact with the simulated objects as if they exist in the same coherent space, and likewise, these simulated objects must be able to interact with the dynamic environment they reside in.

Standardised Interfaces: Martin and Emami [8] point out that a generic and modular robotic HIL simulation architecture can lead to a more reconfigurable and reusable simulation platform, with a key feature for ensuring modularity being interfaces for communication between software and hardware components. To this end, it is proposed that a requirement for improving the reusability of HIL simulations is to adopt standardised frameworks/interfaces for integrating simulated components with the existing experimental setup. This requirement is also essential for supporting unified testing of robot components and facilitating robot-environment interactions.

Variable Test Conditions: Existing hybrid simulation research in underwater robotic applications [30], [29] has emphasized the support for testing robot systems in varying test conditions. This includes varying properties of the test environment which determine the degree to which it represents the target environment in the real world. For example, depending on the progress and the requirements, the real robot controller could be tested either with simulated hydrodynamic forces, or in a physical water tank. For applications that demand high fidelity simulation, the complexity and realism of the simulation could be increased to obtain more accurate results at the expense of higher modelling cost.

3.1 Solution

To fulfill the requirements, this paper proposes that simulation environments be based on the concept of MR, specifically Milgram *et al.*'s Reality-Virtuality continuum [33], [34], and presents and evaluates an illustrative design and implementation prototype. 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 can design a simulation environment with the desired balance of realism and safety as required. The new simulation approach is referred to as MR simulation, which enables users to:

- vary the balance of real versus virtual components over the development process to speed up the development cycle and increase reliability and stability of the solution,
- replace dangerous components in the experimental setup with safe and virtual counterparts,

- simulate resources that are expensive or unavailable and observe them interact with real components,
- create different complex test scenarios (of real and virtual components) for experimentation.

We focus on implementing a reusable solution, proposing three essential features that must be integrated to help shape the design of a generic simulation environment that can be applied to the development and testing of a broader range of robots and applications. The first feature is a *generic, conceptual framework* that formalises the representation of environments composed of real and virtual components, and the interactions between them. The framework provides the basis for creating all MR simulations described in this paper. The implemented MR simulation system integrates the second feature, an *open source, general purpose robot simulator*, that provides readily available and extensible technology for simulating different robot and sensor systems. The implementation also integrates the last feature, *standardised communication interfaces*, in its design to support communication between heterogeneous software components for increased interoperability.

4 Generic MR Framework

As part of the overall solution to provide an enhanced robot experimentation environment, this section presents a conceptual framework, referred to as an MR framework [35], that is inspired by concepts found in the literature of MR. 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. An extension of the framework to robot simulation will be shown in Section 5.

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. 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 MR Entity

The first step in constructing an MR environment is to create appropriate representations of objects that constitute the environment. 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 [34]. The MR entity is modelled with an attribute called *level of physical virtualisation* which describes the degree to which an entity's physical characteristics are virtualised (or digitised) with respect to an object in the real world.

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 [36], [37], [38]. Our 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 leaf node in the tree can contain zero or more objects, and these objects are graphical and/or physical representations of the associated entity. Inspired by the work of [39] that combines physical and virtual objects in an MR application, our 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. An entity model consisting of real and virtual entities 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 1. An entity model is classified as real or virtual if all of its successor nodes belong to the same class, e.g., Figure 1a) and 1b), and is classified as augmented if it has at least two successor nodes of different classes of reality, e.g., Figure 1c). An entity model can also be composed of other entity models, e.g., Figure 1d)

In the implementation level, the value of physical virtualisation of an entity model is computed based on the class of reality of the nodes in the tree. The value is useful for an MR system to determine how a real-virtual interaction should be executed (see Response Generation in Section 4.2.3).

4.2 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. In an MR environment, an entity 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 sensor, or an environmental object. To facilitate interaction between real and synthetic entities, we need to model the process of transforming actions into effects [40].

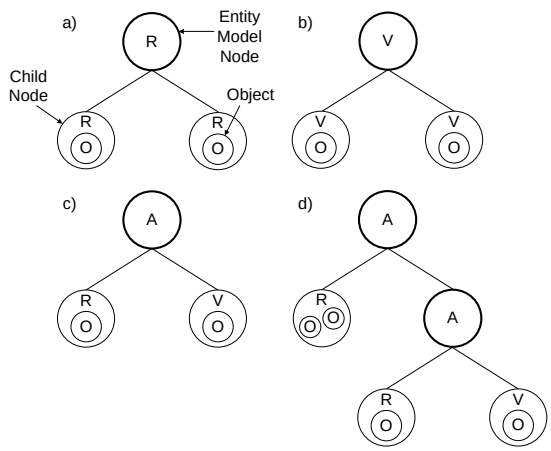

Fig. 1. Entity models represented using scene graphs. R = Real, V = Virtual, A = Augmented, O = Object.

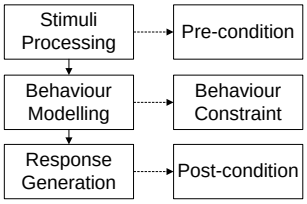

Fig. 2. Mapping of interaction stages to three constraints.

Simply put, if we know the interaction is possible between two objects in the real world, we can try to reproduce the results for interactions between real and synthetic entities. To achieve this, it is required to model and generate the expected behaviours of the participating entities as if the interaction had happened. We need to know how a participating entity reacts to given stimuli, behaves under certain constraints, and also how its response can be generated, i.e., $Stimuli \rightarrow Behaviour \rightarrow Response$. 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 2. State transitions within an MR interaction are illustrated in Figure 3.

4.2.1 Stimuli Processing

As two entities engage in interaction, 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.

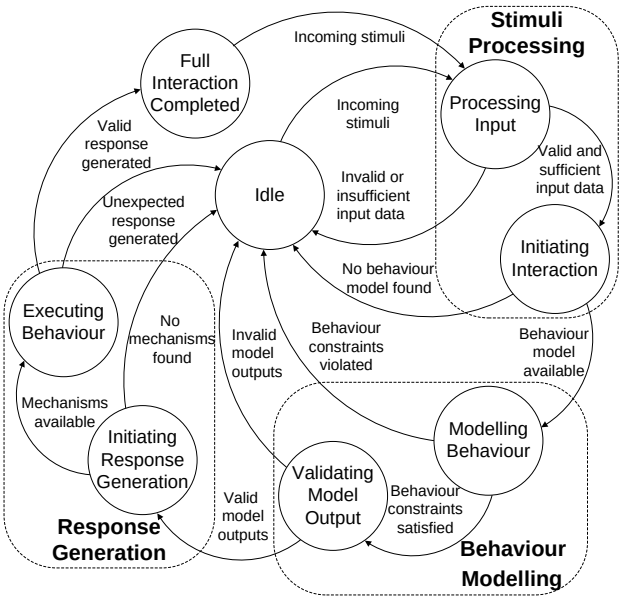

Fig. 3. State transitions in an MR interaction.

4.2.2 Behaviour Modelling

Once the input data has been processed and interaction has been confirmed to proceed, the behaviours of the entities are modelled with the given 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 entities' behaviours are bounded by or other mathematical constraints that need to be satisfied.

4.2.3 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.

The responses generated can be classified into two types:

1) The "physical response" is the result from all laws of physics being consistently applied to the entity. e.g.,

applying external forces to the physical object associated to the entity to generate a natural response.

2) The "logical response" is generated by taking into account the entity's functional capabilities. e.g., altering the state of the entity through its internal controllers.

It is important to note that a full interaction may not always be achieved (see Section 8 for issues to consider). 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.

5 Application to 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. Formalising representations and interactions in an MR environment helps to provide a generalised approach to conducting HIL based simulations for the incremental and parallel development of software systems. 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 a robot arm are available. This section illustrates how the MR framework is applied to creating MR environments for robot simulation and facilitating interactions in common robot tasks.

5.1 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. It also facilitates repeated testing under the different experimental configurations and environment properties. Robots, physical objects in the environment (including humans), as well as objects that do not possess a physical form can be extended from the MR entity. Figure 4 illustrates an example.

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, as illustrated in Figure 5. Each leaf node in the tree can be real or virtual. For instance, when a robot 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 robot arm may require employing surrogate devices for moving the gripper according to the calculated motions of the virtual arm. 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 virtual components.

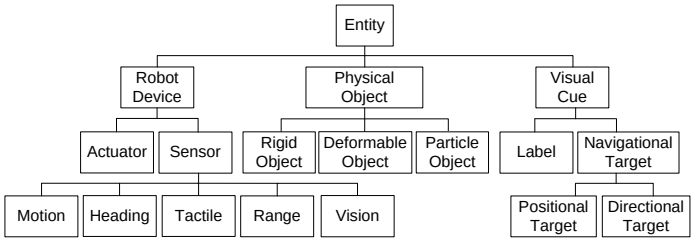

Fig. 4. An inheritance diagram of an entity in simulation.

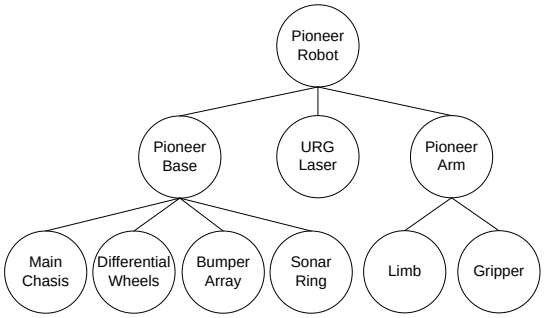

Fig. 5. An example entity model of a Pioneer robot.

5.2 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 [32]. Sensors 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. 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.

5.2.1 Sensor based Interaction

Sensor based interaction is concerned with interactions between a sensor device and other entities in the environment.

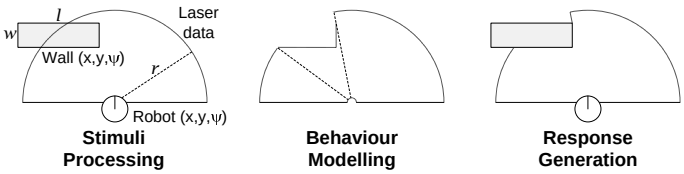

Fig. 6. An example sensor based interaction between a range sensor device and a wall. Stimuli Processing: collect pose of both entities and the raw range data and configurations of the range sensor device. Behaviour Modelling: compute expected obstructed volume from raw range data. Response Generation: alter the output of the range sensor device accordingly.

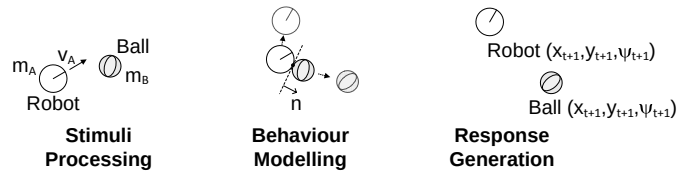

Fig. 7. An example actuator based interaction (collision) between a robot and a ball. Stimuli Processing: collect properties such as entity bounding boxes, masses and velocities. Behaviour Modelling: identify colliding point, collision normal, time of collision, and calculate impulses to be applied to bodies in collision. Response Generation: move the two entities accordingly over time.

Sensor devices are essential components of a robot system for robots to interact with and gain understanding of the environment. We focus on 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 6. 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 process 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.

5.2.2 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 7. 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 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.

In interactions between non-robotic components, e.g., the virtual ball rolling into a real box, the three stages carried out are similar. In practice, an external sensor such as a camera would be placed in the MR environment to track and maintain the state of the real box in the Stimuli Processing stage. In Response Generation, it is possible, though expensive, to achieve a full interaction by generating the response of the real box over a motion platform in a similar manner to [6]. Alternatively, a partial interaction can be created by choosing only to visualise the outcome without physically altering the state of the real entity.

5.2.3 Extending to Other Interactions

Other interactions in tasks required of robots can also be classified into sensor based and actuator based interactions, and modelled in a similar way to the examples described

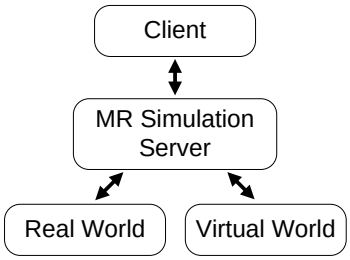

Fig. 8. Overview of elements in MR robot simulation.

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.

6 Software Implementation

To validate the concepts, an MR robot simulator has been implemented. It integrates two features described in Section 3.1 that leads to a reusable solution, 1) general purpose robot simulator, and 2) standardised communication interfaces. This section summarises the implemented system.

6.1 Overview

Figure 8 shows the essential elements in an MR simulation.

- The **client** is the robot software program being developed and tested in simulation.
- The **MR simulation server** is the primary element 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.

The client robot program now exchanges messages with the MR simulation server, instead of exchanging directly with real or simulated devices. The merging of the real and virtual world is transparent to the client who sees real and virtual entities as part of the same coherent world. More importantly, no modifications are necessary to the client program code to use the MR robot simulator.

An MR Simulation toolkit, named 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, we take advantage of existing robot simulation technology and integrate MRSim with a general purpose robot simulator. The Gazebo 3D mobile robot simulator [10] was chosen. It is open source, modular, highly modifiable, and has independent open source rendering and physics subsystems which facilitate the integration of MR technology.

6.2 System Architecture

MRSim integrates well with Gazebo, but it is designed in an application independent fashion using 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. Figure 9 illustrates how MRSim is integrated into Gazebo.

It was seen that the MR simulation server communicates with the real and virtual world for creating MR simulations. MRSim does so by exchanging messages with real and simulated objects over a *device interface* and a *Gazebo interface* respectively. These communication channels are achieved using open source robotic software frameworks, such as Player [41] and OpenRTM [42], that provide platform- and language-independent interfaces to external robot software components. These software frameworks adopt standardised data interfaces for communication, and support network transparency, distributed computing, and hardware abstraction, all of which are important for increasing the interoperability of the system. The device interface (potentially a Player server or an OpenRTM's RT Component running on the physical robot device) provides data captured by robot devices in the real world, while the Gazebo interface (potentially the Player plug-in driver distributed by Gazebo or the OpenRTM simulation interface layer [43]) 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

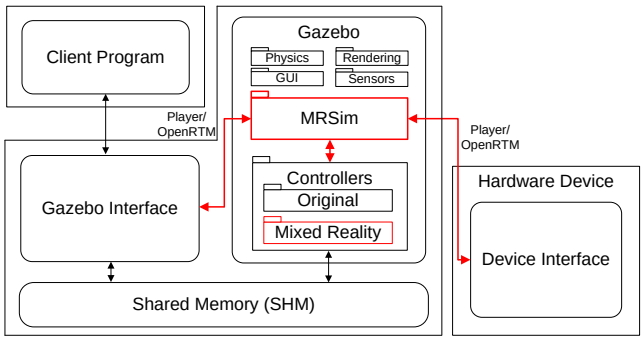

Fig. 9. MRSim integrated into Gazebo. The new modules and data flows are highlighted in red. A set of mixed reality controllers have also been added to Gazebo to facilitate communication between MRSim and the client program.

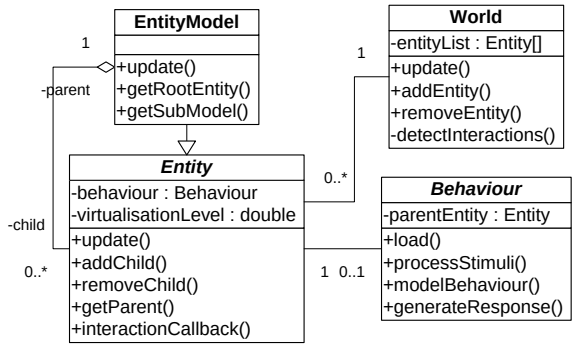

Fig. 10. A UML diagram showing core classes in MRSim.

continue monitoring 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 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.

6.3 MRSim

MRSim is implemented based on the generic MR framework. A UML class diagram capturing the core classes of MRSim and the proposed interaction scheme is shown in Figure 10. 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 10, the *World* is responsible for managing all *Entities*, including *Entity Models*, and updating them at each iteration of the simulation loop. For each entity in the world, a corresponding virtual representation is created in Gazebo. The world is then able to keep track of any physical interactions between the entities by monitoring Gazebo's

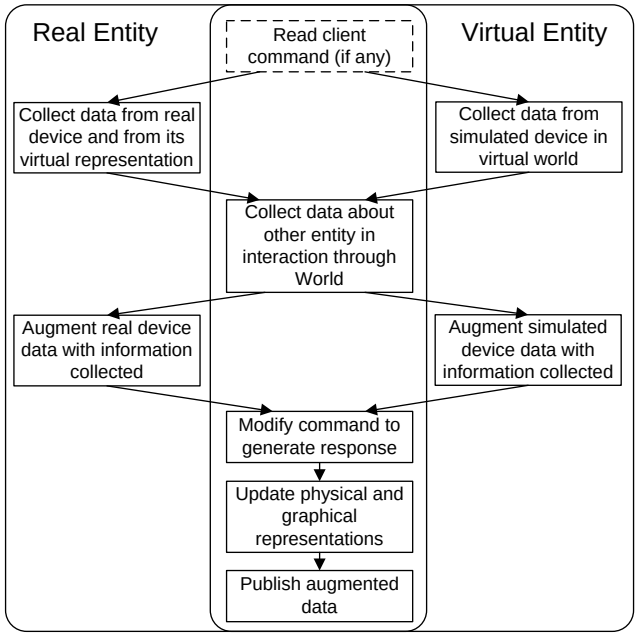

Fig. 11. Work flow inside a behaviour object attached to an entity. The work flow varies slightly between a real entity and a virtual entity, as separated to the left and right columns respectively.

physics simulation. An interaction map is 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 custom behaviours that facilitate sensor based and actuator based interactions.

Over the course of the work, a number of entities and behaviours have been implemented for creating MR simulations, and their designs are based on common interfaces and data types in Player and OpenRTM. The set of implemented entities includes: Position, Actuator Array, Laser, Camera, Force-Torque, and Rigid entities. Figure 11 illustrates the work flow of the implemented behaviours attached to these entities. Although the set of entities are far from being a complete database of the diverse robot devices available or the different types 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 extensible, and other entities and behaviours can be added in the future.

7 Evaluation

Thorough evaluation of a simulation tool is important to extrapolate its use in practice. In the area of virtual simulations, USARSim [44] is a robot simulator that has undergone

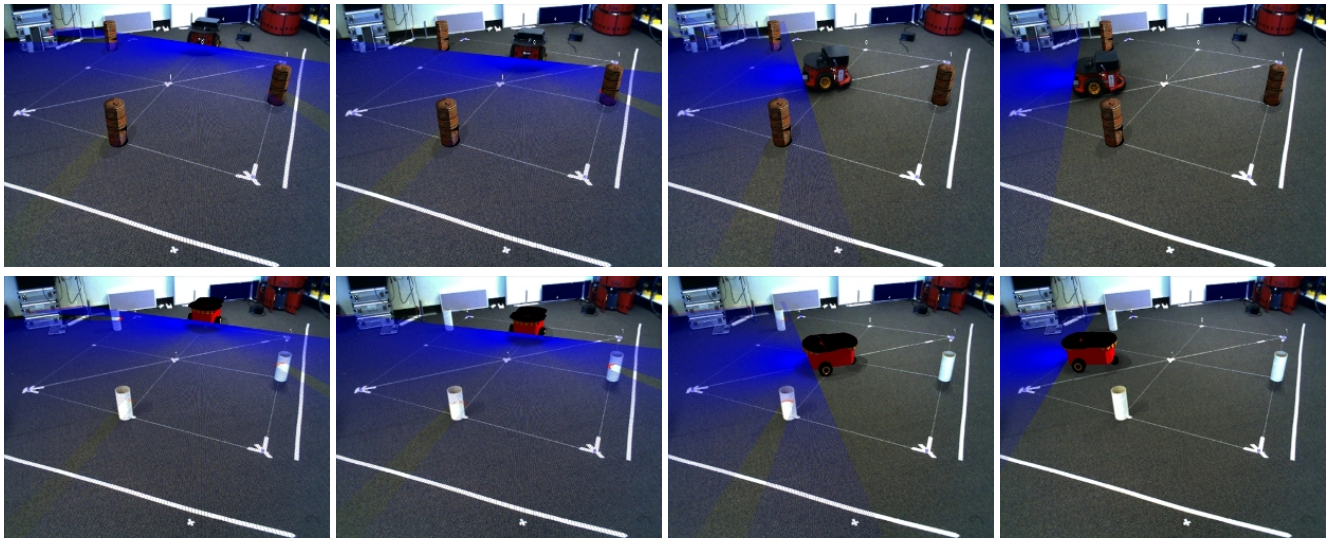

Fig. 12. A sequence of screenshots illustrating a sensor based interaction between a laser sensor device and three cylindrical objects in the simulation environment. Top Row: A real robot detects and avoids virtual obstacles. Bottom Row: A virtual robot detects and avoids real obstacles.

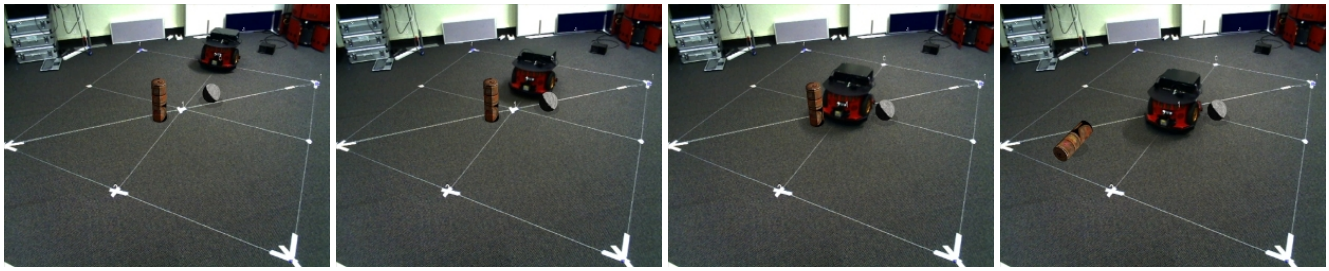

Fig. 13. A sequence of screenshots illustrating a contact interaction between a real robot and two virtual objects.

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 for robot development. This section validates the implemented MR robot simulator, examines the reusability of the framework, and analyses its benefits to the overall robot development process.

7.1 Functional System Validation

This evaluation validates the use of our MR robot simulator for creating the two types of interactions described in Section 5.2. To demonstrate an example of sensor based interaction, an MR simulation of an obstacle avoidance algorithm is created, see Figure 12. The robot (modelled as a position entity), is equipped with a laser sensor (laser entity), and randomly navigates around the environment and avoids obstacles (rigid entities) by analysing its laser range readings. The same obstacle avoidance algorithm was run on a real and a virtual robot equipped with a real and a virtual laser sensor respectively. Virtual obstacles were placed in the environment in the case of a real robot, while real obstacles were used in the opposite case involving a virtual robot. The robot successfully detected the presence of the obstacles in both cases and navigated towards the open space. In contrast to the interaction examples described in Section 5.2, a virtual robot/sensor is shown in this section. The advantage of using a virtual robot in the physical environment is the ability to help robot developers see and validate the behaviour of their system in its intended physical environment [24]. Note that in this example, the real obstacles were each associated with a pre-modelled virtual representation in order to create its interaction with the virtual sensor. It is acknowledged that the additional modelling could mean more work for the robot developer, thus recent extensions of the MR robot simulator have integrated a vision system for tracking the states of real objects in the MR environment so that their virtual representations can be dynamically created and updated.

An example of actuator based interaction is shown in Figure 13. A real robot was operated to move until it collided with virtual objects in its path. The Gazebo's physics engine

was used to model the dynamic behaviour of the 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 will be shown in section 7.2.

7.2 Framework Reusability

To demonstrate that the MR framework is generic, we have conducted three case study evaluations that deployed the MR robot simulator to aid the development of different robot systems and in different applications. The first application extends the functional system validation and creates an MR simulation involving the same ground robot system for a search and rescue scenario [45]. Any potential threats to the robot in the simulation environment were virtualised. A real ground robot was deployed to autonomously navigate in a physical environment filled with real and virtual hazards while searching for the specified target using vision.

The second application uses MR simulations to provide a cost-effective solution to testing of a robotic screw remover system in an industrial building interior demolition task [46]. The task required a robot manipulator system to remove screws from suspended ceiling beams, and MR simulations were created to simulate virtual screws and beams for repeated testing of the screw removal operation, see Figure 14. A full actuator based interaction was achieved as the robot manipulator physically unscrewed the virtual screws using a real custom screw removal tool mounted on it's end-effector.

The last application uses MR simulations as a safe alternative for prototyping a UAV system to be deployed for an agricultural cow monitoring task. To provide robot software developers as much insight to real world tests as possible, a real UAV was deployed in flight to autonomously follow a virtual moving cow using vision, see Figure 15.

Throughout these case study evaluations, the implemented MR robot simulator was also shown to be interoperable between different robotic software frameworks. The MR robot simulator was able to instantiate RT Components that exchange augmented data with the rest of the OpenRTM screw remover system, see Figure 16. It also subscribed to Player devices on the UAV system, and published augmented sensory data to the client programs being developed, see Figure 17.

7.3 Simulation Credibility

It is important that the MR robot simulator is able to produce simulations that reliably represent the real world. A quantitative evaluation was conducted that compared the robot behaviour generated by the obstacle avoidance algorithm in MR simulations with virtual simulations and real experiments [35]. MR simulations involved a real Pioneer robot avoiding virtual obstacles, while Gazebo created the virtual simulations. The robot in all three conditions took on a similar path to avoid the obstacles, but Gazebo produced minimal variations between the robot trajectories as the result of missing noise models, and the rigid body dynamics engine's approximations for friction models (pyramid friction cones), collision resolution (interpenetration depth, correcting forces, contact surface layers), and accuracy (step sizes, numerical solvers, etc); these are common causes of discrepancy between simulations and the real world. 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. The results show the MR simulation produced robot trajectories with

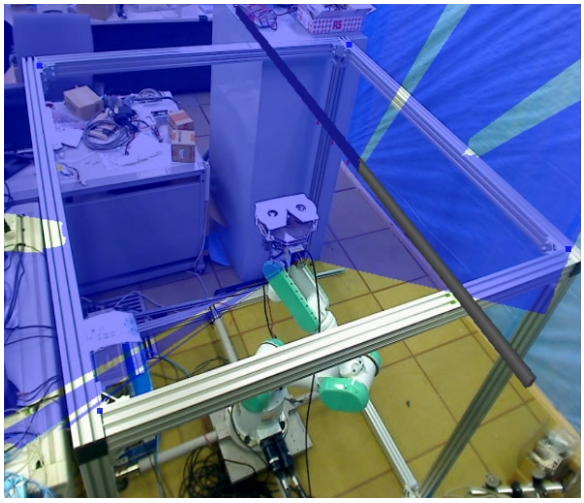

Fig. 14. An AR view of the screw remover simulation.

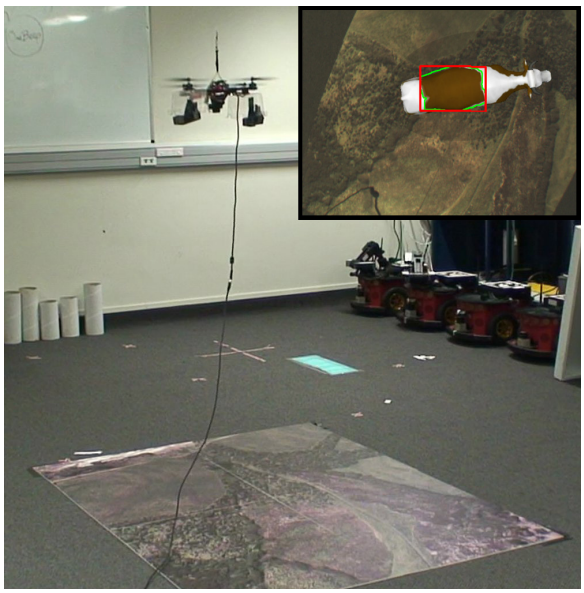

Fig. 15. Indoor UAV system test setup. The UAV hovers over a mock-up agricultural scene. Top right: image processing carried out by the robot for cow detection.

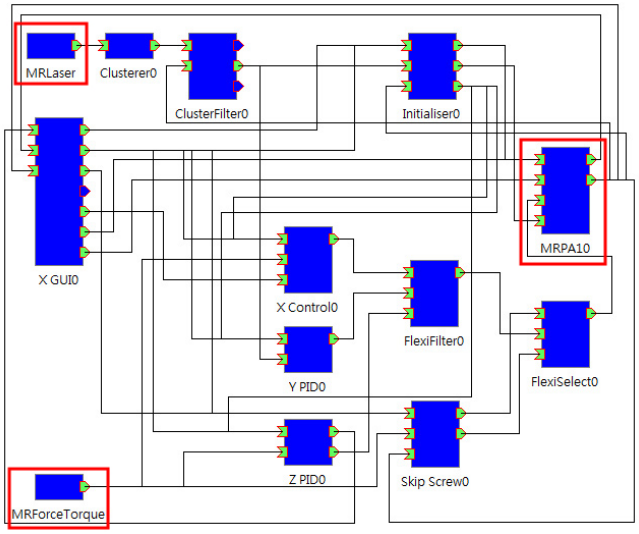

Fig. 16. System diagram showing the software components in the screw remover MR simulation. Highlighted in red are MR sensors (MRLaser, and MRForceTorque) and an MR robot manipulator (MRPA10) that exchange data with the original OpenRTM screw remover system.

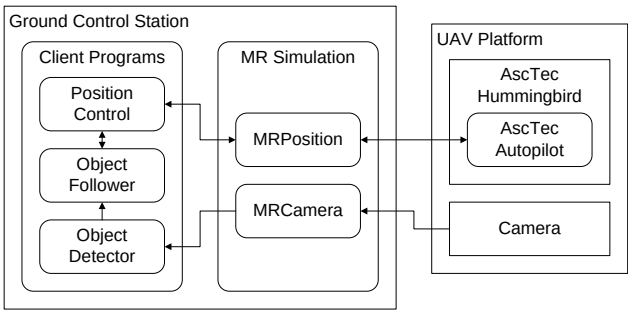

Fig. 17. System diagram of the MR simulation for prototyping the UAV system. Clients process augmented images generated by MRCamera for cow detection.

small deviations from the ones in the real experiment, with an average Root Mean Square Deviation (RMSD) of 0.02m in x and 0.03m in y. 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. This simple evaluation validated the MR robot simulator's accuracy. The MR robot simulator enabled real components to be incorporated in simulation, relieving the need to simulate complex real world variations. The solution proved to be viable as illustrated by the results.

In the evaluation of the screw remover system, MR simulations were also found to produce reliable simulation results. After comparing the trajectories of the robot's end-effector in MR simulation with real world tests over five runs, 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; the deviation was considered sufficiently accurate for the task.

7.4 Usability

Two user studies conducted during the case study evaluations have indicated positive contributions of MR simulations to the robot development process [47]. The first user study involved participants operating the screw remover system in virtual simulations and MR simulations. The study measured user performance using objective metrics, e.g., task completion time and set-up time, and collected user opinions and thoughts using subjective questionnaires. The collected data were then analysed using statistical tests to identify significant differences between the two conditions. The results show that 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, $p < 0.05$).

We also used our MR robot simulator in an observational user study to help robot programmers implement software for the UAV based cow monitoring task. Findings suggested that virtual simulations and MR simulations were complementary, each contributing to different stages of the development cycle. Compared to virtual simulation, the users spent 38% more time using MR simulations in the mid and later stages of the development (evaluation and tuning), and they rated the use of MR simulations significantly higher for helping them tune control parameters of the UAV system for better performance (Z = -2.81, $p < 0.05$). Building the virtual simulation and the MR simulation on the same simulation platform (Gazebo) also allowed users to switch from one method to another without learning to use separate controls and interfaces, enabling them to transition to a more physical environment using the same development tool.

7.5 Cost and Efficiency

The case study evaluations provided a glimpse of how MR simulations may help to address issues related to cost and efficiency during robot development. For example, testing of the screw remover system was originally carried out using real world tests which were constrained by the limited number of spare beams and screws available. Simulating virtual substitutes of these spare parts presented an option to minimise resource requirements and prevent damaging the custom screw removal tool in case of failures. Moreover, the standardised data interfaces allowed the MR robot simulator to be immediately deployed for testing the screw remover system without making modifications to the MR robot simulator or the software system being tested. This could help to minimise the large overhead commonly required to build and set up an HIL simulation system in the industry [3].

A more thorough and formal evaluation is required in future work to verify these findings. This could 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.

8 Discussions

Throughout the evaluations, the MR robot simulator has been shown to successfully create different sensor and actuator based interactions between real and virtual entities, demonstrating benefits to the robot development process in three applications. A number of important lessons were learnt during the evaluation of our MR framework and the MR robot simulator. In particular, 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.

It is often difficult and time-consuming to collect information of objects that exist in the real 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 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 interaction outcome.

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 a collision, forcing a real object to physically change its pose or motion (i.e., a physical response) 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.* [6]. In their hybrid simulation, the use of a nine degree-of-freedom motion table safely generates the resulting motions from a collision between real robot manipulators and a target object (i.e., a logical response). Nevertheless, hardware mechanisms for executing responses can be expensive, thus, a partial contact interaction is sometimes preferred depending on the requirements.

9 Guidelines

General lessons learnt from developing MR simulations have been formulated into guidelines for future MR simulation developers and users. The guidelines help users to 1) determine the appropriate use of real and virtual objects for designing safer MR simulations, 2) design simulation scenarios based on the requirements of the task, and 3) understand the appropriate use of MR simulation to maximise benefits.

9.1 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 algorithm 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.

9.2 Realism vs. Task Requirements:

In applications where consequences of malfunctions are too severe, a virtual environment with high fidelity simulation models should be used, whereas for applications involving a complex but low risk environment, modelling is unnecessarily costly and so the overall level of virtualisation can be reduced. However, lowering the level of virtualisation does not always improve simulation realism. The quality of entity representations and behaviour models influence realism, and determine whether MR interactions can be completed naturally. Full MR interactions should be enforced unless safety becomes an issue or hardware mechanisms for executing responses are unavailable. If partial interactions are preferred, the final outcome of the interaction should be communicated to the users so that they are able to understand the resulting behaviour of the system being tested in simulation, e.g., by visualising subsequent movements over the physical robot [48].

9.3 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, and should not require expensive equipment to set up. It is important to keep in mind that the advantages inherent in using existing simulation methods should not be neglected. When an MR simulation includes real hardware 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 offline simulation tools. 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 [49]. Benefits of MR simulations are more evident in the mid and later stages as found in our user studies. MR simulations should be used as a complementary step in the development cycle, and not as a replacement of all existing methods.

10 Conclusions and Future Work

This paper has presented an MR simulation framework for robot software development. It provides a generalised and reusable simulation environment for the incremental and parallel development of both simple and complex robot systems across different applications. One of the main contributions is a generic MR framework for implementing MR systems. We have demonstrated that it can be used to create rich interactions in simulations including both sensor augmentations of robots and physical interactions between real and virtual entities in the environment. An MR robot simulator has been implemented based on the MR framework. It shares the same goal as open source robotic software frameworks such as Player and OpenRTM to improve interoperability between software components in robot software development. By building on a general purpose robot simulator and integrating standardised data interfaces for communication, the simulator is capable of supporting simulation of a wider range of robot systems.

Future work will focus on conducting a long term observational study to investigate the use of MR simulations in larger robot development projects. This includes identifying the ease and difficulties of robot developers designing, building, and running MR simulations themselves based on given requirements, and the impact of using MR simulations on the development cost and efficiency.

References

[1] J. Banks, *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*. Wiley-Interscience, 1998. 1

[2] 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. 1

[3] J. Ledin, "Hardware-in-the-loop simulation," *Embedded System Program*, vol. 12, no. 2, pp. 42–60, February 1999. 1, 7.5

[4] 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. 1, 2

[5] V. Gavrilets, E. Frazzoli, B. Mettler, M. Piedmonte, and E. Feron, "Aggressive maneuvering of small autonomous helicopters: A human-centered approach," *The International Journal of Robotics Research*, vol. 20, no. 10, p. 795, 2001. 1

[6] 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. 1, 3, 5.2.2, 8

[7] J. Biesiadecki, D. Henriques, and A. Jain, "A reusable, real-time spacecraft dynamics simulator," in *Proceedings of the AIAA/IEEE Digital Avionics Systems Conference*, vol. 2, 1997, pp. 8–2. 1

[8] 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*, 2006, pp. 2162–2167. 1, 3

[9] R. Vaughan, "Massively multi-robot simulation in stage," *Swarm Intelligence*, vol. 2, pp. 189–208, 2008. 1

[10] 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*, September 2004, pp. 2149–2154. 1, 6.1

[11] 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. 1

[12] R. Diankov and J. Kuffner, "OpenRAVE: A planning architecture for autonomous robotics," Robotics Institute, Carnegie Mellon University, Pittsburgh, Tech. Rep. CMU-RI-TR-08-34, 2008. 1

[13] 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, p. 6. 2

[14] 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. 2

[15] 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. 2

[16] 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. 2

[17] 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. 2

[18] 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. 2

[19] 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. 2

[20] 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*, vol. 25, no. 1, pp. 54–63, January–February 2005. 2

[21] C. Nielsen, M. Goodrich, and R. Ricks, "Ecological interfaces for improving mobile robot teleoperation," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 927–941, 2007. 2

[22] T. Collett and B. MacDonald, "An augmented reality debugging system for mobile robot software engineers," *Journal of Software Engineering for Robotics*, vol. 1, no. 1, pp. 18–32, 2009. 2

[23] 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. 2

[24] 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: ACM, 2006, pp. 237–244. 2, 7.1

[25] "RoboCup mixed reality league," http://sourceforge.net/projects/pv-league/. 2

[26] M. Stilman, P. Michel, J. Chestnutt, K. Nishiwaki, S. Kagami, and J. Kuffner, "Augmented reality for robot development and experimentation," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-55, November 2005. 2, 3

[27] 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. 2

[28] 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 2008, pp. 2211–2212. 2

[29] P. Ridao, E. Batlle, D. Ribas, and M. Neptune, "Neptune: a HIL simulator for multiple UUVs," *IEEE TECHNO-OCEAN*, vol. 4, pp. 524–531, 2004. 2, 3

[30] 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. 2, 3

[31] 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. 2

[32] 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. 3, 5.2

[33] P. Milgram and F. Kishino, "A taxonomy of mixed reality visual display," in *IEICE Transactions on Information Systems*, vol. E77-D, no. 12, December 1994, pp. 1321–1329. 3.1

[34] P. Milgram and H. Colquhoun, "A taxonomy of real and virtual world display integration," *Mixed Reality-Merging Real and Virtual Worlds*, pp. 5–28, 1999. 3.1, 4.1

[35] 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. 4, 7.3

[36] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavari, L. Encarnacao, M. Gervautz, and W. Purgathofer, "The Studierstube Augmented Reality Project," *Presence: Teleoperators & Virtual Environments*, vol. 11, no. 1, pp. 33–54, 2002. 4.1

[37] 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. 4.1

[38] J. Looser, R. Grasset, H. Seichter, and M. Billinghurst, "OSGART - a pragmatic approach to MR," in *5th IEEE and ACM International Symposium on Mixed and Augmented Reality: Industrial Workshop*, Santa Barbara, CA, USA, October 2006. 4.1

[39] 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. 4.1

[40] 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*, vol. 11, no. 6, pp. 677–686, 2002. 4.2

[41] 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)*, Coimbra, Portugal, June 2003, pp. 317–323. 6.2

[42] 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: Springer-Verlag, 2008, pp. 87–98. 6.2

[43] 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. 6.2

[44] 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. 7

[45] 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. 7.2

[46] 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. 7.2

[47] I. Y.-H. Chen, "Mixed reality simulation for mobile robots," Ph.D. dissertation, The University of Auckland, 2011. 7.4

[48] 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. 9.2

[49] 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*, vol. 5, no. 3, pp. 185–223, 1998. 9.3

Ian Chen received his BE (1st class) and Ph.D in the Electrical and Computer Engineering Department of the University of Auckland, New Zealand in 2007 and 2011 respectively. He is now a postdoctoral researcher at the University of Auckland working in the areas of healthcare robotics and industrial automation. His key research interests are in mixed reality, augmented reality tracking, and robot simulation.

Bruce MacDonald received a BE (1st class) and Ph.D in the Electrical Engineering department of the University of Canterbury, Christchurch, New Zealand. He spent ten years in the Computer Science department of the University of Calgary in Canada then returned to New Zealand in 1995. There he joined the Department of Electrical and Computer Engineering Department at the University of Auckland, led the initiation of a new Computer Systems programme, and set up the Robotics laboratory. His research interests include robotics and intelligent systems.

Burkhard Wünsche is a Senior Lecturer in Computer Science at the University of Auckland, New Zealand. His research interest include visualisation, computer graphics, human-computer and human-robot interfaces, and healthinformatics. He is the director of the Graphics Group and the Division for Biomedical Imaging. He has published more than 90 book chapters, journal and conference articles, and he has participated in industry projects related to information visualisation, biomedical imaging, robotics and telehealthcare.