Imperial College
London

SHARED CONTROL FOR HANDHELD ROBOTS

JOSHUA ELSDON

PhD Thesis

Supervised by PROFESSOR YIANNIS DEMIRIS

Personal Robotics Lab

Department of Electrical and Electronic Engineering

Imperial College London

May 2019

COPYRIGHT DECLARATION

ORIGINALITY DECLARATION

I hereby declare that this thesis and the work herein detailed, was composed and originated by myself, except where appropriately referenced and credited.

London, May 2019

Joshua Elsdon

ABSTRACT

Hand-held robots are an exciting new extension to the toolkit of hand tools at our disposal. They allow the benefits associated with robots, such as precision and task specific knowledge, to be integrated into a form factor that is convenient and cost effective.

This work isolates two categories of hand-held robots: *reduced degree of freedom* robots must rely on the human for at least some degrees of freedom at the end effector; and *locally capable* robots have the facility to completely decouple user motion of the handle of the robot from that of the end effector. A robot of each variety is designed and built for a set of five degree of freedom tasks, and algorithms for generating end effector trajectories are implemented for both.

Both types of hand-held robots rely on a close collaboration with the user to be successful. To facilitate information flow from the robot to the user an Augmented Reality (AR) headset is used. The level of visualisation detail required to form an effective collaboration is investigated. It is found that the level of detail is less important to performance when the robot has control of the interaction, though high fidelity visualisation are important when the user is in control of the integration.

Which agent is in control of the interaction at a *tactical* and *operational* level is also analysed. *Reduced degree of freedom* robots must share these levels of planning with the user, and this is found to cause conflict with some users. Complete allocation of these levels to an agent is possible with *locally capable* robots. No strong evidence of conflict was observed when these roles were more clearly allocated to the user or the robot.

ACKNOWLEDGEMENTS

I will take this opportunity to thank all of those who have made this work possible. In some way this includes everyone who has wished me well over the years, however there are some I would like to particularly highlight.

The James Dyson Foundation supported me financially during these studies. This purely charitable support of research is highly noble, and I am sincerely grateful to them.

I would like to extent my thanks to Professor Yiannis Demiris, who supervised me throughout the project and provided the genesis of this project. His guidance ensured that I kept my eye on the larger goals, and did not become (too) preoccupied with other things. He has cultivated fantastic work environment within the *Personal Robotics Lab*, ensuring a fantastic mix of talents are well represented, as well as a group of genuinely good people to be around.

To the members of the *Personal Robotics Lab* both past and present, I would like to thank you all for your advice and enthusiasm. I am certainly a better engineer for having known you all. I expect to see great things from you all.

I must also extend thanks to the group of people at the *Imperial College Robotics Society*, the old timers, those who come and go, and the fresh faces. The society provided a place to belong and form a connection with robotics, as well as life long friendships and connections with very many people.

My family have been unendingly supportive of my pursuit of my education, and I hope that what I can build in the future will justify their belief in me.

Finally I would like to thank my wife, whom I married during this project, for being my partner in crime, and for dreaming of the future with me.

CONTENTS

1 INTRODUCTION 19

1.1 Reduced Degree of Freedom and Locally Capable Hand-held Robots 22

1.2 Research Questions 24

1.3 Contributions 25

1.4 Outline of the Thesis 26

2 BACKGROUND WORK 28

2.1 Hand-held Robot Definitions 28

2.2 Shared Control Definitions 29

2.3 Existing Hand-held Robots 30

2.3.1 Gregg-Smith: Cooperative Hand-Held Robots 32

2.3.2 Micron: Eye Surgery Hand-Held Robot 32

2.3.3 Hand-held Liquid Distribution Robots 33

2.3.4 Notable Non-Hand-Held Robots and Devices 34

2.3.5 Comparison and Summary of Existing Hand-held Robots 35

2.4 Visualisations and instructions 36

2.5 Path Planning for Spraying Applications 37

2.5.1 Introduction 37

2.5.2 Coverage Path Planning 37

2.5.3 Car Body Spray Coating 38

2.5.4 Simulation of Paint Density 39

2.5.5 Non-Uniform Spraying 39

2.6 Tracking Robots in 6 Degrees of Freedom 39

2.6.1 Sensor Fusion 40

2.6.2 Augmented Reality Calibration 43

3 HAND-HELD ROBOTS AND THEIR IMPLEMENTATIONS. 45

3.1 Introduction 45

3.2 Single Axis Hand-held Spraying Robot 46

3.2.1 Version One 47

3.2.2 Version Two 47

3.2.3 Version Three 49

3.3 Five Axis Hand-held Drawing Robot 53

3.3.1 Five Axis Hand-held Drawing Robot Construction 57

3.4 Tracking of the Hand-held Robots 62

3.4.1 Retro-reflector Based Infra-Red Motion Capture 62

3.4.2 Sensor Fusion of IMU and Optical Motion Tracking 63

3.5 Conclusion 66

4 AUGMENTED REALITY ALIGNMENT 68

4.1 Introduction 68

4.2 The *SVD method*: Finding a Transform to a Meaningful Local Frame 69

4.3 Augmented Reality World Alignment 71

4.3.1 Spatial Anchor Alignment 71

4.3.2 The *Time Series* Method: Marker Based World Alignment 72

4.4 Conclusion 78

5 USER COMPLIANCE TO VISUALISATIONS 80

5.1 Introduction 80

5.2 Experimental Setup 81

5.3 Results 82

5.3.1 Speed Regulation 84

5.3.2 Orientation Accuracy 84

5.4 Conclusion 85

6 ONLINE PLANNING IN A REDUCED DEGREE OF FREEDOM HAND-HELD ROBOT 87
6.1 Introduction 87
6.2 Method 88
6.2.1 Candidate Path Selection 89
6.2.2 Calculating Required Paint Density 94
6.2.3 Building Graph Structure 95
6.2.4 Calculating Benefit of a Sub-movement 96
6.2.5 Solving for Best Path 97
6.2.6 Simulation of the Selected Path 98
6.3 Validation and Comparison 99
6.3.1 Experiments in Simulation 99
6.3.2 Experiment with Hardware 101
6.4 Conclusion 103
7 TESTING ASSISTANCE IN REDUCED DEGREE OF FREEDOM ROBOTS 106
7.1 Introduction 106
7.2 Using Augmented Reality to Close Action Perception Loop 107
7.2.1 Proposed Visualisation 109
7.3 Contrasting Different Levels of Robotic Assistance in Spraying Task 109
7.3.1 Task Outline 110
7.3.2 Results 111
7.4 Conclusion 116
8 LOCALLY CAPABLE HAND-HELD ROBOTS 119
8.1 Introduction 119
8.2 Strategy and Tactical Planning and Who Controls it 120
8.2.1 Path Selection Framework 120
8.2.2 Octree Nearest Neighbour Search 123

8.2.3 A Method Where the User Generates the Tactical Plan 123
8.2.4 A Method Where the Robot Generates the Tactical Plan 125
8.3 Visualising the Task Using Augmented Reality 128
8.3.1 Dense Vector Line Visualisation 129
8.3.2 Sparse Heatmap Visualisation 129
8.3.3 Secondary Visualisation: Reachable Area, Orientation and Depth Perception 131
8.4 Experimentation 133
8.4.1 Experimental Design 133
8.4.2 Results 135
8.5 Conclusion 138
9 CONCLUSIONS AND ANALYSIS 143
9.1 Limitations 144
9.2 Future Work 146
9.3 Epilogue 147
AUTHOR'S PUBLICATIONS 148
BIBLIOGRAPHY 150
A CONSTRUCTION OF AN INKJET HAND-HELD ROBOT 157
A.1 System Details 157
A.2 Inkjet Robot Usage and Conclusions 158
B QUESTIONNAIRES 161
C SYSTEM SCHEMATICS 167

LIST OF FIGURES

Figure 1.1 System overview 20

Figure 1.2 An overview of the hardware constructed for the thesis. 21

Figure 1.3 Shows a high level conceptualisation of the hand-held robotic systems discussed in this thesis, shows the flow of information. 23

Figure 2.1 This grid summarises some of the robotic systems from the literature, parameterising the Degree of Freedom (DoF) that the robot controls and the required task DoF. 31

Figure 3.1 A grid of the different versions of the single axis hand-held robot. These are used in the *reduced degree of freedom* experiments in Chapter 6 and 7. 48

Figure 3.2 The typical data flow for version 1 of the single axis hand-held robot 49

Figure 3.3 The typical data flow for version 2 and 3 of the single axis hand-held robot system 51

Figure 3.4 A photo of the 5 degree of freedom robot. 54

Figure 3.5 A high level view of how the end effector is positioned. Showing how the user and robot collaborate. 54

Figure 3.6 A simplified diagram of the kinematic structure of the 5 degree of freedom robot. 55

Figure 3.7 Schematic and photographic view of the delta linkage constructed from a single piece of *Hylite*. 60

Figure 4.1 Overview of the frames of reference for the *time series method* of augmented reality alignment. 74

Figure 4.2 A Summary of the transforms involved in the online method proposed for aligning the Augmented Reality (AR) headset with the motion capture coordinate system. 75

Figure 4.3 The error between the measured location of a motion tracked marker and intended position against length of calibration series. 79

Figure 5.1 Shows a 3rd person view of a user performing a trajectory using the detailed version of the visualisation. 81

Figure 5.2 Shows both the detailed and basic visualisations used in the user compliance experiments. 83

Figure 6.1 A visualisation of the grid structure that the online planning for the *reduced degree of freedom* robot uses. 93

Figure 6.2 A flowchart of the *reduced degree of freedom* algorithm. 94

Figure 6.3 A visualisation of the graph structure used in the *reduced degree of freedom* planning algorithm. 96

Figure 6.4 A graph showing the rank of the path picked over 32 trails for the proposed path planning algorithm and a greedy algorithm. 100

Figure 6.5 The experimental setup for testing the *reduced degree of freedom* algorithm. 102

Figure 6.6 Image showing the various components of the single axis robot used in the experiment. 103

Figure 6.7 Images showing the results of the path planning algorithm when spraying on a target object. 105

Figure 7.1 The experimental setup used in Chapter 7. 108

Figure 7.2 The colour scheme for a region that requires spraying, shown to the user over AR 110
Figure 7.3 The three patterns the users were asked to spray in each mode. 111
Figure 7.4 The combined TLX scores by mode, for the assistance level experiment. 113
Figure 7.5 The factor weighting by mode, for the assistance level experiment 114
Figure 7.6 The time to complete the spraying task, for the assistance level experiment. 115
Figure 7.7 The Mean Squared Error (MSE) per pixel for the assistance level experiment. 117
Figure 7.8 A typical example of the 3 assistance level modes used on a given target. 118
Figure 8.1 A diagram of the octree structure used for storing and looking up line segments for the five axis robot. 122
Figure 8.2 A diagram illustrating how line segments are chosen when the user is in control of the *tactical* plan. 126
Figure 8.3 A diagram illustrating how line segments are chosen when the robot is managing the *tactical* plan. 127
Figure 8.4 A set of images showing the vector line visualisation. 130
Figure 8.5 The heatmap visualisation used in the experiments presented in Chapter 8 131
Figure 8.6 The secondary visualisation used in the experiments presented in Chapter 8 132
Figure 8.7 A boxplot showing the finish times of users for each of the different modes for the experiment presented in Chapter 8 140

Figure 8.8 A boxplot of the total NASA TLX score given by the users for each of the different modes for the experiment presented in Chapter 8 141
Figure 8.9 A bar chart showing the individual factors of the NASA TLX survey for each of the four combinations of modes for the experiment presented in Chapter 8 142
Figure A.1 The custom LED array used for debugging the XAAR 128 interface. 159
Figure A.2 An diagonal view of the inkjet hand-held robot. 159
Figure A.3 A side view of the inkjet hand-held robot. 160
Figure B.1 Page 1 of the questionnaire used in Chapter 8 162
Figure B.2 Page 2 of the questionnaire used in Chapter 8 163
Figure B.3 Page 1 of the form given to users for experiments in Chapter 7 164
Figure B.4 Page 2 of the form given to users for experiments in Chapter 7 165
Figure C.1 High level electrical schematic of the 5 axis hand-held robot. 167
Figure C.2 High level electrical schematic of the Inkjet based hand-held robot. 168
Figure C.3 High level electrical schematic of the single axis hand-held robot. 168
Figure C.4 The electrical schematics for the Inkjet hand-held robot. 169
Figure C.5 The electrical schematics for the 35v power supply used for the Inkjet hand-held robot. 170
Figure C.6 The electrical schematics for the shift register visual debugger for the Inkjet robot. 170

Figure C.7 The sub-schematic for eight lanes of the shift register driven Light Emitting Diode (LED) debugging tool. 171

LIST OF TABLES

Table 2.1 A table summarising some of the properties of various tracking methods that could be suitable for handheld robotics 41

Table 3.1 A summary of the robots that were constructed and their strengths, weaknesses and ideal use cases. 46

Table 3.2 A summary of the error of the proposed time correction method for the Error State Kalman Filter (ESKF) compared to not correcting for the time error. 66

Table 5.1 A summary of the metrics analysed for the user compliance experiment. 84

Table 6.1 Comparison or run time between the proposed path planning method and a simple greedy algorithm. 101

Table 7.1 A summary of the independent sample t-test results comparing both Fully and Semi Automatic modes to the Manual Mode for the assistance level experiment. 116

Table 8.1 A table showing the mean and standard deviation for both an expert and user trials accross the 4 tested modes for the experiment presented in Chapter 8. 136

Table 8.2 A table showing the survey results regarding the visualisations used in Chapter 6 137

Table 8.3 A table showing the survey results regarding the *tactical* planning methods used in Chapter 6 138

ACRONYMS

IR	Infra-Red
GPU	Graphics Processing Unit
ROS	Robot Operating System
SLAM	Simultaneous Localisation and Mapping
LED	Light Emitting Diode
BLDC	Brushless Direct Current
PMSM	Permanent Magnet Synchronous Machine
IMU	Inertial Measurement Unit
UWB	Ultra-Wide Band
CAD	Computer Aided Design
DoF	Degree of Freedom
API	Application Programming Interface
ESKF	Error State Kalman Filter
EKF	Extended Kalman Filter
AR	Augmented Reality
VR	Virtual Reality
SVD	Singular Value Decomposition
SHER	Steady Hand Eye Robot
ASAP	Apparatus to Sense Accuracy of Position
PSD	Position Sensitive Device
USB	Universal Serial Bus
PCB	Printed Circuit Board
IIR	Infinite Impulse Response
FIR	Finite Impulse Response

1

INTRODUCTION

Robots are typically thought of as either stationary devices with articulation, such as robotic arms used to paint car bodies; or as mobile systems such as humanoids or Mars rovers. In contrast this work is contributing to a set of robots that falls between these categories; the hand-held robot. A hand-held robot is defined in this work as:

> Hand-Held Robot: *An agentive device, purposed to act in the physical world in order to accomplish a task. The device should be mobile through being moved and supported by a human, and unsupported by the ground. The device may have additional actuation for moving an end effector.*

Hand-held robots do not need locomotion systems that come with additional actuators, weight and expense. Instead they rely on the human user for locomotion, as shown in Figure 1.1. Hand-held robots have niche applications in a number of interesting areas, such as assisting eye surgeons achieve super-human steadiness and force control without having a cumbersome fixed robot in the work space, (Gonenc et al., 2016).

The advantages of hand-held robots come at a cost; these systems inherently share control between the robot and the human user. Formulating the approach to the problem (planning) and moving the end effector are both shared between the user and the robot. If the user does not know what is required of them to complete the task, the completion of the task would be impossible. As such there must be a joint understanding between the robotic system and the human user about how to approach the task. Further, even if the plan is fully understood by both agents, the success of the task also relies on the adequate implementation of the plan by both agents.

The flow of information between the robot and the user will be facilitated through the use of an AR headset. This technology allows the user to see computer generated graphics overlaid on top of the scene. This allows the systems presented in this thesis to provide task specific information that is aligned with real world objects in the user's vision.

This thesis focuses on hand-held robots that are free to move in a full 6 DoF way, and target tasks that require 5 DoF positioning, however the conclusions and discussions found within the thesis can be extended to either higher or

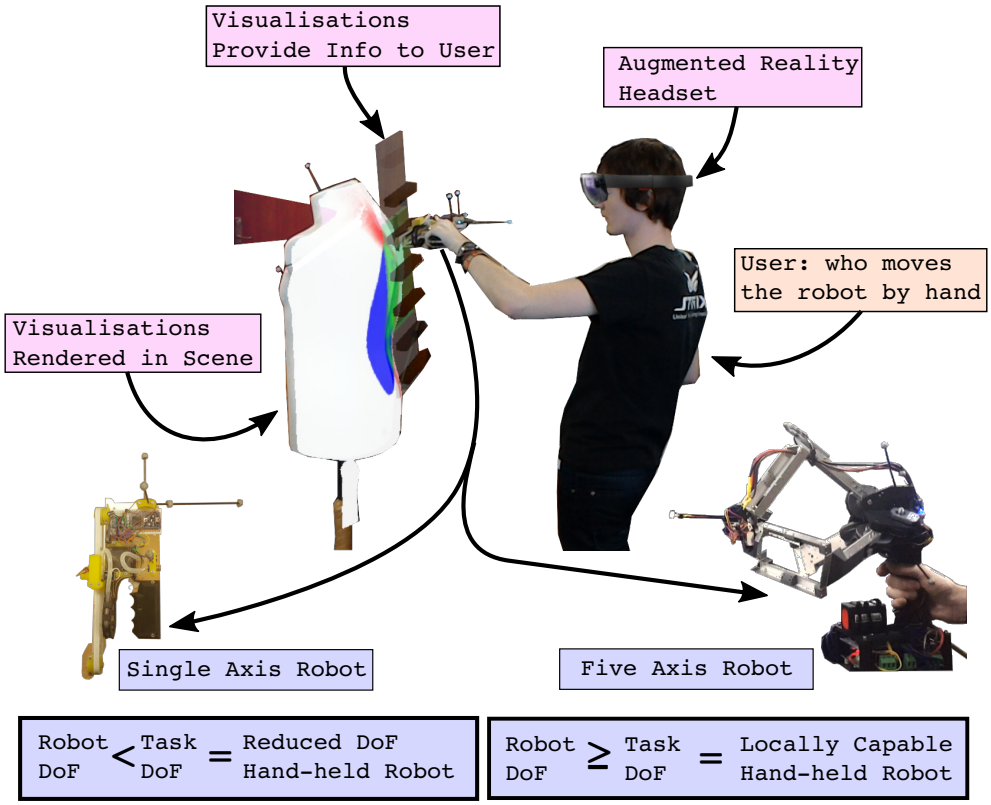


Figure 1.1: This figure shows the typical features of the systems presented in this thesis. The user holds the robot and manoeuvres it around the task. They are provided additional information regarding the progress of the task, instructions on how to proceed, or visual aids via an AR headset. Also shown are two original examples of hand-held robots, one is considered a *reduced degree of freedom* robot, the other is locally capable. This is true given that the task requires 5 DoF to complete.

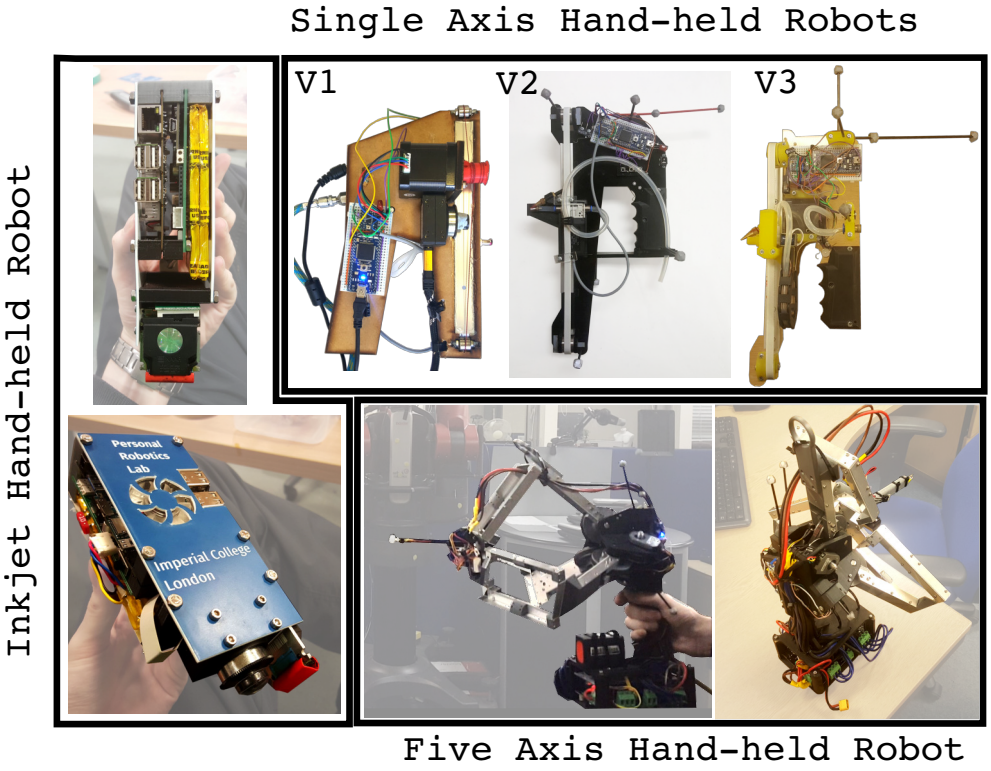


Figure 1.2: This image provides an overview of the hardware that is built to pursue the research questions in this thesis. The top series of robots have a single axis and are used to investigate the properties of *reduced degree of freedom* hand-held robots in Chapter 6 and 7. At the bottom is two views of the five axis hand-held robot used in Chapter 8 to investigate the properties of *locally capable* hand-held robots. Finally, on the left is an Ink-jet hand-held robot that is a *reduced degree of freedom* hand-held robot, that is discussed in Appendix A.

lower DoF tasks and user input. An overview of a system typical of the ones discussed in later chapters is shown in Figure 1.1.

The robots designed and tested in this these are intended to be used in a one handed manner, standardising this allows for some comparison between the robots. Further the robots that were presented by Gregg-Smith and Mayol-Cuevas (2015) and Gregg-Smith (2016) use exclusively a dual handed grip, therefore the work presented here extends the range of form factors analysed in the scientific literature. The robots used in this thesis are show in Figure 1.2.

1.1 REDUCED DEGREE OF FREEDOM AND LOCALLY CAPABLE HAND-HELD ROBOTS

The terms *Reduced degree of freedom* and *locally capable* are defined in this thesis to refer to types of hand-held robots. The definitions used are as follows:

> *Reduced degree of freedom* hand-held robot:
> *This is a robot that has less degrees of freedom than the task requires. This necessarily means that user motion of the base of the robot will couple to the end effector.*

> *Locally capable* hand-held robot:
> *This is a robot that has as many or more degrees of freedom than the task requires. The robot can completely decouple user motion from the end effector, if the desired location of the end effector remains within the reachable area.*

Note that these concepts are task dependent. The single Degree of Freedom robot used in Chapter 6 would be *locally capable* if the task only required the end effector to track a single DoF. For example "end effector height from the ground". The user is only required to have the robot in the vicinity of the correct height, and the robot can decouple the user's motion in this DoF (as long as its axis is not orthogonal to this direction).

Equivalently the five DoF robot used in Chapter 8 could be considered a *reduced degree of freedom* robot if the task was instead a 6 DoF one, as the robot would not be able to decouple the last DoF.

Whether a robot is *locally capable* or a *reduced degree of freedom* robot has strong implications on what kinds of algorithms are applicable, as well as the performance guarantees of the system. A *reduced degree of freedom* robot must share end effector motion with the user at all times, and therefore

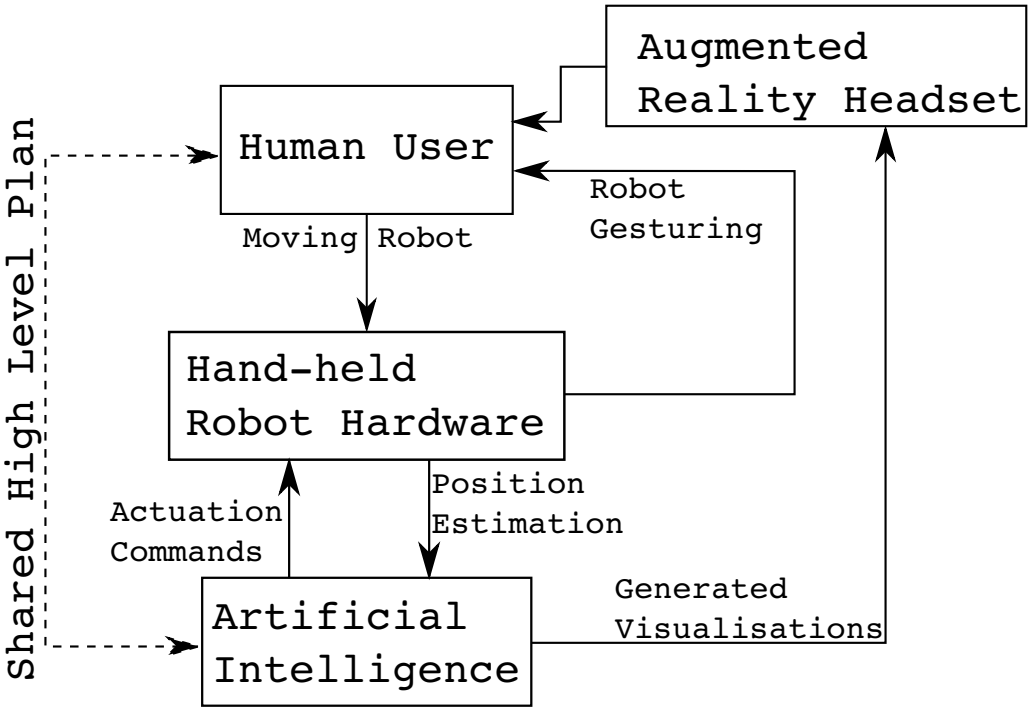


Figure 1.3: This figure shows a high level conceptualisation of the hand-held robotic systems discussed in this thesis. The embodied hand-held robot fundamentally has two agents controlling it, the human user and the automated system. The user has exclusive control of the position of the base of the robot, and the user and the robot share control of the end effector. Feedback to the user is presented via two methods, AR, such as through the Microsoft Hololens, and robotic gesturing. Robotic gesturing is the action of conveying information to the user based on the movement of the end effector itself. Finally there can be a shared high level plan, though there is no guarantee that the user will provide a clear plan that is interpretable by the robotic system, or that the user will fully understand the plan provided by the robotic system.

the accuracy in the un-actuated DoF are bounded by the user in those DoF. Measurements of user accuracy are presented in Chapter 5. *Locally capable* robots however can decouple user motion from the end effector, and as such the accuracy is bounded by the tracking of the robot and the performance of the actuation on the robot. Though it should be noted that this decoupling can only be performed locally; actions from the user that are larger than the reachable area of the robot also must move the end effector. To track a specific trajectory the user must comply with the plan by moving the base such that the end effector can always reach the current set point. Hence, despite local decoupling, the user and the robot are still in a shared control system.

A diagram of the flow of information in the shared control system is shown in Figure 1.3. The block labelled "Artificial Intelligence" is representing the abstract agency of the robot, and is not a separate entity. Not all connections shown in Figure 1.3 are present in all the systems presented and tested in this thesis. The most complete system, described in Chapter 8 does contain all elements shown.

1.2 RESEARCH QUESTIONS

The information in this thesis details an attempt to better understand the following questions, and aims to help future designers answer these questions for their own applications.

Which agent should have ownership of making the plan?

It is shown in Chapter 7 that if the ownership of making the *tactical* plan (which sub element to tackle next), is ambiguous, or if a robot interferes with the user's plan, then users will experience higher task load. Making the allocation of *tactical* planning more explicit can reduce task load. Explicit allocation is investigated in Chapter 8.

What form should the robot take to provide adequate assistance?

Two major categories are explored in this thesis: *locally capable* and *reduced degree of freedom* hand-held robots. The development of original robots in both categories is presented in Chapter 3. These systems form the basis for experimentation in all other chapters. It is found that *reduced degree of freedom* robots are compatible with tasks that do not require the tracking of particular trajectories at the end effector. Online and dynamic path planning approaches such as those presented in Chapter 6 are necessary. *Locally capable* robots on the other hand, as tested in Chapter 8 can track specific trajectories, and are required if the *tactical* plan is going to be controlled exclusively by the robot. The *operational* control of the end effector can also be fully in the domain of the robot.

What information is necessary to facilitate the collaboration?

The user of a *locally capable* robot does not need to know the low level details of the task, as the robot can act on those independently. Does removing this low level information hinder users of hand-held robots for a trajectory tracing task? Chapter 8 details experiments indicating that user comfort is

greatly increased when they can see the underlying actions of the system. If the robot is in control of planning, performance is not hindered much when low level detail is removed, though performance is hindered if planning is left to the human.

1.3 CONTRIBUTIONS

When investigating the aforementioned research questions the following contributions were made:

- Designed and built both a *locally capable* and a *reduced degree of freedom* robot for the completion of a 5 DoF task. These robots were then used in experiments to determine the properties of such systems when paired with a user.
  - Implemented a first of its kind single piece delta robot mechanism. Requiring only one fabrication process, it can be cheaply reproduced with little manual assembly.
- Designed and evaluated a real time path planning scheme for a *reduced degree of freedom* hand-held robot for use in a collaborative spraying task.
- Determined via a user study that providing greater assistance to the user increases task performance, however for a sub-set of people the assistance also resulted in increased task load.
- Calculated an estimate of the bounds of accuracy that can be expected from a user complying with an augmented reality display, and provided recommended specifications for designing a robot to assist with this error.
- Demonstrated that whether the human or robot has control of the *tactical* plan has little effect on task performance. One exception is the case where the human has sufficient, but incomplete information regarding the low level task; in this case the robot having control of the tactical plan aids performance.
- Demonstrated two methods of aligning a AR display with the coordinate system of the robot. The first method is simple to implement, though requires care on behalf of the user. The second method does not require care on the part of the user, though requires additional markers to be attached to the AR headset.

1.4 OUTLINE OF THE THESIS

This thesis is comprised of 9 chapters and 1 appendix. To ensure that the purpose of the chapters is clear they are summarised here:

- **Chapter 1** introduces the core ideas of the thesis and provide motivation for the topics that will be covered.
- **Chapter 2** provides some background for the ideas pursued in this thesis, based on works by others. A brief overview of terminology, existing hand-held robotic systems, AR and Virtual Reality (VR) instruction systems, path planning for spraying applications and six DoF robot tracking is given.
- **Chapter 3** then details the implementation details of the robots that are used throughout the thesis. These consist of a single DoF robot to be used as a *reduced degree of freedom* hand-held robot and a five DoF robot to be used as a *locally capable* hand-held robot. For the latter a innovative single piece composite delta linkage is presented.
- **Chapter 4** details two methods of aligning an AR headset with an external coordinate system. These are used in Chapters 5, 7 and 8 in the context of user studies.
- **Chapter 5** details a user study that was designed to measure how accurately users could follow visualisations shown to them through an AR headset with a hand-held robot.
- **Chapter 6** presents the design and verification of an online method for path planning for a single axis *reduced degree of freedom* hand-held spraying robot. This algorithm is shown to be efficient enough to run in real time via demonstration on a hand-held robot. It also chooses high quality paths.
- **Chapter 7** details a user study that was conducted to analyse how the algorithm designed in Chapter 6 compares to simpler assistance paradigms. It is shown that the more active assistance methods did indeed increase average performance by some measures. However the group of participants became divided on the task load metric. Participants indicated that this was due to conflict at the *tactical* level specifically. This motivated the next chapter, testing more explicit allocation of the *tactical* planning.
- **Chapter 8** details the software design decisions required to allocate the *tactical* plan to either the user or the robot, when using a *locally*

capable hand-held robot. A user study is conducted to investigate how the allocation of the *tactical* plan effects user performance. Further the visualisation detail provided over the AR headset is also investigated, to see how this interacts with the *tactical* plan allocation.

- **Chapter 9** summarises the thesis and the contributions made within it. Limitations of the research are also highlighted, as well as fruitful directions of future research.
- **Appendix A** presents details regarding an Inkjet based hand-held robot. This is a *reduced degree of freedom* robot for liquid application. The robot was functional, though was not used in experiments due to reliability issues regarding the maintenance of the Inkjet head.

2

BACKGROUND WORK

2.1 HAND-HELD ROBOT DEFINITIONS

It is important to define what the key terms in this thesis, and explain how these definitions were synthesised. These definitions will be used to categorise robots in Section 2.3 as well as to help draw distinctions about the level of assistance provided by the robotic system found in the original experiments in Chapters 6 and 8.

First, although the term *"robot"* has a popularly understood meaning, a rigorous definition will aid in making the following discussion more concrete.

> Robot:*"An agentive device in a broad sense, purposed to act in the physical world in order to accomplish one or more tasks."*IEEE Robotics and Automation Society. Standing Committee for Standards Activities. et al. (2015)

This definition conflicts with some of the usages of *"robot"* in common parlance, for example a remote controlled *"battle-bot"* is rejected because no *agentive* aspect to those devices. Similarly some may use the shortened version of the word, *"bot"* to refer to an automated program that acts in a purely digital domain, such as in *"bot-nets"*, this is in conflict with the above definition due to the lack of action in the *"physical world"*.

To refine this definition to include only hand-held robots and not other domains such as *mobile robots*, *wearable robotics*, *humanoids* etc. the following definition is provided.

> Hand-held:*"Designed to be operated while being held in the hand"*
> - Merriam-Webster Dictionary

However in the context of robotics, this definition of *hand-held* is vague, would assisted driving car be a hand-held robot if the user is holding the steering wheel in their hand? Or a surgical robot where the controls are held in the hand? The above definition for *"hand-held"* is refined and combined with that of *"robot"* to form the following definition for *"hand-held robot"*

> Hand-Held Robot: *An agentive device, purposed to act in the physical world in order to accomplish a task. The device should be mobile through being moved and supported by a human, and unsupported by the ground. The device may have additional actuation for moving an end effector.*

In Section 2.3.4 some devices that are very close to the above definition, though fall short in some aspect, are analysed. These examples help clarify the definition by demonstrating its limits.

2.2 SHARED CONTROL DEFINITIONS

As highlighted in Chapter 1, a hand-held robot is necessarily a shared control system. However "shared control" as a concept encompasses many other areas of robotics so it is necessary to define this term and correctly situate it in the context of hand-held robotics.

Abbink et al. (2018) provide a definition for shared control that amalgamates some of the definitions found in earlier works:

> *"In shared control, human(s) and robot(s) are interacting congruently in a perception-action cycle to perform a dynamic task that either the human or the robot could execute individually under ideal circumstances."* - Abbink et al. (2018)

This definition was designed specifically to reject systems where the robot performs control tasks outside the capabilities of a human, for example a high bandwidth fighter aircraft controller. However relevant to us it also excludes the symmetric category, where the human is significantly more capable than the robot. In our case a drawing task could be hypothetically completed by the user alone, though the robot, being hand-held cannot manoeuvre at all without human interaction. This poses a problem, as removing the requirement for locomotion hardware on the robot is the purpose of hand-held robotics as an idea . As such this definition makes it impossible to share control with a hand-held robot in task that require locomotion of the robot body.

An earlier definition by Yanco and Drury (2004) outlines a simpler definition of shared control:

> *"With shared control, the robots are able to do some part of the task and the human operator must do some part of the task."* - Yanco and Drury (2004)

This definition is more liberal in what it includes, essentially only excluding fully autonomous systems and tele-operated or manual systems. For the purposes of our work, we understand shared control to be:

> *In shared control, human(s) and robot(s) are interacting congruently in a perception-action cycle to perform a dynamic task.*

This truncated definition from Abbink et al. (2018), captures the scope of the shared control systems that are presented here.

Further to the idea of *Shared Control*, there is discussion on how to define the specific roles and priorities of the actions that are shared between the robot and the user. Abbink et al. (2018) discuss this using different levels of task execution: strategic, tactical, and operational levels, which were derived from Michon (1985)'s work on driver behaviour. In Chapters 6 and 8 these levels will be used in reference to the spraying and drawing tasks, and how the control is divided between the user and the robot. A brief summary of the terms is presented here:

- ***Strategic:*** A level of planning considering the all of the task immediately at hand. For example "turn right at intersection" for a driving manoeuvre.
- ***Tactical:*** A level of planning that generates the implementation of the next part of the strategic plan. "Come to a stop and indicate", and "Aggressively steer and accelerate" are different implementations which solve the same strategic element with a different tactical plan.
- ***Operational:*** Low level behaviour, such as maintaining the position in a lane, or a particular speed etc. Relates to the low level actuation of the task.

2.3 EXISTING HAND-HELD ROBOTS

There have been few hand-held robotic systems in the scientific literature, especially when compared to other robotic domains. In this section a brief literature review of some key works will be conducted. Some of the robots and devices reviewed have been illustrated in the diagram in Figure 2.1. Note that the *locally capable* robots have at least as many DoF as is required by the task; *reduced degree of freedom* robots have less than required for the task.

Robot DoF \ Task DoF	0	1	2	3	4	5	6
0			Haggar			GS-Tool Prévost[1]	
1						ELS-Ink ELS-Air	
2							LiftWare™ (Steady+Level)
3				Shaper[2]		Micron(Old)	
4			GS-Point4[4]	GS-Pick4[4]			
5						GS-Five[3] ELS-Five	
6						Micron(New)	

Figure 2.1: This grid summarises some of the hand-held robotic systems and devices from the literature, parameterising the DoF that the robot controls and the required task DoF. You can see that five DoF robotic systems are popular, this is due to the fact that many tasks assume that the end effector is rotationally invariant in the axial direction. For compactness *GS* indicates work by Gregg-Smith et al. *ELS* indicates robots presented in this work, *Micron* refers to the work on the Micron project described in Section 2.3.2, *Shaper* refers to the Shaper Origin[1]. Haggar et al. (1983) and Prévost et al. (2016) are self-named. There are some additional notes to consider: [1] The task is a 2D paint distribution task, though 5 DoF position of the robot does effect the paint pattern. [2] This robot is physically constrained to a plane, and is not a hand-held robot by this work's definition, but is notable none the less. [3] This robot has 6 DoF, though there is redundancy, allowing the end effector to only have 5 DoF. [4] This robot does have 4 actuators, thought the DoF are highly inter-dependent, meaning that across its reachable area not all DoF may be independently actuated.

2.3.1 *Gregg-Smith: Cooperative Hand-Held Robots*

Gregg-Smith (2016) published a thesis that is highly related to this one. In that work they present two experiments that are particularly influential on this work.

A simplified painting task was conducted with a 4 DoF robot. Three assistance modes were tested, *"manual" , "semi-automatic", "automatic"*. It was demonstrated that the increasing level of autonomy reduced the amount of movement required from the user, the completion time, and the reported subjective task load. This demonstrates that hand-held robots can be very effective at augmenting tasks that could be performed manually. However a similar task conducted with the same robot showed significantly less well defined results, which indicates that some tasks are more appropriate for hand-held robots than others. A paraphrased quote from a participant in this experiment is illuminating:

> *"The tool won't go where I want it to"* - participant, Gregg-Smith and Mayol-Cuevas (2015)

This implies that there is a conflict in the *tactical* plan between the user and the robot, which was also observed in the experiments presented in Chapter 7, and in experiments inspecting the role of the *tactical* plan, presented in Chapter 8.

A second key experiment was a study of different visualisation methods for aiding the collaboration between the user and the hand-held robot. All visual modes tested performed the same, these were AR, VR and monitor based methods. Gregg-Smith et al. suggest that the visualisations could be of even lesser quality, as long as they are sufficient to get the robot in the vicinity of the task element. The robot used in this experiment was a *locally capable* robot, and therefore the visualisations did not need to impart the state of the task to the user. They also suggest that the *robotic gesturing*, the action of the end effector guiding the user around the task, also aids performance when visual fidelity is low. This suggestion that the visualisation can be of low quality without significantly effecting performance is tested explicitly in Chapter 8.

2.3.2 *Micron: Eye Surgery Hand-Held Robot*

The Micron project, MacLachlan et al. (2012), is a hand-held robot that was designed as a test platform for investigating assistive techniques for eye sur-

geons. It is a set of work that has evolved over many years, and been contributed to by many authors.

The robot's form factor is narrow and pen-like, such that it is analogous to the normal tools that a surgeon would use. This small form factor only allows for a very small accessible area at the end effector, being a 4×4mm cylinder in the most recent implementation. The initial work envelope was even smaller, 560×560×100µm, Ang et al. (2000). This increase in reachable area allowed the assistance modes implemented move from tremor rejection, to task specific and progress aware assistance modes, Becker et al. (2012). This highlights that the accessible area that is required is task dependent, based on the degree of correction necessary to bound the user motion to the ideal case.

Further this project started as a three DoF robot and has been extended to a six DoF robot. Most of the applications demonstrated have been five or six degree of freedom tasks, as such this project represents both *reduced degree of freedom* robots and *locally capable* robots with a variety of use cases.

2.3.3 *Hand-held Liquid Distribution Robots*

Haggar et al. (1983) constructed a hand-held spraying machine to automatically identify areas of vegetation that should be dosed with herbicide. It used the ratio of red and infra-red light reflected from the plant canopies to categorise between unwanted plants and the soil. The machine would activate the herbicide spray when over vegetation. This work demonstrates well the benefits of an intelligent hand-held device for spraying. The machine was low cost, could be deployed to a large area without provisions for a wheeled machine, and reduced the burden on the user compared to manual tools. This machine however does not fit neatly into the category of hand-held robots because the machine does not inhabit and *agentive* role, simply responding to a threshold of one sensory input. Though for the time this was produced it approximates a robotic solution. Such spraying tasks in an agricultural environment are usually now undertaken using wheeled robotic systems as can be seen in the review of variable rate spraying in agriculture by Guan et al. (2015). This is due to the fact the agricultural environment can be highly controlled; hand-held spraying robots are most likely to be useful over wheeled robots when the exact usage pattern of the system cannot be known in advance.

Prévost et al. (2016) presented a hand-held robotic system, where the robot is assisting the human towards the joint goal of painting a mural. In their

design the robot's position is found via external cameras that locate QR codes mounted on the top and bottom of a standard spray paint can. When the robot is judged to be in a good position to add paint to the canvas, a radio controlled servo is actuated to press the valve of the spray paint. The user just has to meander the robot across the canvas, and the paint will be applied such that the state of the painting moves towards that of the target design. This is very similar to the "semi-auto" mode that has been implemented in Chapter 7. Their system provides the user with a graphical representation of areas of the painting that need more paint, and a total possible added value using the current colour of paint. Prévost et al. (2016) simplified the paint distribution by representing it as a symmetric Gaussian distribution, the same is done in this thesis when necessary.

2.3.4 *Notable Non-Hand-Held Robots and Devices*

The above robots all fit the definition of hand-held robot used in this work to a greater or lesser extent. Though to grasp the boundary of the hand-held robot domain it is useful to observe some notable examples of robots that narrowly miss the definition that is presented in Section 2.1.

The Shaper Origin™[2], is advertised as *"the world's first hand-held CNC machine"*. It consists of a router, a tool for cutting away material with a downward facing spinning cutter, that is able to move in three directions. Above the cutter is a screen that shows the user the progress of the cut and guides them where to move the machine next. The entire base one which the router is mounted is free to slide on top of the work piece, making the tool mobile, unlike a traditional CNC cutter. To locate the robot the user adheres some specially patterned tape to the work piece and forward facing cameras on the machine locate these markers and construct a coordinate system around them. To complete a cut, the user loads a file describing the cut to the robot, and the robot directs them to push it around the work piece, actively correcting the position of the router head as needed. This process can produce a cutout on the work piece that is effectively unbounded in size.

The Shaper Origin™ product is very close to the definition of a handheld robot used in this thesis, having all elements of the definition with the exception of *"...supported by a human, and unsupported by the ground."*. This aspect changes the interaction between the robot and the user significantly, as at any point the user can release their hold on the robot and the dynamics of the system become stationary. Further this work piece supported struc-

2 https://www.shapertools.com/

ture removes the weight and other fatigue related issues that are related to holding a robot free-hand.

Another example of a robot that highlights the edge of the hand-held robot domain is the Steady Hand Eye Robot (SHER), Uneri et al. (2010). This system is designed for eye surgeons, aiding them in many of the same ways that were discussed in the Micron project in Section 2.3.2. However the SHER system has the end effector held by both the user and the robot. The user and the robot jointly move the end effector, this is achieved via force/torque sensors in the handle of the end effector. Again, this system fits most of the definition, though the robot and end effector is not supported exclusively by the human user. This is even more significant in the case of the SHER as compared to the Shaper Origin™ as the kinematics of the robot are also rigidly connected to the ground. This means that the SHER could be programmed to move around the task without participation of the user. For example a position hold behaviour can be implemented trivially by simply locking the actuators, however in an un-supported hand-held robot position hold must require active control. Further the working area of the SHER is defined by the range of motion of the actuators, however in a hand-held system the global work space is theoretically unbounded.

2.3.5 *Comparison and Summary of Existing Hand-held Robots*

It can be seen by the examples given above that there can be quite a large variety in form factor and task scale. For example the global work space for the Micron is a 30mm cube with a robot accessible area of up to a 4mm by 4mm cylinder and a typical task requirement of 10µm. The robots presented by Gregg-Smith however had a global work space in the scale of a room, a reachable area of 0.6 × 0.55 × 0.3m and a task precision of around 5mm (estimated) at the end effector tip. These values differ from each other by more than an order of magnitude.

The complexity of assistance also varies significantly, from simple activation of a spray nozzle in Haggar et al. (1983), to assistance that is aware of the meta-progress of the task and augments end effector based on sensitive measurements, such as with the peeling task using the Micron presented by Wells et al. (2014). Most of the examples use visual feedback to the user to facilitate collaboration between the user and the robot, however in the work presented in Gregg-Smith and Mayol-Cuevas (2016*b*) it is shown that embodied feedback in the form of *robotic gesturing* can be a natural mode of

interaction between the robot and the user, and can be used with visualisations for multi-modal feedback.

The choice of actuators was primarily based on the scale of the robot, the Micron primarily used variations of piezo electric actuators due to there compactness, mid scale robots like that presented by Prévost et al. (2016) used hobby servos and the larger robots presented by Gregg Smith et al used heavy duty robotics servos such as the Dynamixel MX-64T.

Tracking choices appear to be based on both the scale of the task and the required accuracy. For example on a large scale task that required fairly little precision, the marker tracking solution used by Prévost et al. (2016) was acceptable. Though for the space constrained work area of the eye surgery task a high accuracy and compact tracking method was needed that had high bandwidth and low latency to correct the hand tremor in real time. For that situation a purpose made high precision tracking system was proposed.

These considerations are applied to the design of the robots presented in this work in Chapter 3.

2.4 VISUALISATIONS AND INSTRUCTIONS

Zolotas, Elsdon and Demiris (2018)[3] conducted a study attempting to measure whether visualisations shown to the user of a smart wheelchair could help them build a mental model of the assistance. The mental model is described as the users ability to anticipate and predict the assistance that will be provided. A good mental model is anticipated to reduce conflict with the shared control system. The study did not show many positive results, though it did show that users can be hindered by visualisations, and that visualisations that are inconveniently located are almost entirely ineffective.

There is a wealth of work regarding sharing robot trajectory information with users and bystanders (Chadalavada et al. (2015); Walker et al. (2018)). Walker et al. (2018) for example explored a range of techniques to help a bystander understand the future movements of a flying robot. They found that their *Nav Points* method was particularly effective. Further this method appears to be most suited trajectory requests, if a user was part of the loop to implement the trajectory, such as they are in later chapters. This consisted of floating way-points that had the time till arrival displayed above, as well as the time till departure, for the case where robot intends to stay stationary at the way point for some time. Other methods presented offer less detailed information and would be less useful if the user was in control of imple-

3 note this work was contributed to by the present author

menting the path. However, in the use case the authors were discussing, the other methods also aided the user in perceiving the future trajectory of the flying robot.

Wu et al. (2016) demonstrated that augmented reality can be useful in guiding manipulation tasks. They used a monitor to display the assembly area with overlaid graphics giving contextual information on how to manipulate the various parts required to build a children's toy. This can be seen as similar to this work, though the information provided to the user is categorical and not time critical. For example, the command may be to *'Rotate the component!!'*, where there is no need to do this within a particular time frame, and no continuous amount of rotation is indicated. The work presented in Chapter 5 could be seen as an attempt to do similar instructions with continuous and time critical actions.

2.5 PATH PLANNING FOR SPRAYING APPLICATIONS

2.5.1 *Introduction*

Path planning for liquid distribution is an area of great interest for those wishing to optimise the performance of robots used for spraying panels for the auto industry. That application has a significant difference from the one that is presented in this thesis, spraying robots in industry can nearly always recompute the paths. This is because the robots are usually painting identical parts, and as such the same program can be used every time. In the domain of hand-held robots there is significant introduction of uncertainty in the behaviour of the human user. However gaining an understanding of some of the methods used in the auto industry will inform the discussion in Chapter 6 significantly, as a hand-held robot will be tasked with collaboratively completing a spraying task with a robot. The algorithm presented in Chapter 6 leverages a number of concepts that are analysed here.

2.5.2 *Coverage Path Planning*

Galceran et al. Galceran and Carreras (2013) presented a review of algorithms that aim to complete the *Coverage Path Planning* problem (CPP). Spray painting could be conceived as a CPP problem, similar to the aims of a robotic system that aims to scan the entire surface of a 3D structure, such as a building or under water terrain, with its sensors. The review concerns itself with both 2D and 3D problems, though in the latter most of the algorithms are

optimising for coverage of a 2D surface embedded in a 3D workspace, which is the case tackled when spray painting on a 3D surface.

2.5.3 *Car Body Spray Coating*

An even coat of paint is usually the target in car body painting, this is confirmed by Chen et al. (2009) in their review of path planning for spray painting. As such most algorithms in this domain are aiming to minimise the deviation from the preferred paint thickness.

Most of spray painting specific algorithms presented aim to separate a coverage problem into a set of simple problems that can each be solved by a simple *"zig zag"* pattern. Typically these approaches are entirely offline, and at run time the coating is performed open loop, with no checking of the final coat evenness.

Atkar et al. (2005) approaches this problem by considering simple patches, which are those that are diffeomorphic to a disk with no holes, and geodesically convex. With each of these patches they aim to solve the uniform coverage problem, which is the aim to distribute the paint on the surface evenly, while minimising the deviation from the specified dose. This problem is then decomposed further into 3 fairly independent problems, seed path generation, the speed profile along the path and the separation between passes in the generated path. A realistic model for the distribution of paint from the nozzle is used, though the method is independent of the model used to represent the paint distribution.

Hegels *et al.* Hegels et al. (2015) presented a method for modifying existing robotic arm trajectories for painting car panels. The method maximised the evenness of the paint, whilst keeping the accelerations acceptable for the large robotic arms. They specify a method of capturing the real coating distribution by spraying for a short time onto a plate, which is then sampled across its area. They use the sampled distribution in evaluating the cost function of the current iteration of the trajectory, but in order to use optimisation methods that utilise gradients, they fitted this sampled distribution to a distribution that could be described analytically. This simplified distribution is used to calculate the next direction to search in the perimeter space. Their choice to use a simplified model for path planning followed by an expensive method for tracking the cost is one that has been emulated in the experiments presented in Chapter 6.

2.5.4 *Simulation of Paint Density*

Konieczny and Meyer (2009) give an account of their work regarding simulating airbrushes for graphics creation on a computer, this is completed with the design of an electronic airbrush for the user to interact with. Their system tracks the electronic airbrush using a magnetic tracker, and has a dual action trigger (allows control of paint flow and air flow). They also present a mature algorithm for simulating the paint droplets, including the methods used to blend colours for a realistic looking finish. They implemented all of the computationally expensive operations on a Graphics Processing Unit (GPU).

The droplet simulation used in this work is based on some elements of this work, particularly the ray casting approach, that allows for an arbitrary spray distribution. Also the use of a texture map to store the resultant paint distribution, and allow mathematics to be accelerated using the GPU.

2.5.5 *Non-Uniform Spraying*

Whilst uniform coverage is a common goal in car body spray coating, it can be seen as a limitation when considering more diverse spray applications, such as spray painting for artistic pursuits, or applying medicine to specific patches of skin.

Seriani et al. (2015) produced a system that frees the path planner from the requirement of uniform dosing. They can prescribe a target grey scale image, which the robot will then aim to produce. This is achieved by generating trajectories that are carried out at different distances from the work piece, changing the effective size of the paint distribution when it reaches the work piece. Spraying from a distance produces a wide spray pattern, spraying from a shorter distance creates a thin spray pattern. This way the robot can spray from a larger distance initially, then move closer to add detail. The trajectories for each layer are highly curved, and as such not appropriate for a user to emulate manually. A user could plan a path in the same manner using a hand-held robot, changing the "resolution" of the spraying by changing the distance to the target surface.

2.6 TRACKING ROBOTS IN 6 DEGREES OF FREEDOM

When a robot is held in the hand and unsupported by the ground, as specified in the definition of *"hand-held"* robot given in Section 2.1, it necessarily

is capable of moving in 6 DoF. Therefore, in order to complete most tasks, a robot will need to be tracked in 6 DoF also. However, methods that track in lower DoF can still be useful with the addition of sensor fusion, as detailed in Section 2.6.1

As the robot is not in contact with the ground and it does not command it's own movement, the two simplest methods of robot tracking are not feasible, dead reckoning and odometry, which are typical for ground robots. There are however many options for solving this problem, some of the more relevant ones are listed in Table 2.1.

There are a few key properties to consider. The first are requirements that need to be met that are external to the robot. These are usually in the form of base stations, or other sensors viewing the tracking area. An example would be the cameras that view the tracking area in a Vicon™ system. If the application of the robot requires the system to be moved to new environments regularly, the re-installation and calibration of such cameras could be a large limitation. On the other hand, a robot that uses Simultaneous Localisation and Mapping (SLAM) as its primary tracking method, can be redeployed to a new area with little overhead.

2.6.1 *Sensor Fusion*

The above methods all have their own advantages and disadvantages, though the combination of two or more of these systems can produce a system that is better than the sum of its parts. The dominant paradigm for combining such measurements is the Kalman filter (Kalman, 1960). A Kalman filter is an iterative algorithm that maintains a state estimate and the error covariance, the latter of which is comparable to the current uncertainty in the state variables. To do so it requires measurements of some variables that render at least some part of the state observable, and a value for the covariance of the measurement noise and process noise. Measurement noise is the additive noise that corrupts the measurement signal, and the process noise is the noise that perturbs the state directly.

The five DoF hand-held robot that will be described in Chapter 3, will make extensive use of an Error State Kalman Filter (ESKF). Therefore, a fairly complete description will be presented here.

The simple Kalman filter is designed for linear systems and makes an optimal estimate of the state of a system, accounting for the uncertainty of the measurements and providing a confidence of the estimate. It is optimal in the sense that the state estimate would return the lowest possible mean

Tracking Method	DoF	External Requirements	Update Rate (Typ)	Comments	Useful Reference
Retro-reflective Motion Capture (Vicon™)	3/6	Usually 4+ cameras	100-200Hz	Often considered a reference baseline, accuracy < 0.5mm	Merriaux et al. (2017)
QR Code Tracking	6	1+ cameras	30-60Hz	Low cost, Accuracy 1cm	Garrido-Jurado et al. (2014)
IMU Integration	6	No external	1000-3000Hz	Unbounded drift	Kok et al. (2017)
Magnetic Tracking	6	1+ transmitter	100-200Hz	Distortion if near conductive objects	MacLachlan et al. (2017)
SLAM	6	No external	60-90Hz	Can suffer drift	Davison et al. (2007)
Apparatus to Sense Accuracy of Position (ASAP)	3/6	2 Position Sensitive Device (PSD) sensors	1000Hz	Needs active markers, custom implementation	MacLachlan et al. (2009)

Table 2.1: A table summarising some of the properties of various tracking methods. These numbers are only estimates, as most implementation are not operating close to a physical limitation, hence some effort can increase the update rate or the accuracy, often in complicated trade offs with cost and other limitations. When listing DoF of the sensor method, notice some entries have 3/6 listed, this is due to the fact these methods are tracking points rather than 6 DoF poses directly, however these sytems are used to track constellations of points allowing the construction of 6 DoF poses. UWB does not usually do this due to the size of the constellation required.

square error versus the ground truth. The uncertainty is also optimal, in the sense that it will be minimal. The prior optimality statements are under a range of assumptions, such as a perfect model of the system, whiteness of the process and measurement noise etc.

However for non-linear systems, such as one that requires modelling of robot rotations, a simple Kalman filter is not appropriate. A common modification is the Extended Kalman Filter (EKF), which was presented by Uhlmann (1992). This method linearises the system around the current estimate of the state. For the sake of brevity, a complete description of the EKF is omitted. However, a variation called an ESKF (Roumeliotis et al., 1999) will be used in Chapter 3.

2.6.1.1 *Error State Kalman Filtered Sensor Fusion*

The ESKF Roumeliotis et al. (1999, n.d.) provides a number of benefits over other Kalman filter implementations, which are described at length by Mady-

astha et al. (2011). The primary benefits for the applications presented in this thesis are highlighted:

- The error state system uses a minimal representation for orientation.
- The error state system is always close to the origin, helping avoid singularities.
- Updates to the nominal system can be performed at a different rate to the updates to the error state system.

Sola (2017) provides an excellent tutorial on the application of an ESKF to an Inertial Measurement Unit (IMU) based navigation system. For the benefit of the reader, and for clarity regarding modifications to the ESKF made in Chapter 3, the method will be briefly detailed here. Further, notation will copied from Sola (2017) when discussing the ESKF in this work.

The ESKF has 3 state vectors, the true state, the nominal state and the error state. The true state can be calculated at any time by combining the nominal state and the error state. The nominal state represents a non-linear system, tasked with tracking the large signal of the state. The error state is a linear model that is estimating the error in the nominal system, utilising knowledge of the signal noise and perturbations.

When a observation is made that renders the error observable, the error in the nominal state can be corrected for by injecting the mean of the error state into the nominal state. After this injection stage the error state system is reset, meaning that in normal operation its state is always near the origin.

In practice this means that IMU observations are being integrated into the nominal state over time. Then when there is some measurement that provides some absolute reference, in our case a motion capture system, the drift that would be gathered in an IMU only system can be accounted for.

2.6.1.2 *Correcting Time Misalignment*

The above ESKF relies on the fact that incoming measurements are a representative of the current state of the system, and with no delay (with added noise). However, there could be significant delays in acquiring the measurements from a motion capture system. During periods of large acceleration, the difference between the IMU based error state system and the absolute observations, that are supposed to render the drift in the error state system observable, are large. This will either result in an increased error in the

predicted state of the robot, an increased estimate of the uncertainty, or erroneous changes to the accelerometer and gyro drift state variables. Hence some method must be used to account for the time misalignment.

Larsen et al. (1998) describes a method that is based on extrapolating the out of date measurement up to the current time. This method maintains the optimality properties of the Kalman filter, as described at the beginning of Section 2.6.1. It requires that the observation time is known such that the appropriate correction factor can be integrated from the observation time to the arrival time of the data. This condition is available in some optical motion tracking systems, as the sampling is at regular intervals, even though the latency is variable based on network conditions.

The method requires maintenance of a matrix product as defined below, where N is the number of updates that happen since the estimated time that the absolute measurement was taken, and k is the current time step index.

$$\mathbf{M}_* = \prod_{i=0}^{N-1} \mathbf{F}_{k-i-1} \tag{2.1}$$

This new matrix can then be used in the Kalman gain update and in the covariance update.

2.6.2 *Augmented Reality Calibration*

Tuceryan et al. Tuceryan et al. (2002) proposed a system of calibration called 'Single Point Active Alignment Method' or SPAAM. In their system the position of the augmented reality headset is reported by a magnetic tracking system, the offset between the magnetic marker affixed to the headset and the optical centre of each eye is not known an is to be calibrated. The offset between the magnetic tracking system and the world coordinate system is analogous to the Hololens world frame and the world coordinate system in this work, Section 4.3. The calibration of the tracker system was described in their previous work Tuceryan et al. (1995), they used a pointer object that was tracked using the same magnetic system to index known locations in the world coordinate system. The offset between the eye and the headset marker was found by having the user align a single known location in the environment up with a cursor that is displayed through the headset.

Tuceryat et al's approach seems to be simple and effective, though is not directly applicable to the hardware used in this project. The primary problem is that the high frequency and low latency measurement of the Hololens

is calculated internally, and there is no way to have a secondary object in the same frame used for indexing the world coordinate frame. In their work the headset and the wand were tracked with the same system, allowing them to match real world points with hologram locations directly. Also there is no attempt at recalculating any of the offsets in an online manner, due to the magnetic tracker's coordinate system not shifting over time, however the Hololens does shift it's coordinate system due to adjustments in the map used for the visual odometry.

3

HAND-HELD ROBOTS AND THEIR IMPLEMENTATIONS.

3.1 INTRODUCTION

Whilst there are some commercially available actuated hand-held devices, such as the Shaper Origin[1] and Liftware[2], they are not suitable for modification to an experimental platform. Existing hand-held robots are either specialised to a particular domain, such as the Micron for eye surgery (MacLachlan et al. (2012)), or are one off experimental platforms for the pursuit of quite specific research questions, such as those produced by Gregg-Smith and Mayol-Cuevas (2016*a*).

Therefore in order to pursue the research questions in this thesis it was necessary to produce custom research platforms. To investigate the boundary between hand-held robots with *reduced degrees of freedom*, and those that are *locally capable*, two robots specified by these constraints have been produced. A robot with a single DoF will be described in Section 3.2; this robot was used for the experiments presented in Chapter 6. A five DoF robot will then be presented in Section 3.3; this robot was used in the experiments in Chapter 8.

A third robot, that uses an Inkjet head as its primary tool is discussed in Appendix A. This robot was not used in any experiments, and hence understanding its design is not critical to the understanding of this work, however some qualitative comparisons to that robot will be made. When considering the central ideas of this work, the Inkjet robot can be considered a smaller version of the one DoF robot presented in Section 3.2. This is because its array of nozzles are arranged in a line, and cannot move, which makes the robot a *reduced degree of freedom robot*.

Table 3.1 highlights some of the differences between the robots that were built for this project. The primary reason for attempting a reduced degree of freedom robot is to lower the weight and cost of the handheld system. The weights listed for each system could be reduced by more rigorous industrial design and access to more sophisticated fabrication techniques. However the magnitude of their weights relative to one another is highly indicative that

1 https://www.shapertools.com/
2 https://www.liftware.com/

reducing DoF and complexity can be a large weight saving. Cost is another factor that cannot be precisely analysed when discussing prototypes, however the increased complexity and number of motion components in the five DoF design ensures that it would be more expensive than the single axis version, even after optimising for cost of production. The Inkjet design uses a long Inkjet array, which is an expensive component, however the rest of the design could be optimised for low cost.

Robot Prototype	Strengths	Weaknesses	Ideal Use Cases	Weight
Inkjet Design	• High Detail • Low weight	• Small coverage area • Small liquid output • Easily blocked	• Small scale liquid application	206g
Single Axis Design	• Low mechanical complexity • Low cost	• Lower Spacial Resolution • Less ability to correct for user error • Usage can be confusing	• Lower detail applications like spray painting. • Where order of liquid application is not important	443g
Five Axis Design	• Complete ability to correct for user error • High speed trajectories are possible	• High complexity • High cost	• Applications where order of paths matter • Where complicated trajectories are needed	1407g

Table 3.1: A summary of the robots that were constructed and their strengths, weaknesses and ideal use cases.

3.2 SINGLE AXIS HAND-HELD SPRAYING ROBOT

In order to experiment with the minimal mechanical design that would still be able to effectively aid a human user, a single axis robot was developed. The robot can also emulate a zero DoF robot by not using its motion axis. A zero DoF robot is one that does not have the ability to move its own end effector, though is able to provide assistance by automating the triggering of the spray head. There were three iterations of this robot design, however

version 2 as shown in Figure 3.1b was not used in any experiments, and was only a development precursor to version 3.

3.2.1 *Version One*

Version one of the single axis hand-held robot was a proof of concept design, which is used in Chapter 8, and can be seen in Figure 3.1a. It was designed to be held in a single hand and have a similar form factor to a power hand drill. A forward facing camera to track markers in the scene was utilised as a tracking method. The ArUco marker system developed by Garrido-Jurado et al. (2014) was used. The markers were arranged into a grid, and the "board" feature of the ArUco library was used. This allows the position of the robot to be estimated if any one of the markers is visible; multiple markers being visible simply increases the quality of the estimation. A summary of the system can be seen in Figure 3.2.

The end effector could slide along a linear guide that had a length of 150mm, and was made with drylin® N guide rail (17mm width). This version used rigid string as a transmission from the motor to the end effector slide. This was to maximise the ability to reconfigure the design. For example the pulley size could be changed by printing a new one, and the simple cylinder needed for a string pulley is reliably printed on any 3D printer.

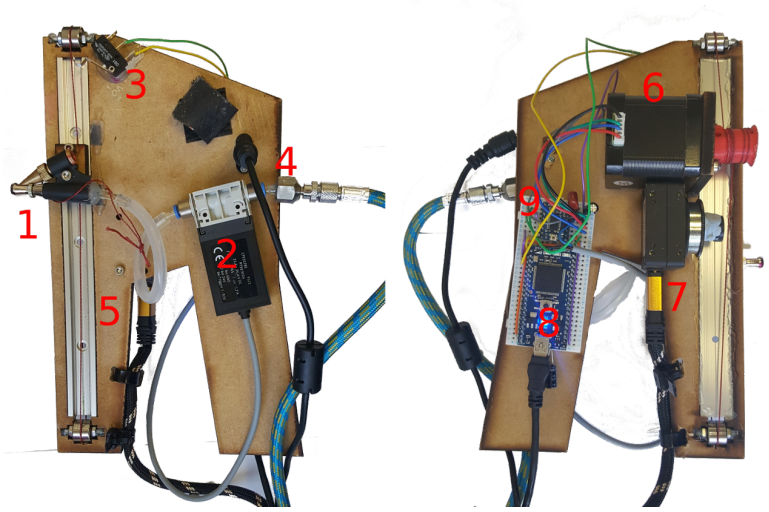
The robot uses a nema 17 stepper motor to drive the linear stage. This choice was made to simplify the design of the electronics and control logic. However this came at the price of increased weight and power usage. This precluded the use of on-board batteries, as run time would not be sufficient. These issues informed the choice to move to Permanent Magnet Synchronous Machine (PMSM) drives in the 5 DoF robot presented in Section 3.3.

On board the only computation is a micro-controller (MBed LPC1768) to control the timing of the stepper motor, polling the user buttons and communicating to the external computer via Universal Serial Bus (USB).

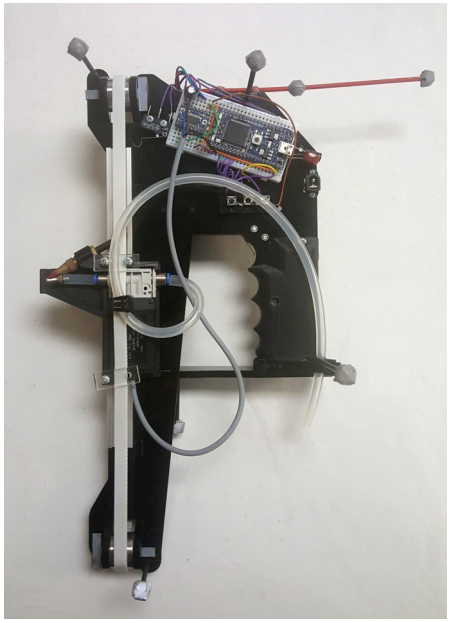
The robot has an airbrush nozzle attached to its end effector. This is supplied with compressed air via a hose. The nozzle can be activated by an in-line solenoid air valve (FESTO MHJ10-S09) mounted to the body of the robot.

3.2.2 *Version Two*

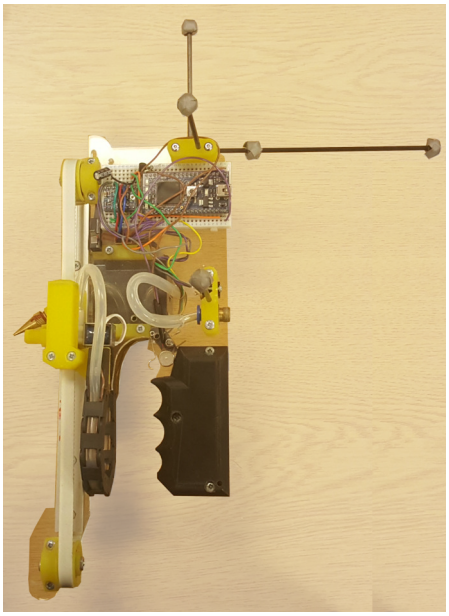
A second version of the design in Section 3.2.1 was implemented which increased the linear stage length to 200mm and improved user comfort



(a) Single Axis Robot V1



(b) Single Axis Robot V2



(c) Single Axis Robot V3

Figure 3.1: The different versions of the single axis hand-held robot. a shows the first version, which used a on board camera for tracking relative to the target. This is the primary robot used the preliminary trial in Chapter 6. c is the newest version, that was used in the experiments in Chapter 7. This uses a motion capture system for tracking, the markers for this system are the silver spheres located near the top of Version 2 and 3. b was an intermediary stage for testing user ergonomics upgrades, as well as moving the spray actuation valve closer to the nozzle to maximise the response sharpness from the airbrush nozzle. Figure a includes red labels that indicate the different components, which for the most part are similar across the three designs. 1: Spray nozzle, 2: Solenoid valve, 3: End stop, 4: Air hose attachment, 5: Linear slide, 6: Stepper motor 7: Camera (V1 only), 8: Micro-controller, 9: Stepper motor driver.

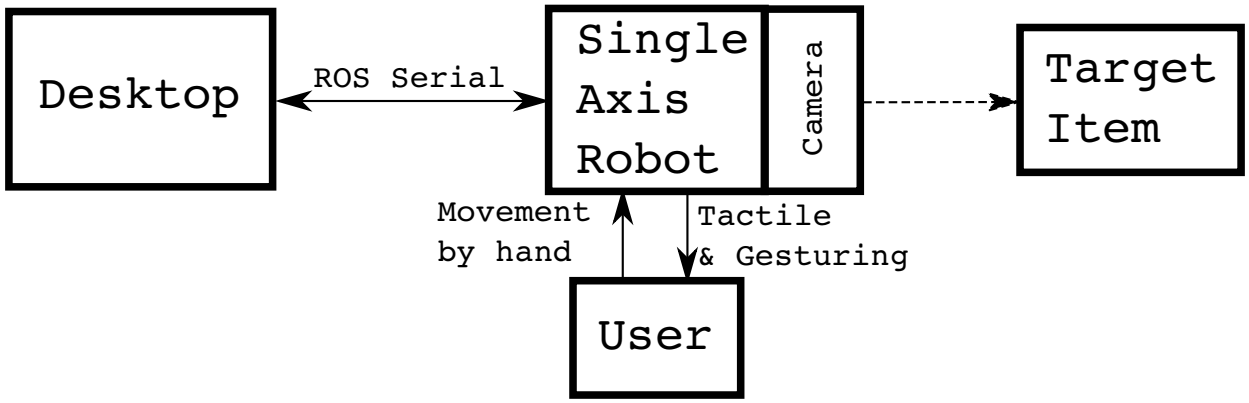


Figure 3.2: This shows the typical data flow for version 1 of the single axis hand-held robot system. The dotted arrows towards the camera block indicate that this is an observation by the camera, rather than a traditional communication channel.

by adding scales to the sides of the handle. Also the tracking system was changed from a two dimensional marker tracking system mounted on the robot, to a retro-reflective marker based motion capture system, with static cameras around the arena.

The tracking system was changed such that the target object could be made larger without risk of obstructing the grid of two dimensional markers. With a fixed camera motion capture system, the cameras can be mounted in such a way that the user and task object do not occlude the hand-held robot under normal usage conditions.

The solenoid valve was also moved onto the moving platform to minimise the volume of air between the valve and the nozzle. This was done to help improve the sharpness of response of the airbrush, by minimising the amount of air that is between the nozzle and the valve.

3.2.3 *Version Three*

The third and final version of this robot is displayed in Figure 3.1c. This robot is similar in form and function to version two, but was made more robust, such that it could work reliably in the user trial presented in Chapter 7.

Robustness was added by ensuring that the air hose for the airbrush nozzle, and wires for the solenoid valve that controls it were neatly routed through a drag chain mounted adjacent to the linear stage. Additionally all of the rods that hold the retro-reflective markers were changed to use carbon fibre rod, rather than wooden dowels, as the latter were not dimensionally stable which lead to poor tracking by the motion capture system. The rigid

string and pulley system was replaced with a T2.5 timing belt, again for additional robustness.

The third version also included a vibration motor adjacent to the trigger. The purpose of this was to add tactile feedback for when the robot was spraying. This was necessary as the majority of the experiments using the robot were simulating the liquid spraying and visualising the result to the user via an AR headset, and as such there was no sound or feeling associated with the actuation of the nozzle. The motor vibrates whenever the nozzle would have been spraying, allowing the user some degree of tactile feedback during use.

A smaller solenoid valve was used to minimise the swept volume of the linear stage, making the robot more compact. Additionally a higher capacity paint chamber was installed that is carried on the linear stage, reducing the number of tubes that need to be carried in the drag chain.

The weight of the final version was 443g. The majority of the weight in this design is the stepper motor used to drive the linear stage. This could be significantly decreased by utilising a closed loop positioning system. The large stepper motor was only needed to ensure that no steps were missed in the worst case situations, such as if there is a collision with the target object. With a closed loop system, these worse case situations could be recovered from, whilst specifying the torque requirements for the normal use case. A small PMSM drive such as those used in the five axis hand-held robot in Section 3.3 would be more than sufficient.

The motion of the robot is controlled by three Robot Operating System (ROS) topics, a stream of times to turn the nozzle on, a stream of times to turn the nozzle off, and position way points with a future time to arrive at the way point. The nozzle timing information is held in an ordered buffer; when the system time passes any of the stored times, the corresponding action is taken and then the entry is removed from the buffer. The position commands are acted on as soon as they arrive, with the velocity chosen such that the end effector arrives at the time associated with the position command. A diagram of the data flow can be seen in Figure 3.3.

3.2.3.1 *Calibration*

For the robot to work well with the algorithms presented in Chapter 6, the location of the end effector relative to the motion capture markers is required. When the constellation of markers is initialised in the motion capture software it is given a local coordinate system automatically. However this coordinate system is essentially arbitrary and not useful for referencing key

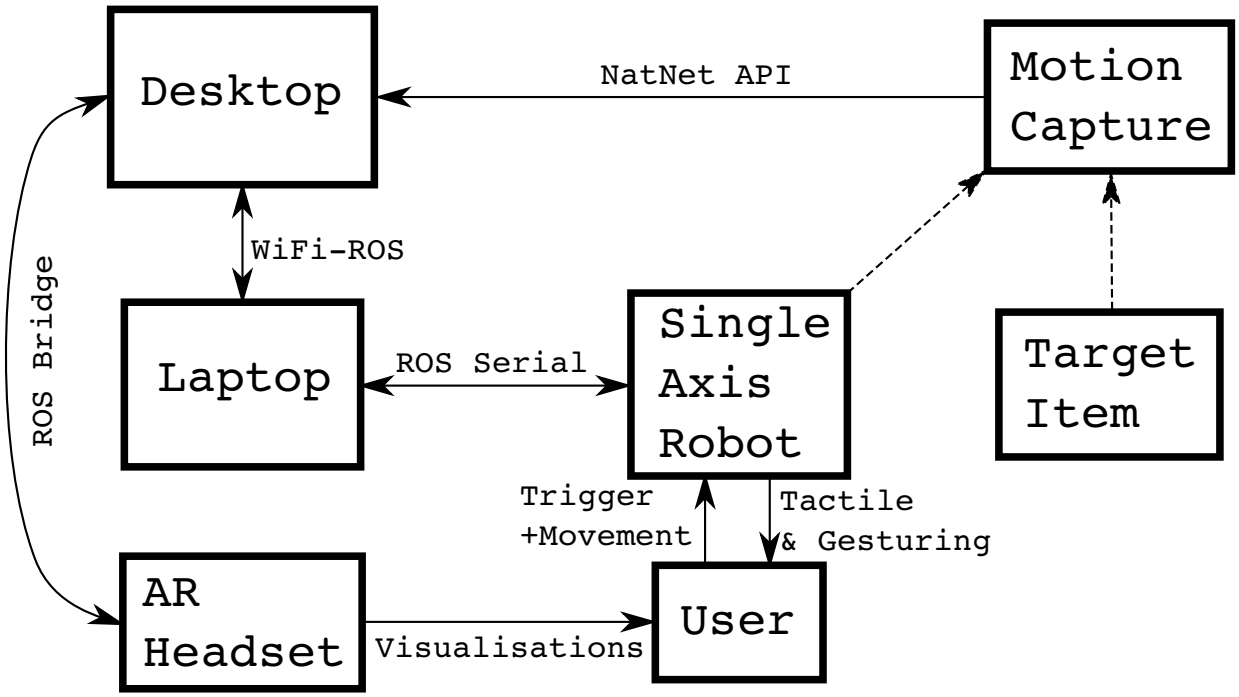


Figure 3.3: This shows the typical data flow for version 2 and 3 of the single axis hand-held robot system. The dotted arrows towards the motion capture block indicate that this is an observation from the motion capture system, rather than a traditional communication channel. The laptop is used as an intermediary to connect to the robot so that the system can be mobile around the motion capture area. A secondary function is that of a user interface for the system, especially for the administrator of the user trials.

points on the robot. A transform between this coordinate frame and one that is meaningful is required.

The motion capture markers on the robot were arranged in a specific way that can facilitate finding key information. Specifically, two of the markers are mounted on the same rod, and this rod aligned with the forward/backward direction of the robot. Therefore the difference between the location vectors of these markers, in the coordinate system assigned to the constellation, defines the forward direction, X, in the new coordinate system.

$$\mathbf{X} = \mathbf{P}_{x2} - \mathbf{P}_{x1} \tag{3.1}$$

Next the vertical direction is defined, Z, this is the axis in which the linear stage can move. To do this an additional marker is used, whose position can be reported in world space by the motion capture system. The robot is commanded to move the nozzle to its top of its range of movement, the additional marker is placed at the nozzle and a measurement of both the position of this point and the pose of the constellation are recorded. The same is repeated at the bottom of the range of movement of the nozzle. The top and bottom points can then be calculated in the coordinate space assigned to the constellation. $\mathbf{G}_{z1,2}$ are the bottom and top points as measured in global coordinates, $\mathbf{T}_{\text{con}}$ is the transform assigned to the constellation by the motion capture. $\mathbf{T}_{\text{hhr}}$ is the transform assigned to the centre point of the range of motion of the end effector, that is aligned with the meaningful directions, X, Y and Z, which are forward, left and up relative to the robot. Forward is the direction that the nozzle points.

$$\mathbf{P}_{z1,2} = \mathbf{T}_{\text{con}}^{-1}\mathbf{G}_{z1,2} \tag{3.2}$$

$$\mathbf{P}_{\text{c}} = \frac{\mathbf{P}_{z1} + \mathbf{P}_{z1,2}}{2} \tag{3.3}$$

$$\mathbf{Z} = \mathbf{P}_{z2} - \mathbf{P}_{z1} \tag{3.4}$$

$$\mathbf{Y} = |\mathbf{Z}| \times |\mathbf{X}| \tag{3.5}$$

$$\mathbf{T}_c = \begin{bmatrix} X_x & Y_x & Z_x & P_{c_x} \\ X_y & Y_y & Z_y & P_{c_y} \\ X_z & Y_z & Z_z & P_{c_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

$$\mathbf{T}_{hhr} = \mathbf{T}_c \mathbf{T}_{con} \tag{3.7}$$

3.3 FIVE AXIS HAND-HELD DRAWING ROBOT

Both the Inkjet robot and the single axis robot described above can operate in all the required DoF for a painting task, as long as the user holding the robot can collaborate effectively. However some tasks may require particular paths to be acted on in a particular order. An example is for high quality painting tasks such as those used for automobile body panels, the path is designed such that the spray head does not need to start and stop spraying while over the work piece, as can be seen in Chen et al. (2009). When a hand-held robot has reduced degrees of freedom, such as with the single axis robots described above, exact paths cannot be tracked unless the human acts perfectly. Instead the above robots are more suited to ad-hoc path generation, such as the method described in Chapter 6. To trace exact paths a hand-held robot must be *locally capable*, which is the ability to move the end effector in all the DoF required by the task, as defined in the global coordinate frame, without interaction from the user. With the exception being if the robot cannot reach the target, the user would need to move the robot into the vicinity of the target for it to reach.

3.3.0.1 *Kinematics*

The robot consists of a delta robot with revolute arms, which is based on that by Clavel (1990). The 4th and 5th axis stages are attached to the moving platform of the delta stage.

To set the end effector to a particular world position the inverse kinematics must be calculated. The inverse kinematics take the current position of the base ($\mathbf{B}$) as an input, which is centred at the IMU, and the desired location of the end effector ($\mathbf{X}$). The inverse kinematics outputs the motor angles $(\alpha_1, \alpha_2, \alpha_3, p, y)$ to achieve a particular end effector location. In Figure 3.6 the kinematic structure of the robot is shown in a simplified format. To calculate the desired motor angles there are two stages, calculating the angle between

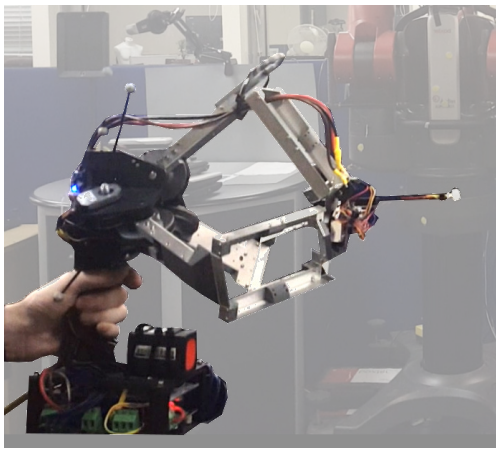


Figure 3.4: A photo of the 5 degree of freedom robot. It consists of a 3 degree of freedom delta stage followed by a 2 degree of freedom rotational stage. Tracking is achieved by fusion of IMU measurements and motion capture tracking markers on the base of the robot.

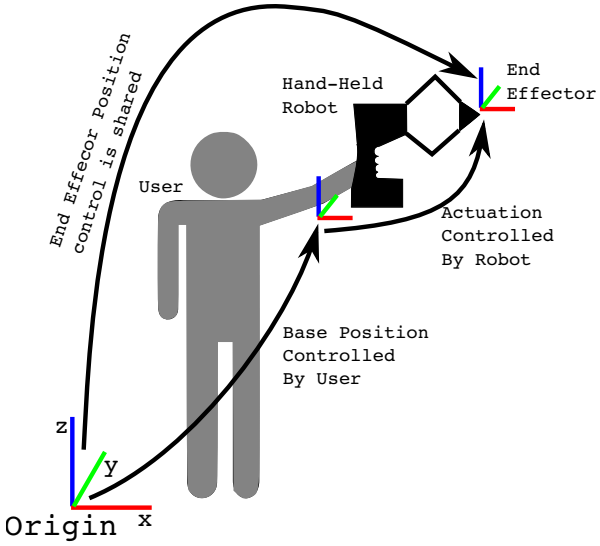


Figure 3.5: A high level view of how the end effector is positioned. The user has exclusive control of where the robotic base is. The actuation of the end effector is exclusively controlled by the robot. However, as the end effector is connected to the base of the robot, the total position in world space of the end effector must be a shared effort between the user and the robot.

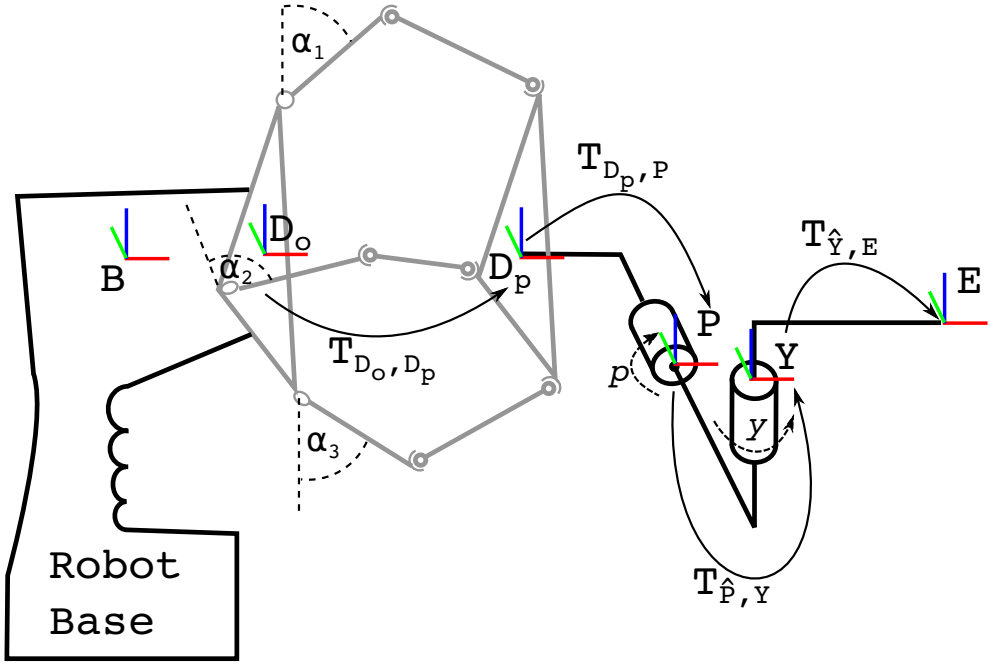


Figure 3.6: A simplified diagram of the kinematic structure of the robot. Values $\alpha_1, \alpha_2, \alpha_3, p$ and y are the motor angles that can be controlled. $\alpha_1, \alpha_2, \alpha_3$ are the input to the delta stage and are jointly responsible for the movement of the platform in x, y and z. p controls the pitch of the end effector and y controls the yaw of the end effector. Not shown are the post rotation poses of the pitch and yaw axis ($\tilde{\mathbf{P}}, \tilde{\mathbf{Y}}$), and the redundant roll transform at the end effector. These extra transforms are used in the discussion of the kinematic calculations, but are not crucial to understanding the structure of the robot.

the delta origin ($\mathbf{D}_o$) and the target ($\mathbf{X}$), which sets the rotation axis angles, then solving for the translational offset of the delta stage. For the following $\mathbf{T}$ denotes a rigid transform between the poses specified in its subscript.

Calculating the rotation offset is done as follows:

$$\mathbf{D}_o = \mathbf{T}_{B,D_o}\mathbf{B} \tag{3.8}$$

$$\mathbf{T}_{D_o,X} = \mathbf{X}\mathbf{D}_o^{-1} \tag{3.9}$$

$$y, p, r = \text{getEuler}(\mathbf{T}_{D_o,X}) \tag{3.10}$$

These values can be used directly to set the position of the rotation servos located on the moving platform ($\mathbf{D}_p$) of the delta stage. Roll, pitch and yaw are then separated into their own transforms, $\mathbf{T}_{E,X}$, $\mathbf{T}_{P,\tilde{P}}$, $\mathbf{T}_{Y,\tilde{Y}}$ respectively. Where $\tilde{\mathbf{P}}$ and $\tilde{\mathbf{Y}}$ represent the pose of the post-rotation pitch and yaw axis.

The roll axis is redundant in this design an so the rotation r used to correct for the misalignment of the end effector with the target location in roll axis ($\mathbf{T}_{E,X}$).

Now that the rotational axis are set, there is only one unknown, which is the transform that is defined by the delta stage ($\mathbf{T}_{D_o,D_p}$):

$$\mathbf{T}_{D_o,D_p} = \mathbf{D}_o^{-1}\mathbf{X}\mathbf{T}_{E,X}^{-1}\mathbf{T}_{\tilde{Y},E}^{-1}\mathbf{T}_{Y,\tilde{Y}}^{-1}\mathbf{T}_{\tilde{P},\tilde{Y}}^{-1}\mathbf{T}_{P,\tilde{P}}^{-1}\mathbf{T}_{D_p,\tilde{P}}^{-1} \tag{3.11}$$

Or equivalently:

$$\mathbf{T}_{D_o,D_p} = (\mathbf{T}_{D_p,P}\mathbf{T}_{P,\tilde{P}}\mathbf{T}_{\tilde{P},Y}\mathbf{T}_{Y,\tilde{Y}}\mathbf{T}_{\tilde{Y},E}\mathbf{T}_{E,X}\mathbf{X}^{-1}\mathbf{D}_o)^{-1} \tag{3.12}$$

If $\mathbf{T}_{D_o,D_p}$ represents a translation that is within the accessible envelope of the delta stage, then the delta kinematics is solved using the method presented by Clavel Clavel (1991). The implementation used here was an open source solution by Szymon Szantula[3]. In the case that the translation falls outside of the accessible envelope of the delta stage then $\mathbf{T}_{D_o,D_p}$ is scaled towards the centre of the accessible envelope ($\mathbf{D}_c$), with a scale transform ($\mathbf{F}_{scale}$). $\hat{\mathbf{T}}_{D_o,D_p}$ defines the modified input to the delta kinematics.

$$\hat{\mathbf{T}}_{D_o,D_p} = \mathbf{T}_{D_o,D_c}\mathbf{F}_{scale}\mathbf{T}_{D_o,D_c}^{-1}\mathbf{T}_{D_o,D_p} \tag{3.13}$$

The scale variable is defined as follows, only positions are needed, so vectors are used instead:

$$\text{scale} = R\frac{\mathbf{D}_c\vec{\mathbf{D}}_p}{\left\|\mathbf{D}_c\vec{\mathbf{D}}_p\right\|} \tag{3.14}$$

This is entered into a scale transform matrix:

$$\mathbf{F}_{scale} = \begin{bmatrix} \text{scale} & 0 & 0 & 0 \\ 0 & \text{scale} & 0 & 0 \\ 0 & 0 & \text{scale} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.15}$$

This has the effect of limiting the delta platform to a sphere of radius R centred on location D_c, which is the centre of the accessible envelope. Now the above system taken as a whole will give a valid input for $\alpha_1, \alpha_2, \alpha_3$ for the delta stage and p, y for the pitch and yaw motors that will either align

3 https://github.com/byq77/DeltaKinematics

the end effector $\mathbf{E}$ with the target X or put the end effector in the closest position within the working envelope of the robot, $\hat{X}$.

3.3.1 *Five Axis Hand-held Drawing Robot Construction*

The primary benefit of using a delta mechanism in this application is that it is a parallel mechanism. All the motors for the delta stage are located on the main body of the robot, which the user holds. This allows the moving parts of the robot to be very light, compared to a serial robot, where there are multiple articulated joints connected in a chain. This means that the motors for the later joints need to be carried by the preceding joints, or the articulation must be achieved with tendons. Having the motors mounted on moving parts of the robot would necessitate each axis of the robot to be larger and more rigid to carry the extra weight. Using cable driven joints can be mechanically complicated, and introduce unwanted backlash in the joints if the cables stretch. Delta mechanisms are also quite complicated when fabricated in a traditional way, though in Section 3.3.1.1 a simplified fabrication technique is introduced. Therefore a delta mechanism solves both the moving mass and complexity issues.

The 4th and 5th axis mounted on the moving platform of the delta stage are constructed in a serial manner, with the 5th axis motor being connected to the output of the 4th axis motor. These axis allow for rotation of the end effector, in the yaw and pitch directions. The roll axis is omitted as many tasks are invariant to roll, such as spray painting with a rotationally symmetric nozzle. The 4th and 5th axis were constructed in a serial manner, rather than in a parallel manner because the majority of designs that are analogous to the delta design with 5 or 6 DoF have a limited range of motion in which the end effector can rotate significantly. This is demonstrated by Mirshekari et al. (2016) for a small selection of the common six DoF parallel manipulator designs. The serial design allows a range of motion of 60 degrees in both pitch and yaw across all of the valid envelope for the delta mechanism.

The motors driving the delta stage are Brushless Direct Current (BLDC) motors, also know as a PMSM. These motors are typically used for model aircraft and aerial robotics, and as such are light weight and have a high peak power handling capability. In this case the motors are the *Tarot 4008 330kv* variety, with a weight of 80g and a peak power handling of 600W. These are driven by two ODrive motor controllers[4]. An encoder is connected to the back of each motor, the *CUI AMT102* was used in this case, which gives

4 www.odriverobotics.com

a precision of 8196 counts per revolution. The motors drive a belt reduction stage, which has a ratio of $\frac{89}{24} = 3.71$, the output of the belt reduction is rigidly connected to the primary arms of the delta stage.

The robot is powered by a 1300mAh 5 cell lithium polymer battery, chosen to be light weight while also able to supply the maximum current of all the motors on the robot, which is roughly 100A peak. The battery is rated to 65C discharge, allowing for $1.3 \times 65 = 84.5$A under steady discharge conditions, or double this for short bursts. This choice of battery shifts the power handling limitations to the thermal performance of the motors and the acceptable torque applied to the users hand, as well as the strength of the delta linkage itself.

3.3.1.1 *Constructing the Delta Linkages*

One of the advantages of a Delta robot is that the moving mass can be made very light, and hence can be accelerated more aggressively for the same force, as well as limiting deformation of the mechanical elements. However to maintain this benefit care must be taken in constructing the linking arms and joints, allowing these elements to be heavy will reduce the benefit of the delta topology. The joints in the delta linkage are traditionally ball joints, sometimes known as rose joints. Designing these with high precision, and large movement throw, while maintaining low weight is difficult. This problem is described by Lung-Wen Tsai et al. (1996). Their solution was to use revolute joints rather than ball joints. Revolute joints are those which only allow the joined elements to rotate around one axis, and do not allow any translation. For each ball joint 3 revolute joints must be used. Lung-Wen Tsai et al. (1996) also describe a simplified topology that does not emulate the traditional ball joint linkage, but instead uses only 2 revolute joints at each articulation location on the delta linkage. This removes the DoF that allows the secondary members of the delta linkage to rotate, which is not necessary for the correct operation of the delta linkage. The reduction in joint number also reduces cost, weight and complexity.

To further reduce cost, weight and complexity a linkage was designed using a novel composite material processing method. All of the delta arms and joints are made from one material, in one manufacturing process. This is achieved by using an aluminium-polypropylene-aluminium composite sandwich sheet material, which is marketed under the name *Hylite*. The aluminium in the composite sandwich can be selectively removed by a CNC milling operation to expose the polypropylene. When the aluminium is removed from both sides the structure can bend freely, and thus a hinge can

be created. With a selection of hinges and a cut-out milling stage, a full delta linkage can be constructed in a single piece, complete with various mounting holes and features to increase arm rigidity. The design of the hinges and cuts can be seen in Figure 3.7.

This design was influenced by the work of Marc Peltier, who designed and marketed the *Zatsit* 3D printer design[5]. That design was also a delta construction, though the linkage is actuated in via linear actuators, rather than a rotational arm connected to the motors. A summary of the various types of linear delta has been detailed by Bouri and Clavel (2010). The *Zatsit* delta mechanism consists of multiple sub assemblies that must be fastened together, which simplifies the tessellation of the pieces on the stock material. The drawbacks of multiple sub assemblies are the introduction of additional assembly stages, as well as increasing the likelihood that the critical lengths won't match those that are used in the kinematics.

To ensure the rigidity of the arm sections it is useful to make sure the cross section of the arms have a closed loop. This is because closed loops significantly increase the torsional rigidity of beams, as originally described by de Saint-Venant (1856), and summarised into practical advice by Hughes et al. (2012). Hence closure of beam sections is achieved with optional 3D printed brackets attached to the composite structure with screws. These can be omitted at the cost of reduced rigidity from the delta mechanism.

5 http://www.zatsit.fr

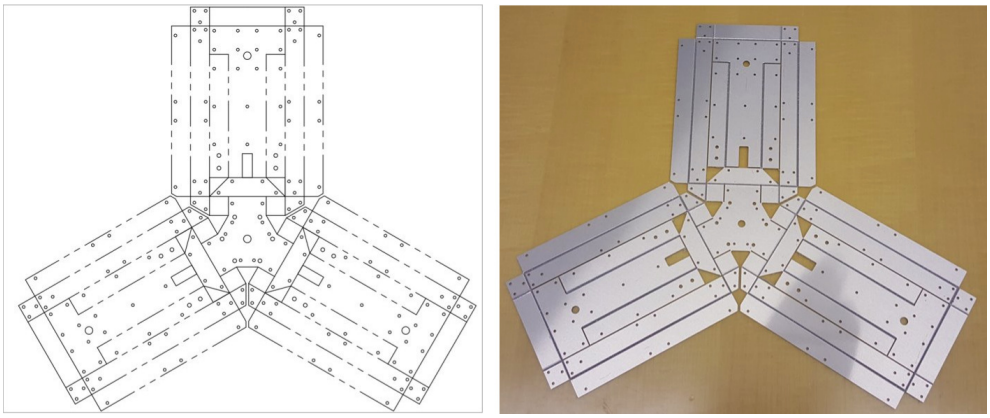


Figure 3.7: This shows the delta linkage before being folded into it's final form and brackets being added. The material is *Hylite*, aluminium-polypropylene-aluminium composite sandwich. Hinge sections have had the aluminium milled away leaving only the polypropylene to act as a living-hinge. No additional parts are needed for the correct functioning of the linkage, though to maximise rigidity mounting points for brackets are cut. Brackets to fix bends at a certain angle are also used to ensure the active hinge joints are centred correctly, minimising over-excursion of the joints during use.

3.3.1.2 *Arm Geometry*

When constructing a delta robot there are a number of parameters that can be altered to change the performance of the robot. Clavel (1991) discusses these decisions in detail, of particular interest is the ratio between the first arm, which is actuated, and the second arm, which is connected between the first and the platform. His terminology refers to the first arm length as L_A and the length of the second arm as L_B. Clavel presents arguments regarding the trade off in altering the ratios of these lengths, they can be summarised as follows: (L_A is held constant.)

- A large L_B allows for a greater reachable area.
- A small L_B allows for a greater precision.
- A small L_B allows for more space to connect arms at the platform, reducing the chance of self-intersection.
- $L_B <= L_A$ will create inadmissible motor angles.

Clavel concludes that a value of $\frac{L_B}{L_A} \approx 2$ provides a good compromise between these factors. Further, at this value the precision for the X, Y, and Z movements at the moving platform are roughly similar.

Clavel also discusses the value $r = \frac{R}{L_A}$, where R is the additional offset from the centre of the base and the motor axis, when compared to the distance between the centre of the platform to the second arm attachments. Some effects of r are as follows:

- A large r increases precision.
- A large r decreases reachable area.
- A large r increases the space available to connect arms, reducing the chance of self intersection.
- r must be less than $L_A + L_B$.
- A large r moves singularities closer to the centre of the work area.

Through experimentation with various values Clavel suggests that the following is a good fit for most applications.

$$r = 0.63 \tag{3.16}$$

The above discussion is convincing in the case where the designer is not aware of the specifics of the use case for the delta mechanism, and must design to meet a general case. However optimisations for particular cases may differ from the suggested parameters. For example a final use case could require high precision in Z, and low precision in X and Y with a large working area in X and Y. In that case it could be suggested that $\frac{L_B}{L_A}$ should be a larger value.

For the use case in hand-held robotics there are some unusual optimisations that are not covered by the discussion above. It is useful for the hand-held delta robot to have a compact resting position for storage and handling when the robot is not in use. Further the average distance from the moving platform to the users grip should be minimised, to reduce the torque on the users hand during platform accelerations. The distance from the platform to the retro-reflective marker constellation should also be minimised to reduce the error in the calculated platform location due to uncertainty in the robot orientation.

For these extra considerations the following can be said:

- The reachable area should be located close to the centre point of the motors.
- r should be small to minimise the size of the motor arrangement.

To move the reachable area close to the centre point of the motors implies that $\frac{L_B}{L_A} \approx 1$ and $r \approx 0$ as this makes the centre point of the motors an accessible location for the platform. However this location would not be available due to intersection with the robot body, hence a value > 1 is required. This also allows the motors to move back to a negative angle, that is so they are pointing away from the platform. This position is desirable as a resting position as it is likely to be stable when the motors are not engaged, not allowing the platform to sag under gravity when the robot is turned off. Some small amount of friction is needed in practice, though this negative arm position minimises this required friction. In the hand-held design r is to be minimised based on space constraints of gearing and other self-intersection considerations.

In the case of the linkage shown in Figure 3.7, $\frac{L_B}{L_A} = 1.02$. In future work this could be increased further to allow additional range of motion in the primary arms, such that they can simultaneously press their homing switch mounted to the robot body. The additional motor axis offset is $r = 0$. Although precision of the end effector could be increased by increasing r, this is of little practical importance due to the precision of the end effector being adequate for the tasks investigated in this work.

3.4 TRACKING OF THE HAND-HELD ROBOTS

Due to the fact that the hand-held robot's end effector can be moved in two ways, either the robot actuating its motors to move the end effector, or the user manoeuvring the whole robot, we must track the position of the body of the robot to infer the position of the end effector at any given time. There are many ways to do this, for example: optical tracking, on-robot visual odometry, magnetic tracking, and UWB radio tracking. This section will focus on the particular design choices and methods of tracking used in the robots presented above.

3.4.1 *Retro-reflector Based Infra-Red Motion Capture*

This motion capture method makes use of retro-reflective spheres attached to the object of interest, which are observed and triangulated by cameras mounted in the environment. The cameras have Infra-Red (IR) Light Emitting Diode (LED)s around their lens, which shine into the working area. Retro-reflective markers return the light back towards the source direction, ensuring that the cameras have a very high contrast between marker and

non-marker areas of the image. This type of motion capture is used with version 2 and 3 of the single axis robot, as well as with the five axis robot.

The specific system that is used with the robots in this work is an *Optitrack* system using the *Motive* software. There are 8 *Flex 3* cameras surrounding a work space of approximately $4 \times 4 \times 2\text{m}^3$. Data is communicated via the *NatNet* Application Programming Interface (API) to the ROS system running on a client machine which serves as the hub for the robotic system. The whole system reports positions at a frequency of 100Hz with 13ms of latency.

3.4.2 *Sensor Fusion of IMU and Optical Motion Tracking*

The optical motion tracking system described above works well for tracking an object or robot. However, if this position information is included in a feedback loop on the robot, such as a controller attempting to fix the pose of the end effector of the 5 DoF hand-held robot, the system can become unstable. This is due to the latency of the system being significant compared to the bandwidth of the actuation system on the robot. There are a number of ways this could be fixed:

- The motor bandwidth could be artificially reduced.
- The measurements from the motion capture could be smoothed with an Infinite Impulse Response (IIR) or Finite Impulse Response (FIR), lowering the measurement bandwidth.
- More selective filters IIR or FIR could be used to target particular resonant nodes of the system.
- A mechanical model could be developed to make predictions of the current state and provide a feed forward control term to minimise overshooting based on estimated dynamics.
- The position of the robot could be estimated using an IMU combined with the optical motion capture data, allowing a low latency prediction of the current state.

This section will focus on the last solution. Solutions based on filtering necessarily depend on factors such as how the user is holding the robot, and filters sufficiently broad to encompass all likely modes of resonance under all conditions are likely to limit the performance of the robot. This is also true for reducing the bandwidth of the motor system, which is can be considered a crude filtering method.

The method chosen was to use an ESKF to combine IMU measurements with those that are gathered from the motion capture. The ESKF was summarised in Section 2.6.1 and a more detailed description of the mathematics and notation can be found in Sola (2017). The ESKF produces an estimate of both the current state of the robot ($\mathbf{x}$) and the uncertainty of the state ($\mathbf{P}$). The state of the robot consists of its position, velocity, orientation, and the biases of both the accelerometer and the gyro. The advantage of using ESKF is that it optimally combines the current estimate of the state and measurements from the motion capture and IMU into the new estimate of the state based on their individual uncertainties.

If the measurements from the IMU and the motion capture were both up to date at the time of processing by the ESKF, this would be an adequate solution. However, due to processing and transfer time from the motion capture system, there is a 13ms latency from when the cameras capture the scene and the measurement arrives at the robot. This causes a lag in the pose estimate, as well as potentially corrupting the sensor bias estimates. Therefore it is required to correct for this time delay.

As summarised in Section 2.6.1.2, Larsen et al. (1998) proposed a method that can account for measurements with latency without loss of optimality. However, the time that the measurement is initiated must be known, as factor $\mathbf{M}_*$ needs to be accumulated until the measurement arrives.

To be able to relax this requirement a modification is proposed. A buffer of the previous states and their timestamps is maintained, which is then compared to the timestamp of the incoming optical motion tracking based measurement. Using a matching set of observation and states based on the timestamp, the update is calculated as normal, which is then applied to the current state estimation.

More formally, the implementation is described as follows. Hold the set of k previous estimates of the state.

$$\mathcal{X} = \{\hat{\mathbf{x}}_{t-ks}, \hat{\mathbf{x}}_{t-(k-1)s} ..., \hat{\mathbf{x}}_{t-s}, \hat{\mathbf{x}}\}. \tag{3.17}$$

This set is updated on each new IMU measurement, with period s, and is held in memory as a ring buffer.

The measurement is taken the normal way, however the state observed is out of date by d

$$\mathbf{y} = h(\mathbf{x}_{t-d}) + v. \tag{3.18}$$

When calculating the estimate of the error, the buffer is queried for the state whose timestamp is closest to the time of the delayed measurement

$$\hat{\delta \mathbf{x}} \leftarrow \mathbf{K}(\mathbf{y} - h(\mathcal{X}[t - d])), \tag{3.19}$$

where $\mathcal{X}[t - d]$ denotes retrieving the corresponding state estimate of the buffer at time $t - d$.

The state is then updated according to Sola (2017)

$$\mathbf{x} \leftarrow \mathbf{x} \oplus \hat{\delta \mathbf{x}}, \tag{3.20}$$

as well as the Kalman gain ($\mathbf{K}$) and the covariance ($\mathbf{P}$).

3.4.2.1 *Verification of Time Correction Method*

This method was verified by recording a series of motion capture and IMU time series from the 5 axis robot during highly dynamic movements. A benchmark of accuracy against a ground truth measurement was not possible, as using a reference system that was more trustworthy than the motion capture used in the sensor fusion was not feasible. Instead, a reference method was produced by pre-processing the time series of IMU and motion capture data such that they could be entered into the unmodified ESKF with strictly monotonically increasing timestamps. This emulates having a system that has no latency. The modified method can then be compared to this latency free reference.

The results can be seen in Table 3.2. The time correction method reduces the mean error by a factor of 26 to 0.29mm, and the worst case error over the sequence by a factor of 5.85 to 6mm. As was demonstrated by Merriaux et al. (2017), the expected error for a motion capture system can be between 0.15 and 2mm. Therefore, measurement error is likely to dominate the error due to time misalignment under typical conditions. Even the worst case condition of 6mm is not significant in the experiments performed later in this work, though this worse case error was found under high accelerations, which is not typical of the motion in those experiments.

Correction Method	Mean Error (mm)	σ (mm)	Worst Case Error (mm)
No Correction	7.6	7.5	35.1
Proposed	0.29	0.32	6.0

Table 3.2: A summary of the error of the proposed time correction method compared to not correcting for the time error. The ground truth is generated by re-ordering the input data such that it emulates a system that has no latency.

3.5 CONCLUSION

In this chapter the original two hand-held robot designs and the tracking methods used with them were presented.

The first robot was one with only a single DoF, that was designed in three iterations. This robot was designed to be as simple as possible, such that experiments could be undertaken to test the properties of a *reduced degree of freedom* hand-held robot. These experiments will follow in Chapter 6 and 7. A method to calibrate the nozzle position for this robot was outlined, which required the placement of markers that indicate useful direction vectors.

A second robot was presented that has five DoF. This robot was designed to have adequate degrees of freedom to fully correct for user motion when applied to a five DoF task. This robot was produced for the purpose of testing the properties of a *locally capable* hand-held robot. These experiments will follow in Chapter 8. A novel delta mechanism construction technique was presented that minimises cost and complexity of manufacture by using only a single material, processed in one stage and that requires no assembly of sub-parts. Testing the mechanical properties of this construction method in a rigorous way is a fertile direction for future work, as well as extending the method to different robot topologies.

A framework for an ESKF with time correction was presented, and a modification to the time correction method was tested and found to significantly reduce estimation errors. This method removes the requirement that the robot knows the time at which an external measurement is taken before that measurement arrives. This method works by keeping a buffer of previous estimated states. A late arriving external measurement is then temporally matched with one of these states. The calculation of the error state then continued as described by existing literature, as well as the update applied to the current state.

The low computational overhead and lack of assumptions on the generation time and frequency of motion capture observations are beneficial, and

the error compared to an ideal system is small enough not to be a concern in developing a hand-held robot in this case.

4

AUGMENTED REALITY ALIGNMENT

4.1 INTRODUCTION

There are a number of instances within this thesis where items, robots or coordinate systems must be aligned with an augmented reality display. *"Aligned"* in this context means that the computer graphics displayed through the AR display appear where they should. Examples being the mannequin in Chapter 7 or the secondary visualisation presented in Section 8.3.3. For these items to be aligned, the transform between the robot world space and the AR headset world space is needed. Additionally, the model space coordinates of the items being rendered onto must be known. This is especially necessary due to the motion capture system assigning an arbitrary coordinate system to newly instantiated marker constellations. These arbitrary coordinate systems are unlikely to be meaningful to the model, the origin of the scanned 3D model, or the origin of the robot's own coordinate system for example.

This chapter presents two methods for achieving these alignments. The first is the *Singular Value Decomposition (SVD) method*, which is mathematically simple, using a classical SVD point alignment strategy. This is presented both as a method to discover a transform that can redefine the pose provided by a motion capture system to be more meaningful, as well as a method to find the offset between the coordinate systems of the motion capture and the AR headset. The *SVD method* requires manually defined reference points to be identified by the user.

A second method for aligning the coordinate systems that removes some of the constraints of the *SVD method* is proposed. This is called the *time series method*. With the *time series method* the user does not need to carefully align any reference points, instead they must walk around the arena. This provides a time series of points from both the AR headset, and from the motion capture marker attached to it.The result of the method is a transform that can convert between the motion capture space and the AR headset space, and a calibrated transform between the attached marker and the AR headset.

The *SVD method* was used in the experiments in Chapter 5 and 8. The *time series method* was used in the experiments in Chapter 7. These decisions were made only for convenience in running the user trials. Whilst the *SVD*

method does require careful alignment of reference points, this can be done by the experimenter. This is often simpler than requiring the user to walk around the arena. In the case where a new user needed to initialise the system themselves the *time series method* is likely to be more convenient.

4.2 THE *svd method*: FINDING A TRANSFORM TO A MEANINGFUL LOCAL FRAME

When using a motion capture system, a constellation of retro-reflective markers is attached to the object to be tracked. When initialising the object in the motion capture software, a local coordinate system is assigned to the object. The marker locations are then stored in this local coordinate system, for use in matching the tracked item.

However, the local coordinate system in which the constellation is defined is arbitrary, and difficult to modify to be meaningful. For example, it would be useful to have the origin of the coordinate system be located at a easily defined reference point on the robot, and the orientation to conform to the ROS coordinate system (X forward, Y left, Z up).

In Section 3.2.3.1 this was performed using markers that have been placed in a specific way to facilitate finding the direction vectors, as well as requiring additional measurements taken manually. This is not always possible to achieve, for example, if there is no convenient place to mount markers aligned with meaningful direction on the object.

The objective of the *SVD method* is to find a transform that can correct the pose that is provided by the motion capture system, with the resulting pose being centred and oriented in a meaningful way.

For the *SVD method* it is required to know the position of the motion capture markers in the coordinate system that is considered meaningful. In the case of the five axis robot, the location of the markers was modelled in the 3D Computer Aided Design (CAD) package. In the case of the mannequin use in the experiments in Chapter 6, the location of the markers was found by 3D scanning the mannequin and locating the markers in the scan. Additionally the correspondence between points in the meaningful coordinate space and those provided in the arbitrary constellation coordinate space should be known.

To achieve this alignment a strategy described by Ho (2013) is used. The method is as follows:

First the centroids of each set of points are found by averaging their respective points. Subscripts a and b refer to the two sets of points to be

matched, set a is the set defined in the arbitrary coordinate system, set b is the set defined in the meaningful coordinate system. N is the number of matched points, which must be more than three, and C is the centroid position for a set of points.

$$\mathbf{C}_a = \frac{1}{N}\sum_{i=0}^{N-1}(\mathbf{P}_a^i) \tag{4.1}$$

$$\mathbf{C}_b = \frac{1}{N}\sum_{i=0}^{N-1}(\mathbf{P}_b^i) \tag{4.2}$$

Next the relative rotation to best align the data sets is found, here R is a rotation matrix. U,S,V are the typical outputs of SVD and t is the translation between the sets.

$$\mathbf{H} = \sum_{i=0}^{N-1}(\mathbf{P}_a^i - \mathbf{C}_a)(\mathbf{P}_b^i - \mathbf{C}_b)^T \tag{4.3}$$

$$[\mathbf{U}, \mathbf{S}, \mathbf{V}] = \mathrm{svd}(\mathbf{H}) \tag{4.4}$$

$$\mathbf{R} = \mathbf{U}\mathbf{V}^T \tag{4.5}$$

$$\mathbf{t} = \mathbf{C}_b - \mathbf{R} \times \mathbf{C}_a \tag{4.6}$$

Using the transform described by t and R, points of interest defined in the first space can be moved into the second space by:

$$\mathbf{B} = \mathbf{R}\mathbf{A} + \mathbf{t} \tag{4.7}$$

Or equivalently:

$$\mathbf{T}_{hhr} = \mathbf{T}_c\mathbf{T}_{con} \tag{4.8}$$

Where $\mathbf{T}_{con}$ is the transform from the origin to the constellation pose, provided by the motion capture, $\mathbf{T}_c$ is the transform defined by R and t above, and $\mathbf{T}_{hhr}$ is the transform from the origin to the pose centred and oriented in a meaningful way.

Therefore if the markers can be found in respect to a point of interest, either through direct measurement, or using the CAD software in which the tracked object is defined, the pose generated from the motion capture system can be transformed to a pose that is representative of some meaningful point in the object. This could be the origin of the 3D scan for the mannequin, or the centre point of the delta arms for the five degree of freedom robot.

4.3 AUGMENTED REALITY WORLD ALIGNMENT

In later chapters visualisations are shown to the user whilst using handheld robots to facilitate communication between the robotic system and the user. However, for these visualisations to be useful they should be rendered precisely such that there is alignment between the real world objects and visualisations.

To achieve this alignment the offset between the coordinate system of the head mounted display and the rest of the robotic system needs to be known. Presented here are two methods for achieving this, the first is a simple method where the user is required to identify known points in the environment, and a second one where no specific actions are required of the user.

4.3.1 *Spatial Anchor Alignment*

This method is very similar to that presented in Section 4.2. However instead of trying to find the transform between the constellation of markers in model space and the same constellation in the arbitrary coordinate system, the transform between the AR headset coordinate system and the robotic system's coordinate system is solved.

The two sets of constellations that must be aligned in this case are locations that are known in both the AR headset coordinate system and the robotic system's coordinate system. In the case where a motion capture system is being used, it is convenient to use the camera locations, as generally these will not be moved. The Optitrack system that is used in experiments presented later in this work can report the location of the cameras within its own coordinate system. The motion capture coordinate system is also used as the primary reference coordinate system for the robots.

The corresponding points must now be reported by the AR headset. To do this, a user must wear the headset and place holograms that coincide with the motion capture cameras. These holograms are numbered and should be placed with the corresponding cameras.In the case of the HoloLens, a system known as *spatial anchors* can be used to simplify the placement of the holograms. This system uses the geometry present in the vicinity of a placed marker to make it be persistent between launches of the application on the HoloLens, meaning that this process only needs to be done once.

Now the point constellation as reported by the AR headset and the motion capture system can be processed in a similar manner to that described in Sec-

tion 4.2, using SVD to estimate the least squares approximation of the offset translation and rotation between the two spaces. Whenever a visualisation is to be displayed, the offset transformation can be applied to the location that is specified in the robot coordinate system to find its equivalent location in the AR headset coordinate frame.

The *SVD method*, with spatial anchors, was used successfully in aligning simulated spraying experiments in Elsdon and Demiris (2018*b*) and in aligning visualisations for assisted wheelchair experiments in Zolotas, Elsdon and Demiris (2018).

4.3.2 *The* Time Series *Method: Marker Based World Alignment*

The *SVD method*, with spatial anchors, described in Section 4.3.1 is simple and works well, however it has two major downsides. Firstly it requires the user to place markers manually and align them carefully using the apparent location as shown through the AR headset. This is cumbersome and must be done every time the AR application is reinstalled. Secondly, due to the fact that the spatial anchors used in the HoloLens system are defined by local geometry and continually refined based on incoming tracking measurements, the apparent location of objects will drift as the reference constellation locations are changed. The method presented in this section solves both these issues, as it requires no specific user interaction to set reference points, and it uses only the location of the headset, which is less prone to drift than reference points.

This method is based on the idea that a correspondence between the motion of a marker attached to the AR and the motion reported by the AR headset can be found.

Due to there being no well defined datum to use as a reference on the Microsoft HoloLens there is no simple way to align a marker with its axis, or a known point. Therefore any marker attached to it has to be assumed to be arbitrary, containing no helpful information to find the points of interest on the headset.

The HoloLens does provide its location within its own coordinate frame that was generated with visual odometry. The challenge is to calculate the offset between the HoloLens world frame and the global world frame, in this case defined by a motion capture system.

4.3.2.1 *The* Time Series *Method: Implementation*

In contrast to the techniques introduced in Background Section 2.6.2, the *time series* method does not rely on user alignment tasks or on well defined marker setups, though it must be noted the difference in aims of the calibrations. The *time series* method implicitly relies on the built in calibration of the HoloLens and the interpupillary distance calibration routine provided by Microsoft, which can be replaced with a manual interpupillary distance measurement. Also the other methods of calibration, such as SPAAM (Tuceryan et al., 2002), can correct for some other modes of error such as perspective correction, which will be taken for granted in this method.

In order for the visualisation to be rendered smoothly to the users, the HoloLens calculates the movement of the user's head in real time on the headset itself. Whilst this can be overridden, setting the user's virtual position based on external measurements, the additional latency makes for an unacceptable experience. Also if tracking from an external system is lost, the user would notice the lack of position updates. Therefore this system only acts on changing the location of the HoloLens origin point, then allows the HoloLens to maintain high frequency updates of the user's position in that frame.

In order to gain information about the HoloLens' location relative to the world frame, defined by the motion capture system, a rigid body of reflective markers was attached to the headset. Due to the fact that the optical centre of the HoloLens depends on the particular calibration to the current user, the offset between the attached marker and the current optical frame is not static. Thus there are two unknowns in the system, offset between motion tracking origin and HoloLens world origin and the offset between the headset marker position and the optical centre of the HoloLens. With two unknowns it is not possible to solve for both with one measurement. These unknowns are found by a process of optimisation over a time series of observations. Further, the result should be continually adjusted due to the drift introduced by the HoloLens visual odometry system.

Figure 4.2 can be formalised using matrix equations where each matrix is representing a rigid rotation and translation. In Equation 4.9 T_{ho} is a rigid transform from the HoloLens frame to the motion capture frame, T_{ro} is the rigid transform that describes the offset between the marker on the headset and the optical centre of the headset. P_h is the pose of the HoloLens as reported by the HoloLens' internal visual odometry, in its own frame of reference. P_r is the measured pose of the marker attached to the HoloLens in the motion capture frame.

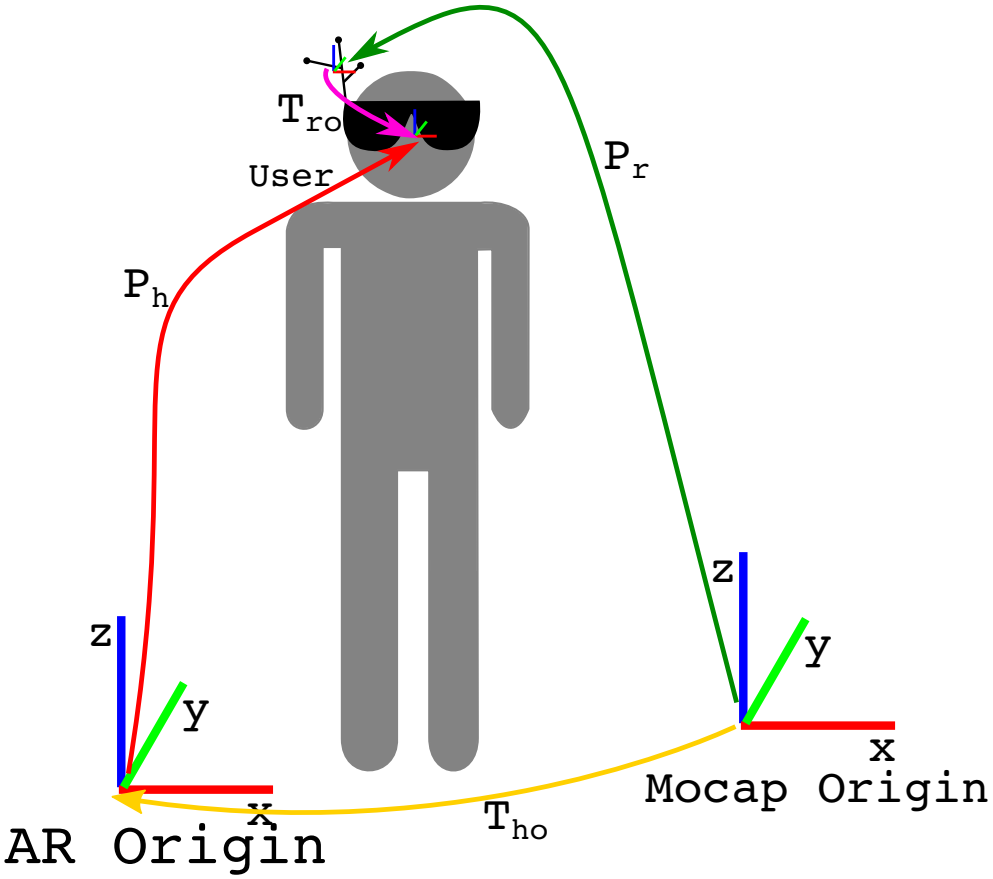


Figure 4.1: A diagram of the transforms used in the *time series method* of AR alignment. This diagram shows one time instant, refer to Figure 4.2 to see how these are structured in a time series.

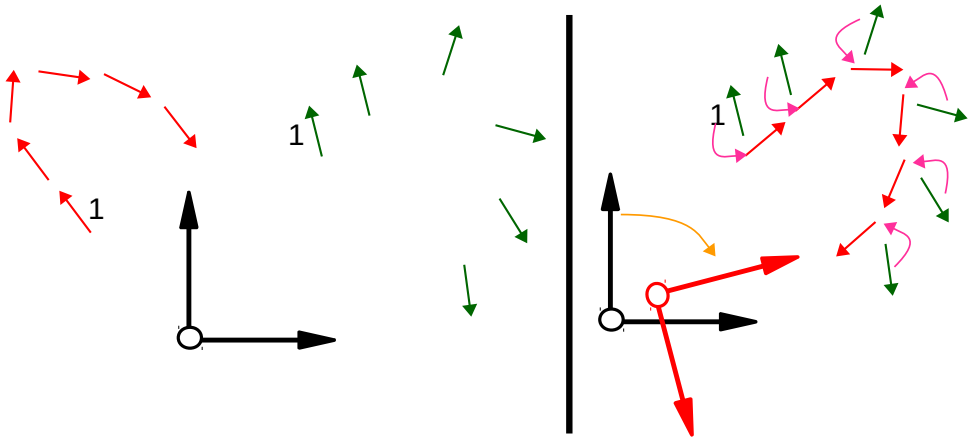


Figure 4.2: A Summary of the transforms involved in the online method proposed for aligning the AR headset with the motion capture coordinate system. Red: HoloLens position in HoloLens frame of reference. Green: Rigid body marker position in the motion capture frame of reference. Pink: marker offset from optical centre. Orange: transform between HoloLens frame of reference and the motion capture frame of reference. The '1' indicates that these are matched measurements, the other pairs are matched but are not numbered for clarity. It can be seen that there is one transform that will align the HoloLens reported values (red) with the motion capture reported values (green) in such a way that there is a unique transform after this coordinate shift that links the optical centre and the headset mounted marker. This unique transform describes the transform between the marker on the HoloLens and the optical centre of the HoloLens. See Figure 4.1 to clarify the position of these transforms relative to the user. As published in Elsdon and Demiris (2018*a*).

$$\mathbf{T}_{ho}\mathbf{P}_h = \mathbf{P}_r\mathbf{T}_{ro} \tag{4.9}$$

$$\mathbf{T}_{ho}, \mathbf{T}_{ro}, \mathbf{P}_h, \mathbf{P}_r \in \mathbb{R}^{4\times4} \tag{4.10}$$

It is important to note that given one observation of $\mathbf{P}_h$ and $\mathbf{P}_r$ finding the appropriate transforms is impossible as there are infinite solutions. Therefore any solution must solve over a time series of such observations. This was achieved by forming an optimisation that was solved numerically. The optimisation was initialised using the *SVD method* described in Section 4.2. When considering only the position of the poses as points to be aligned an initial estimate of $\mathbf{T}_{ho}$ is produced. This is equivalent of assuming that the marker is located at the optical centre of the HoloLens.

The following defines the marker and HoloLens poses in terms of the notation used in Section 4.2, trans() refers to taking only the translation element of the pose, and treating it as a point.

$$\mathbf{P}_a^i = \text{trans}(\mathbf{P}_h^i) \tag{4.11}$$

$$\mathbf{P}_b^i = \text{trans}(\mathbf{P}_r^i) \tag{4.12}$$

The result of interest from the calculation is:

$$\mathbf{T}_{ho} = \mathbf{T}_c \tag{4.13}$$

This provides a starting point for the transform $\mathbf{T}_{ho}$. Next, a numerical optimisation library (SciPy.Optimize[1]) is used to minimise Equation 4.14, which can be thought of as a circular set of transforms; if the two transforms being calculated are correct applying them in this order should give the identity matrix. The marker offset is initialised to the identity matrix, $\mathbf{T}_{ro} = \mathbf{I}$.

$$\min_{\mathbf{T}_{ro},\mathbf{T}_{ho}} \sum_{i=0}^{N-1} \left\| \mathbf{P}_r^i \mathbf{T}_{ro} {\mathbf{P}_h^i}^{-1} {\mathbf{T}_{ho}}^{-1} - \mathrm{I} \right\| \tag{4.14}$$

The transform ${\mathbf{T}_{ho}}^{-1}$ can now be applied to an object to be rendered and it will appear in the correct place within the motion capture arena when viewed through the HoloLens. It can be assumed that the marker offset transform ($\mathbf{T}_{ro}$) will remain static for each use of the system. Importantly however, moving around will cause the HoloLens' visual odometry to drift over time, or a loss of tracking could be slow to recover from. In this case, objects that were previously well registered to the real world environment will

1 https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html

drift away from their intended position. Therefore the offset between the HoloLens world and the motion capture world ($\mathbf{T}_{ho}$) must be continually recalculated.

If $\mathbf{T}_{ro}$ has been previously calculated, there is only one unknown, $\mathbf{T}_{ho}$, this can be solved using only one data point.

$$\mathbf{T}_{ho} = \mathbf{P}_h{}^{-1}\mathbf{P}_r\mathbf{T}_{ro} \tag{4.15}$$

This is not advised however as the noise in measuring $\mathbf{P}_r$, the marker in the motion capture arena, causes large jagged movements of all rendered items. This is especially a problem if the noise is primarily in the rotation of the marker. Therefore it is useful to use a method that smooths the result of this calculation. A simple infinite impulse response filter is adequate for this task.

$$\mathbf{m}_t = \text{trans}(\mathbf{T}_{ho}) \tag{4.16}$$

$$\mathbf{Q}_t = \text{quaternion}(\mathbf{T}_{ho}) \tag{4.17}$$

$$\hat{\mathbf{m}}_t = (1 - f)\hat{\mathbf{m}}_{t-1} + f\mathbf{m}_t \tag{4.18}$$

$$\hat{\mathbf{Q}}_t = \text{slerp}(\hat{\mathbf{Q}}_{t-1}, \mathbf{Q}_t, f) \tag{4.19}$$

$$\hat{\mathbf{T}}_{ho} \leftarrow \hat{\mathbf{m}}_t, \hat{\mathbf{Q}}_t \tag{4.20}$$

Where $\mathbf{m}_t$ represents the translation of the transform and $\mathbf{Q}_t$ represents the rotation quaternion taken from the calculation of the world offset transform ($\mathbf{T}_{ho}$) at the current time. f represents the filtering coefficient, nominally $f = 0.01$, smaller f leads to smoother filtering with more latency. Slerp is the spherical linear interpolation between the given quaternions, where f denotes the interpolation coefficient. The filtered version of the translation and rotation, $\hat{\mathbf{m}}_t$ and $\hat{\mathbf{Q}}_t$ can be combined back into a filtered version of the world offset transform $\hat{\mathbf{T}}_{ho}$.

4.3.2.2 *Validation*

A validation experiment was conducted to measure the error between the perceived location of an object in AR and where it was intended to be located, after using the *time series* calibration method.

From the user's perspective, the calibration procedure is walking for a short period around the arena while wearing the AR headset. To collect the time series for $\mathbf{P}_h$ and $\mathbf{P}_r$ their timestamps are matched and a hold-off of

5mm is used. The hold-off represents the minimum distance required to move from the previous sample before a new sample is collected, this is used such that the data collected would span some reasonable physical space. The time that the user must walk depends on the number of samples required, 500 samples takes around 4 seconds to collect.

Unfortunately due to the nature of this calibration it is difficult to demonstrate the accuracy of the alignment without experiencing it first hand, or having some alternative ground truth method. As such it is necessary to have a human in the testing loop, and the results will be somewhat dependant on their perceptions and care in completing the tasks.

The author of this work completed all the trials to ensure consistency. It can be assumed that the exact results would change with a different user, though this study is aimed to give an approximate value for accuracy of the calibration and how this changes with the length of the calibration period.

To validate the calibration the user must place a known model such that it lines up well with the rendering that they see through the augmented reality system. From each placement iteration the true location of the placed object, and the intended position of the hologram can be measured. Ideally the location that the user places the object exactly lines up with where the hologram was intended to be rendered, relative to the real world.

4.3.2.3 *Results*

The *time series method* described previously was tested for 10 iterations using different numbers of calibration samples. The results can be seen in Figure 4.3. After about 500 samples there is little increase in accuracy, and the best achieved accuracy is about 10mm position error and 20mRad angular offset (1.15 degrees). It is likely that the user's ability to accurately perceive the depth information and manoeuvre the real world tracked object sets a floor on the error.

4.4 CONCLUSION

A pair of methods for aligning the virtual world and the real world when using a motion capture system were presented. The *SVD method* is conceptually simple, though requires the interaction and care of the user to place reference points that align with features in the environment. The second method, the *time series method* takes no active input from the user, and relies on an additional marker being attached to the AR headset. The motion between this marker (as recorded by the motion capture system) and the reported posi-

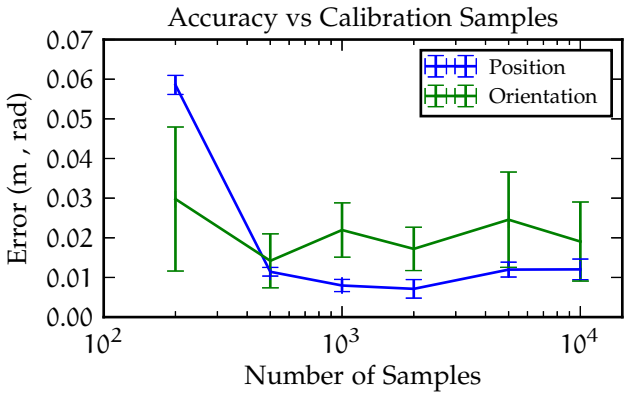


Figure 4.3: The error between the measured location of a motion tracked marker and intended position against length of calibration series. Error bars represent the standard deviation in the measurement. N=10 for each experiment. After 500 samples there is no discernible improvement. Graph was previously published in (Elsdon and Demiris, 2018*a*)

tion of the optical centre (as reported by the AR headset) are compared. This comparison leads to the discovery of the offset transform between the optical centre of the headset and the marker, as well as the offset transform between the AR headset origin location and the motion capture origin location. The latter was tested with an item alignment task, where the user aimed to align a physical object with a rendering of the same object. The difference between the intended position and the perceived location was found to be 1cm when the calibration procedure was performed with at least four seconds of data.

5

USER COMPLIANCE TO VISUALISATIONS

This chapter has been adapted from the original work published in:

Elsdon, J. and Demiris, Y. (2018), Augmented Reality Instructions for Shared Control of Hand-held Robotic Systems, *in* 'IEEE International Conference on Robotics and Automation, Workshop: Robotics in Virtual Reality'.

5.1 INTRODUCTION

In Chapter 3, two robots were introduced, one with a single DoF, and one with five. However, hand-held robots cannot achieve any objective without an collaboration with the user. This chapter presents an investigation into the extent to which a user can take orders from a hand-held robot system, especially when paired with an AR headset.

If the user can follow the orders given to them, then a *reduced degree of freedom* robot can trace a predefined trajectory with its end effector. Unfortunately this work finds that accuracy with which the user can follow orders is unlikely to be sufficient for most trajectory tracing tasks. There was on average a 63mm error in robot position and 0.18rad (10°) in orientation.

This work also provides insight into design considerations for *locally capable* robots. By measuring the ability of a user to comply with an instruction, an estimate on the size of the reachable area of a *locally capable* robot is established. However the size of reachable area required is likely to vary based on the form factor of the robot, complexity of trajectory and scale of movement required for the task. For example a reachable area defined by a 4mm $\times$ 4mm cylinder is deemed to be appropriate for a pen-like form factor, when performing detailed tasks, such as membrane peeling, in eye surgery (Yang et al., 2015).

The effectiveness of augmented reality guidance is also compared to a situation where the user has full knowledge of the trajectory expected of them, detailed visualisations to guide them. It was found, perhaps unsurprisingly, that performance was significantly increased when the users had access to additional visualisations to guide them. AR guidance improved the accuracy

of speed regulation, position error and orientation error by 19.3%, 15.5% and 39.2%, respectively.

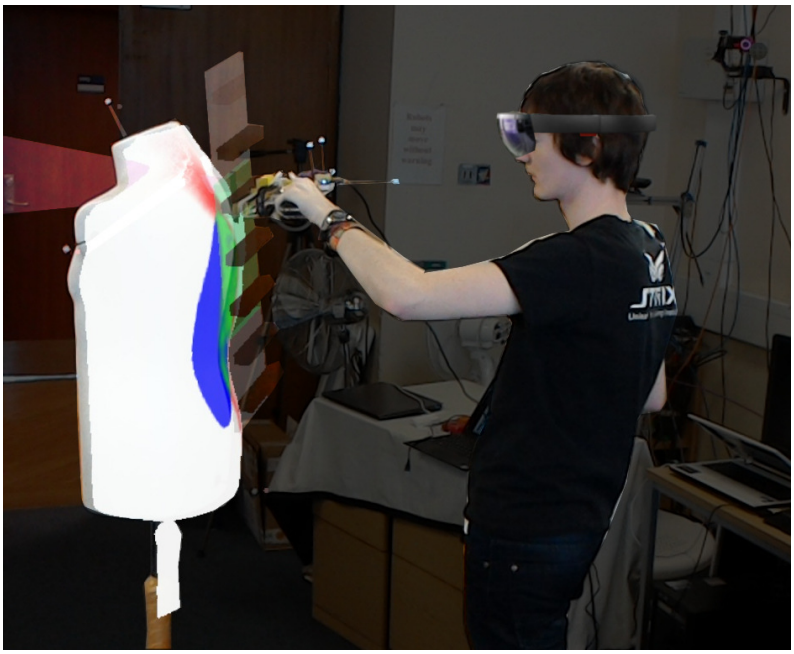


Figure 5.1: This figure shows a 3rd person view of a user performing a trajectory using the detailed version of the visualisation. The hand-held robot and the mannequin are motion tracked by an infra-red camera system. This image was taken with the Microsoft Hololens, and would not normally be visible to a 3rd party. The Hololens has been superimposed over the image and foreground elements are highlighted.

5.2 EXPERIMENTAL SETUP

The experiment consists of eight trajectories that the users were asked to complete with the hand-held robot, once with a detailed visualisation and once with a basic one. The visualisations are shown in Figure 5.2.

The basic visualisation was designed to act only as a prompt to the user, such that they are simply aware of which direction their next trajectory should be. On the other hand, the detailed visualisation shows the user the plane they are supposed to sweep with the gantry of the hand-held robot, where they should press and release the trigger, and the speed at which they should be travelling.

For all experiments, the plane that they should sweep is the same distance from the mannequin. The speed was also the same for all trials, set at 30cm/s. This consistency was designed such that the user can have a full understand-

ing of what trajectory they are expected to complete, even when there is only the basic visualisation indicating direction of the path to undertake.

The mannequin is both physically present and rendered in the AR system. Physical presence of the mannequin was intended to assists the users depth perception. Rendering the mannequin allows the experimenter to ask "is the mannequin rendering aligned with the real mannequin?", which provides confirmation that the system is calibrated correctly for the user.

Participants completed eight trajectories, repeating each twice, alternating between the detailed and basic visualisations. Half of the participants undertook each trajectory with the detailed visualisation first, the other half with the basic one. Interleaving the two types of trials is to help ensure that users have a good understanding of the trajectory parameters (speed, height above the mannequin etc) even when they are not shown in the visualisations.

The trajectory of the spraying robot is captured by an infra-red camera-based motion capture system. The mannequin is also tracked so that the experimental area can be moved conveniently, though the mannequin was not moved during any user trials.

To ensure the visualisations are located accurately, the visualisation alignment method that is described in Section 4.3.1 was used.

5.3 RESULTS

A total of eight participants (two female and 6 male) were recruited for this study, all of whom were already familiar with the hand-held system and augmented reality headset. There are two categories of results of interest: relative quality measures and absolute quality measures. Relative quality measures do not reference the set trajectory, and absolute ones do. This distinction is important because it would be unreasonable to expect users to match parameters of a trajectory without being shown them, as is the case with the basic visualisation. However the movement should meet the general criteria that was asked of them, namely, the path should be straight, at a constant speed, and the robot should be orthogonal to the direction of travel at all times. These general criteria match the assumptions that the algorithm presented in Chapter 6 uses. All of the results are summarised in Table 5.1.

5.3.0.1 *Position Accuracy*

The position accuracy was measured by taking the measured position of the robot and measuring the perpendicular distance to the trajectory. This calculation is shown in Equation 5.1, where D is the distance from the line, S is the

(a) Detailed Visualisation. (b) Basic Visualisation.

Figure 5.2: This figure shows both the detailed and basic visualisations used in the experiments. The detailed visualisation has bars which move along the graphic at the speed that the user is expected to emulate. The user should aim to sweep the gantry of the hand-held spraying robot across the green section of the visualisation, whilst maintaining orthogonality of the robot to the trajectory. The user is expected to do all of the same things with the basic visualisation. The basic visualisation is only to help the user remember the direction they are expected to move the robot along. The blue area has no bearing on this work.

position vector of the start of the line, E the end, and P is the point under consideration. The absolute value of the distance was taken and averaged over the trajectory to arrive at the mean error from the trajectory. Participants performed better with the detailed visualisation with an average error of 0.0636m compared to 0.0720m (p=0.12).

However when inspecting the error from the best-fit straight line of the user's trajectory, there is no difference between the visualisation types, both diverting from the best fit line by an average of 6.4mm. This shows that the visualisation is not helping users move along a straight trajectory, though it does help them stay in the vicinity of the target trajectory.

$$D = \frac{\left\| \vec{SE} \times \vec{PS} \right\|}{\left\| \vec{SE} \right\|} \tag{5.1}$$

Table 5.1: A summary of the metrics analysed, their standard deviations, and the p-value when considering the proposition "the error is lower in the case of the detailed visualisation".

Error Metric	Units	Detailed	SD	basic	SD	p-value
Abs. Position	mm	63.7	28.8	72.0	30.8	0.122
Rel. Position	mm	6.42	308	6.43	3.72	0.992
Trajectory Speed	mm/s	73.2	38.3	88.8	39.2	0.0255
Instant Speed	mm/s	157	51.6	178	62.4	0.0418
Abs. Orientation	rad	0.183	0.111	0.238	0.156	0.0265
Rel. Orientation	rad	0.314	0.129	0.350	0.146	0.136

5.3.1 *Speed Regulation*

Users were required to move the robot at 0.3m/s for all trials. There are two metrics that are informative here, error in average speed over the whole trajectory and instantaneous speed error during the trajectory.

The average speed of the robot ($S_{trajectory}$) was significantly more accurate with the detailed visualisation, 0.073m/s error, compared to the basic visualisation, 0.089m/s error ($p = 0.025$, calculated with the students' t-test). During the movement it was possible to see some variation in the speed as users were trying to match the moving bars in the visualisation. Looking at the average instantaneous error during the trajectory ($S_{instant}$), the detailed visualisation performs better with 0.16m/s error compared to 0.18m/s ($p = 0.042$) for the basic counterpart. The fact that the instantaneous speed error is significantly larger shows that the users are better at estimating the speed over the whole trajectory, rather than keeping a correct speed at any given moment. The methods of calculating the average speed error ($S_{trajectory}$) and instantaneous speed error ($S_{instant}$) are shown in Equation 5.2 and 5.3, respectively.

$$S_{trajectory} = \frac{1}{N}\left(\sum_{i=0}^{N} s_i\right) - s_{target} \tag{5.2}$$

$$S_{instant} = \frac{1}{N}\sum_{i=0}^{N} \left\| s_i - s_{target} \right\| \tag{5.3}$$

5.3.2 *Orientation Accuracy*

The users were asked to keep the robot orthogonal to the direction of movement at all times, which was shown in both visualisation categories. Therefore, there are two metrics to measure the performance of the users' align-

ment accuracy, the relative orientation of the robot with respect to its movement direction, and the alignment with the requested orientation. In both of these metrics the detailed visualisation outperforms the basic visualisation, 0.31rad vs 0.35rad ($p = 0.13$) for the relative alignment, and 0.18rad vs 0.23rad ($p = 0.026$) for the absolute alignment.

5.4 CONCLUSION

It can be seen that the more detailed visualisation allowed the users to perform better across all evaluated metrics. This is not a particularly surprising result, given that users had access to more information from the detailed visualisation. However, demonstrating the performance of the chosen visualisation over that of a lesser visualisation was not the aim of this preliminary work. Instead, the central focus has been to demonstrate a baseline for user movements whilst holding a hand-held robot with effectively no guidance, and that even a basic visualisation displaying key information aids users rather than hinders them.

It is hoped that a designer of a similar system can use the data provided here to allow them to design assistive algorithms that are using assumptions about the user's ability to comply with the instructions given. For example, an active head on such a spraying robot should be able to account for roughly 63mm of deviation from the planned path and an orientation error of 0.18rad (10 degrees), when the user has detailed information provided via an AR headset. However this would increase to 72mm position error and orientation error of 0.238rad (13.6 degrees) error if provided with less informative spatial cues.

The low accuracy of users following AR visualisations has important ramifications for the two sub-categories of hand-held robots highlighted in this thesis. Even when instructions are very simple, and delivered in a repetitive way, the tracking accuracy is not particularly good. With *reduced degree of freedom* robots this inaccuracy must couple to the end effector, at least in the DoF that are not controllable by the robot. Therefore a future designer should choose the DoFs that are actuated on the robot to allow the error coupled to the end effector to be acceptable for the task at hand. For example, a designer may be happy to accept a 0.18rad deviation in rotation from the planned trajectory, though not a 63mm deviation from the trajectory along the linear axes. In this case they should use a three DoF *reduced degree of freedom* robot that is actuated with the three DoF.

For the *locally capable* category of robots, a designer should ensure that the range of motion of the end effector can encompass at least 63mm in the linear directions and 0.18rad in the rotation axes. This was used as a guideline for the design of the five axis robot presented in Section 3.3. However, this bound is likely to be conservative in such a situation, as this is ignoring the active feedback loop that is formed between the user and a *locally capable* robot. The robot would *gesture* towards the set point that it is tracking, and the user could adjust their own trajectory to ensure the robot can continue reaching its set point.

Most of the remaining error in tracing trajectories with the robot is likely due to human difficulty in perceiving depth, as well as obstruction of the real world by overlaid visualisations. Also it should be noted that the simplicity trajectories requested of the user could have lead to low investment from the users to perform the task to the best of their ability, as some users approached the trajectories quite casually. Future work could attempt to provide visualised feedback to the user regarding their performance, helping to emphasise any mistakes they perform. Further, assistive algorithms for hand-held robots can be improved with realistic knowledge of the capability of the human user to comply with trajectory requests.

6

ONLINE PLANNING IN A REDUCED DEGREE OF FREEDOM HAND-HELD ROBOT

This chapter has been produced from the original work published in the following locations:

Elsdon, J., and Demiris, Y. (2017), Assisted Painting of 3D Structures Using Shared Control with Under-actuated Robots, *in* 'IEEE International Conference on Robotics and Automation', pp. 4891–4897.

6.1 INTRODUCTION

This Chapter and Chapter 7 explore the concept of *reduced degrees of freedom* hand-held robots. A *reduced degrees of freedom* hand-held robot is one that must rely on the human user at all times for precise motion of the end effector. This is because these robots have less DoF than the task requires, and the access to these missing degrees of freedom is supplied by the motion of the human.

If a trajectory for the end effector is required, one or more of the DoF must be actuated exclusively by the user, as such the error in tracking the trajectory is in part tied to the accuracy with which the user can move the robot through the predefined trajectory. As shown in Chapter 5, even when very simple trajectories are required of the user, large tracking errors are likely. For example, even with a detailed visualisation guiding them, the absolute positioning error average was 63.7mm.

The reason why a *reduced degrees of freedom* hand-held robot robot is worth of study is the fact that they are by their nature, simpler, lighter and cheaper. Weight particularly can be overbearing in the case of a handheld robot. These benefits are traded for the fact that the robot and user must be more tightly coupled.

Whilst it is not reasonable for a user to track a particular trajectory with the end effector, not all tasks require an exact trajectory, for example painting an area with a particular paint distribution. In pursuit of the final paint dis-

tribution the particular trajectories used to realise it are not fundamentally important.

This chapter describes an algorithm that is designed to solve this painting problem using the robot described in Section 3.2.1. The goal of the algorithm is to place as much paint in the correct locations within a short time horizon. It does this by considering all paths available in this horizon, choosing the one that is likely to allow the most paint deposition. This plan is then partially implemented on the robot, and an update to the task state is made to represent the paint deposited in this implemented section. Then a new path is planned, it continues in this receding horizon manner. There are no strong constraints on the surface to be painted, as input to the system is an arbitrary triangulated model of the target object. A primary contribution of this algorithm is how all solutions in a large search space can be considered in a tractable time. The algorithm is verified to perform well, choosing paths that are close to optimal within the decision time of 100ms, taking only 32ms.

The inputs to the system are:

- A map of the required density of the paint, in the form of a texture map.
- A 3D triangulated model of the target object to be sprayed.
- The pose of the robot relative to the target.

The outputs of the algorithm are:

- A position and velocity command for the end effector.
- A set of times to activate and deactivate the spray nozzle during this movement.

6.2 METHOD

It is intended that the user will sweep the gantry of the linear slide such that the robot travels in a direction orthogonal to both the direction of the linear slide and the direction of that the nozzle points. This can be visualised more easily by looking at Figure 5.1 in Chapter 5. As such over a short time horizon the end effector will have access to a two dimensional space that is defined by its own gantry and the direction of movement. Planning a path through this space is the purpose of this algorithm, this space is depicted in Figure 6.1.

There are two main stages to the software, candidate path selection and path simulation. The function of the candidate path selection is to suggest a path for the future movement that is likely to provide rich opportunities for the nozzle to dispense paint. The path simulation is then used to both calculate when the nozzle should dispense paint within this path and to update the internal representation of the state of the painting. Only the very beginning of the path is simulated and implemented on the robot. The purpose of the candidate path extending further into the future than will be implemented on a particular planning cycle is to ensure that the algorithm can pick a path that is closer to the global optimum, rather than simply operating on a greedy principle of only inspecting parts of the space immediately available to it. If this was not implemented areas of a small positive value could make the end effector move to a position that then forces it to miss an area of much higher value that could have been accessed with a different path.

A flow chart of the method is presented in Figure 6.2 and a pseudo-code listing using verbose variable names is presented in Algorithm 1 (with subroutines in Algorithm 2,3,4 and 5).

6.2.1 *Candidate Path Selection*

Simulating paths fully as demonstrated in Section 6.2.6 is an expensive operation; it cannot be parallelised across time, as future paint deposits will rely on past paint deposits. Therefore using such a simulation to evaluate many paths, then picking the best one is not a tenable solution if the allowed solution space is large. Some assumptions must be made to accelerate the search for good candidate paths. Firstly it is assumed that the operator is moving the robot primarily perpendicularly to the gantry direction and the spray direction, as to sweep the largest possible area. This helps minimise the situation where the head could revisit the same physical location at a later time, thus reducing the dependence of later paint deposits on previous paint deposits. Small discrepancies in the alignment of the robot can be handled by the algorithm, as the current estimate of the velocity, including misalignment is used to generate the grid locations shown in Figure 6.1. In the case of misalignment the grid would be representing a skewed version of the future movements. It is also assumed that the user is trying to keep the robot's velocity smooth at all times, thus the space that is swept can be well defined by the gantry length and the distance that the user is anticipated to travel over a fixed time period. Small variations in speed are also handled well by the method, as the most important segments of the generated path

Algorithm 1: This is a high level outline of the algorithm presented in Chapter 7, See Algorithm 2,3,4,5 for clarification of the subroutines. The "solveDjikstra" applies Djikstra's algorithm to find path with the highest total score. The "sendToRobot" function transmits the the variables used for actuation to the robot.

```
while True do
    Nodes nodes = initGraphNodes(); // see Algorithm 2
    Edges edges = initGraphEdges();// see Algorithm 3
    calculateEdgeWeights(nodes, edges, robot);// see Algorithm 4
    Path path = solveDjikstra(nodes, edges);
    Float score, Vector⟨Time⟩ valveOn, Vector⟨Time⟩ valveOff =
     simulateFirstSegment(path);// see Algorithm 5
    sendToRobot(valveOn, valveOff, path);
```

Algorithm 2: The algorithm for populating the grid of nodes. Not all nodes will be accessible in the final graph structure. The grid dimensions will be $gantryQuanta \times timeQuanta \times divisions$ in size, which can be seen more clearly in Figure 6.3 and Figure 6.1, though only used nodes are shown in those figures.

```
Function initGraphNodes():
    for y ← 0 to gantryQuanta do
        for x ← 0 to timeQuanta do
            densityTexture(x, y) = samplePaint(x, y);
            for k ← 0 to divisions do // Init all nodes
                nodes.add(x, y, k);
    return nodes;
```

Algorithm 3: Algorithm for initialising the valid edges between nodes in the graph structure. "edgeValid(x, y, m)" if a function that checks whether the node at position (x, y, m) is reachable from the original starting node, and allowable change in speed.

```
Function initGraphEdges() → Edges:
    for y ← 0 to gantryQuanta do // Build Graph on CPU
        for x ← 0 to timeQuanta do
            for k ← 0 to divisions do
                for m ← −m/2 to m/2 do
                    if edgeValid(x + 1, y + m, m) then
                        startNode = nodes.get(x, y, k);
                        if x = horizon then
                            endNode = terminalNode;
                        else
                            endNode = nodes.get(x + 1, y + m, m);
                        edges.addEdge(startNode, endNode);
    return edges;
```

Algorithm 4: This algorithm assigns weights to edges in the graph structure. The weight is the sum of the spray density that would arrive at this location, as long as this spray would not overdose that region.

```
Function
 calculateEdgeWeights(Nodes nodes, Edges edges, Robot robot):
    for edge in edges do // Find edge weights on GPU
        nozVel = robot.velocity + edge.gantryVel;
        sprayDensity = robot.sprayFlux / nozVel;
        for step in steps do
            sprayNeeded = densityTexture(step);
            if sprayDensity < sprayNeeded then
                score += sprayDensity;
        edge.score = score;
    return;
```

Algorithm 5: Algorithm for simulating the first line segment in the candidate path and updating the current representation of paint coverage in memory. The result is a final score for painting within the first segment, and the times that the spray nozzle valve should be turned on and off to achieve the calculated paint distribution. The steps size in this simulation is usually much finer than the grid size in Algorithm 2 or the step size in Algorithm 4. "sprayTexture" is the texture that represents the current paint distribution and "targetTexture" is the texture representing the ideal paint distribution. The score is the summation of paint that is correctly placed combined with a negative score for paint that is overdosing a region. This punishment factor can be made larger or smaller with "PunishFactor".

```
Function simulateFirstSegment(Path path)
  → (Float, Vector⟨Time⟩, Vector⟨Time⟩):
    for fineStep in path.firstSegment do // Find score and update
     coverage on GPU
      sprayQuantity = robot.sprayFlux × fineStep.interval;
      sprayQuantaNum = sprayQuantity / dropQuantity;
      sprayQuantaSet = sprayDistribution(sprayQuantaNum);
      for sprayQuanta in sprayQuantaSet do
        // duplicates are accumilted into one sprayQuanta,
        // here they are treated separately for clarity
        currentPaint =
         sprayTexture.get(sprayQuanta.intersection(targetModel));
        targetPaint =
         targetTexture.get(sprayQuanta.intersection(targetModel));
        if currentPaint + sprayQuanta.value < targetPaint then
          score += sprayQuanta.value;
        else
          score += PunishFactor × (currentPaint +
           sprayQuanta.value − targetPaint);
      if score > 0 then
        sprayTexture.update(sprayQuantaSet);
        valveOn.pushBack(fineStep.time);
      else
        valveOff.pushBack(fineStep.time);
    return (score, valveOn, valveOff);
```

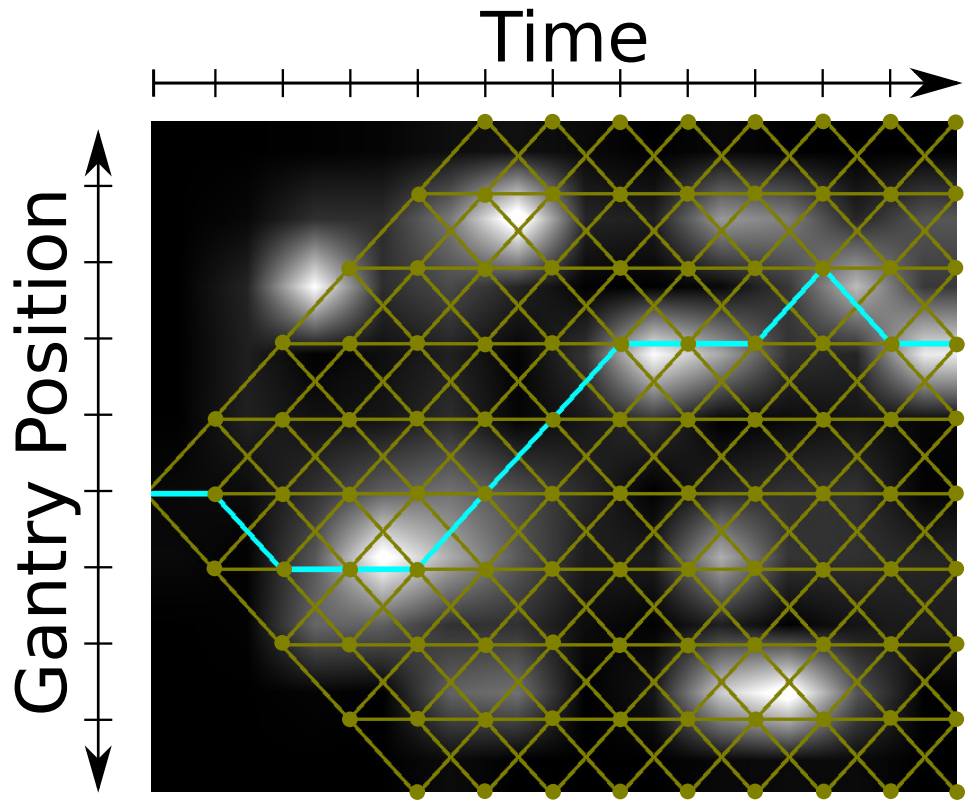


Figure 6.1: The path that the nozzle takes is based on a grid of positions with line segments connecting them. The path is chosen such that it maximises the amount of paint that can be dispensed into the target areas. The white sections of this image are in need of paint. The cyan line is the chosen path, yellow represents all possible paths. The time intervals shown are typically 0.1 seconds, this is the time period at which new commands are sent to the robot. The gantry positions evenly spread along the length of the gantry, typically 1cm apart depending on desired granularity of the algorithm.

are the ones closest to the present time as this is the section of the path that will be implemented the soonest, later parts of the path will be recalculated in future iterations of the receding horizon, which will correct for the lack of consistent velocity. Primarily the user should act in a smooth manner.

This software module has 4 stages to its implementation:

1. Calculating the required paint density across the space defined by the spray nozzle's location on the gantry and time.
2. Producing a graph structure that can account for mechanical feasibility.
3. Calculating the maximum benefit of line segments in this graph.

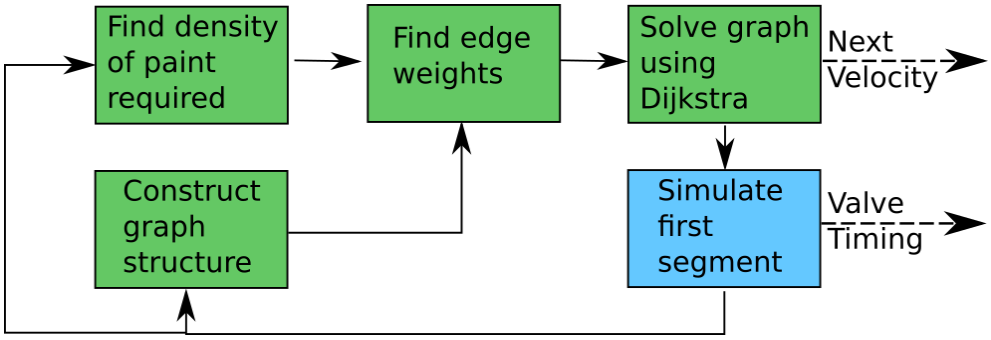


Figure 6.2: Green sections are part of the candidate path selection. The blue process is the path simulation described in Section 6.2.6. The dashed arrows on the right indicate outputs to the hardware described in Section 6.3.2.

4. Solving for the path most likely to place the maximum amount of paint in the correct place using Dijkstra's algorithm Dijkstra (1959).

These four stages are highlighted in green in Figure 6.2.

6.2.2 *Calculating Required Paint Density*

The spray painting process has a number of variables that will affect the rate at which the paint covers a particular area, for example, if the nozzle is at a large distance from the object being painted the paint per second arriving at a given area will be less than if the nozzle were close. Other considerations include the distribution pattern of the spray nozzle, obliqueness to the surface etc. These effects should be accounted for and all simplified into one variable: at this location on the gantry and this point in time, how much paint is needed. This quantity is sampled at regular intervals in a grid across time and nozzle position in the manner defined in Equation 6.1. At each location 1024 rays are cast into the scene and each ray returns the difference between the current paint coverage, c, and the target coverage, p. This is multiplied by the distance squared, d^2. This operation provides the required flux of paint at unit distance for this ray. The direction of the rays is decided by the distribution pattern of the airbrush nozzle. The mean of this value is now representative of the average paint flux at unit distance at this location on the gantry, g, at a given time, t. Repeating this operation in a grid across many gantry positions and time instants produces a map of required paint density in all positions that the nozzle will have access to over the allotted time horizon. This can be visualised as in Figure 6.1, where the white sections are requiring a large paint density, and the black areas require little or no paint density. This section does not make any strong assumptions about

the geometry of the object to be painted, as long as the model is a good approximation of the real object to be painted and is triangulated reasonably, avoiding very thin triangles and unwanted gaps between triangles. This map of required paint density is used as the basis for generating the edge weights of a directed graph in Section 6.2.4.

$$d(g,t) = \frac{1}{\text{rays}} \sum_{i=0}^{i=\text{rays}} d^2(c_i - p_i) \tag{6.1}$$

6.2.3 *Building Graph Structure*

There are an infinite number of paths that the nozzle could take between the start and the time horizon, so for the purposes of this work the movements will be discretised. This discretisation is defined by allowing the head to only be in discrete locations along the gantry at given intervals of time. This forms a grid on the space defined in Section 6.2.2. The nozzle can then move between these locations in line segments. At each grid location there is a predefined number of alternative routes going forward, this will be called the number of *divisions*. In order to account for the fact that the nozzle should not be allowed an infeasible change in speed the graph structure needs to account for the incoming speed to each grid location and only allow the nozzle to leave the grid location at an allowable velocity. This is encoded by having multiple nodes in the graph at each grid location, one for each possible incoming speed. These nodes are not connected to nodes that would be infeasible given the incoming speed it represents. For example, if the nozzle was moving at full speed along the gantry in one direction it may be infeasible for it to move full speed in the other direction during the next time period. The allowable change of speed at each juncture is an integer called *speed change*, this value indicated the number of discretised speed levels the end effector is allowed change by at each juncture.

Many of the paths from one node to the next will represent the same physical movement, and therefore the weight assigned to these duplicates should be the same. Hence the cost for all possible edges between nodes is calculated, not accounting for feasibility. The edges on the graph then inherit the costs from this simplified graph. This encoding of feasibility and the inheritance of edge costs is described fully in Figure 6.3.

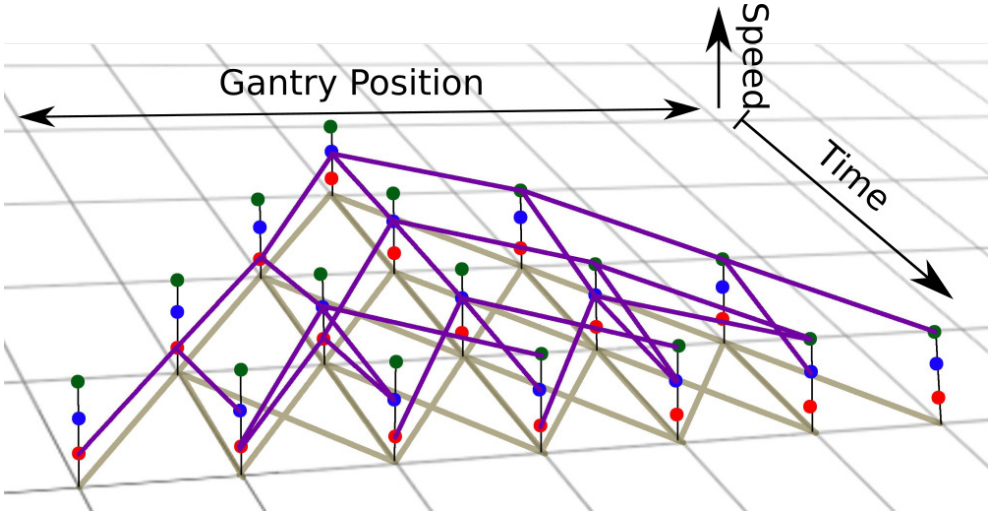


Figure 6.3: A visualisation of the graph structure. The stacked nodes encode the speed incoming to the node, blue is 0, red is +1, green is −1. The purple lines indicate the edges of the graph, notice they will not allow a transition between +1 and −1 speed or the reverse. In a realistically proportioned graph there is a parameter *speed change* which defines the allowable change in speed at each node. The shadowed lines at the bottom of the image indicate the edges of the basic tree structure. The purple edges will inherit their edge costs from this base graph, notice there is often more than one edge representing the same physical movement.

6.2.4 *Calculating Benefit of a Sub-movement*

Each edge in the graph described in Section 6.2.3 represents a physical movement of the end effector. This can be assumed to be a line segment over the surface of the model to be painted. This line segment, and the graph edge that represents it need to be allocated an edge weight. This weight will represent an estimated value of the spraying nozzle having access to this region. This weight will be low if the area has lots of painting still left to do, and it will be high if the area is already fully painted, or didn't need paint in the first place.

Given the robot's velocity perpendicular to the gantry, v, and the gradient of the line segment, $\frac{d}{dt}(g)$, we can calculate the nozzle velocity, n.

$$n = \sqrt{(\frac{d}{dt}(g))^2 + v^2} \tag{6.2}$$

Combining this with the flow rate of the nozzle, flow, and the distribution pattern of the nozzle (simplified to a cross section area at unit distance, A) the paint flux at a unit distance, f, can be calculated for this line segment.

$$f = \frac{\text{flow}}{A \cdot n} \tag{6.3}$$

Comparing this paint flux with a number of samples (from the required paint flux map) along this line segment, it can be evaluated whether this is an appropriate flux value for this location.

If the flux is less than that required then the score, w_i of this sample on the line can be positive. If it is more than is required the score is zero, this is because if the nozzle were to perform this movement it would overdose this region with paint. The total score of the line segment, W, is the summation of all the samples along its length. Typically 32 samples are used along each line segment.

$$w_i = \begin{cases} d(g,t) & \text{if } f < d(g,t) \\ 0 & \text{if } f > d(g,t) \end{cases} \tag{6.4}$$

$$W = \frac{1}{\text{samples}} \sum_{i=0}^{\text{samples}} w_i \tag{6.5}$$

The positive score given to a sample effects how the system will behave, the optimum score to assign is the calculated flux for this line segment. This will maximise the amount of time that the nozzle can be switched on over the whole path. Alternatively, using the paint required at this position as a score prioritises the robot to visit areas that are more in need of paint. Using the target paint quantity as the score was found to achieves better results. The paths chosen have more margin for error compared to picking paths where the calculated flux for the path is very close to the approximated required flux. When there is a small margin for error, the discrepancy between the required paint density approximation and the full simulation described in Section 6.2.6 can lead to the nozzle being left off to avoid an overdose condition. In this case there will be a large discrepancy between the score expected and the actual calculated score for a line segment, meaning that the path found using this method will be far from optimal.

This method is used to assign a weight to each element of the graph described in Section 6.2.3.

6.2.5 *Solving for Best Path*

Now that a graph as defined in Section 6.2.3 is constructed, and its edge weights have been assigned based on the function described in Section 6.2.4.

Using Dijkstra's algorithm the path from the start node to the time horizon that maximises score can be found. This should give the route that maximises the opportunity for the nozzle to apply paint to the correct locations. The path selected should be such that it chooses swift diagonal movements through areas that require a sparse covering in paint, for areas that require high densities it will chose a path that maximises the amount of time that the nozzle is within the region, as to maximise the amount of paint distributed over that time period. Solving by Dijkstra's algorithm was implemented using the Boost Graph Library, and this is the only computationally significant part of the algorithm that takes place on the CPU.

6.2.6 *Simulation of the Selected Path*

All of the above calculations rely on the assumption that paint dispensed earlier in the movement will not affect paint dispensed later. For maintaining an accurate internal representation of the state of the paint coverage this is not acceptable. Therefore a simulation that can take account of previous paint coverage must be used. This only needs to be applied to the first section of the candidate path, as this is the only section that will be implemented on the robot, therefore this high computational cost is acceptable.

This simulation is achieved by modelling the paint droplets as rays and casting them into the geometry, these rays are used to lookup the corresponding pixel in the texture map, which holds the current state of paint coverage on the object, a similar texture holds the target paint coverage. At each time step r rays are cast, each represents the appropriate amount of paint based on the time step, flow rate and rays per time step. It is often the case that multiple rays within each casting operation will point to the same pixel in the paint state texture map, therefore it is important to use atomic operations to ensure they are totalled correctly on the GPU. After the casting operation a list of pixels in the texture map that have received paint is produced, the quantity of which is defined as q. q is compared to the amount of paint required at this location to reach the target. If the amount of paint cast to this location (q) added to the current paint at the location (h) is less than or equal to the target amount required, p, the total amount of paint is recorded as the score, s. If there is an overdose condition the excess is punished by a punishment factor, P.

$$s_i(t) = \begin{cases} q_i & \text{if } q_i + h_i \leqslant p_i \\ (q_i + h_i - p_i)P & \text{if } q_i + h_i > p_i \end{cases} \tag{6.6}$$

$$S = \sum_{i=0}^{r-1} s_i \tag{6.7}$$

At each time step along the path an aggregate of the score of all the rays cast is calculated; S. If this total is greater than zero the paint distribution at this time instant is considered a success. In this case all of the paint quantities will be added to their respective pixel in the texture map. If the *score* is negative, spraying at this location is detrimental to the painting of the object, and the paint quantities are not written back to the texture. The series of whether each time step was beneficial to the painting task is kept, this is used to produce timings for the valve that controls the paint flow on the robot.

6.3 VALIDATION AND COMPARISON

To demonstrate the efficacy of the proposed method two sets of experiments are presented. The results that were gathered from simulation are presented in Section 6.3.1, those gathered from hardware tests are presented in Section 6.3.2.

6.3.1 *Experiments in Simulation*

An exhaustive evaluation of all possible paths in a particular scenario can be used to give a robust bench mark for the quality of a selected path. The path that is generated can be given a rank out of possible paths, with 1 being the best rank. The ranking is ordered based on the quantity of liquid that would be sprayed if the full simulation described in Section 6.2.6 is applied along the entire candidate path. In order to compare the proposed method with a baseline method a greedy path planner was developed. The greedy algorithm picks the most valuable linear movement over the next time period until it reaches the time horizon. The evaluation of the line segments will be done using the simulation outlined in Section 6.2.6.

Firstly, both the proposed method and the greedy method are presented with the same 32 scenarios, consisting of a number of patches, at various different rotations. Each method then generates a path for the airbrush nozzle

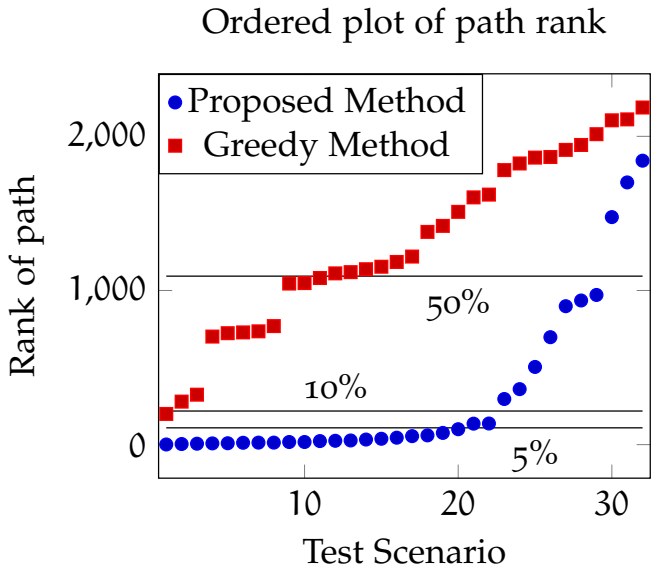


Figure 6.4: For each of the 32 trials, the path produced by each algorithm was given a rank for quantity of ink placed out of all possible paths (calculated exhaustively). These ranks were then ordered from best to worst for clarity. We can see that the proposed method picks a good solution for the majority of trials, most in the top 5% (the average being 15%), though the greedy method does not seem to outperform a random pick as it returns paths with an average rank of 53%.

to take, these are then given a rank against all possible paths. Both methods select paths from the line segments in a tree structure as shown in Figure 6.1 such that both algorithms obey the same feasibility constraints. The results can be seen in Figure 6.4. The proposed method averages a rank of 328 out of 2186 (15%), though the majority of the paths are in the top 5% of all paths. The outliers seem to be in scenarios where there is very little paint to be placed within the time horizon. The path chosen then maximises its time in areas of sparse paint requirement, though due to the approximate nature of the paint density estimate produced in Section 6.2.2 this marginally fails to pass the test of whether the valve should be on presented in Section 6.2.6.

The greedy method on average produced a path of 1302 out of 2186, which is worse than chance. This is because the greedy method has no capability to bias the path towards fruitful areas that occur outside of the next time period.

Computational efficiency is important because the algorithm will be used to plan paths on a real time spraying robot. The path must be planned whilst the first segment of the previous plan is being acted upon by the robot. Therefore if the run time of the algorithm is shorter the frequency of commands

Table 6.1: Comparison or run time between the proposed method and a simple greedy algorithm. Data taken over 32 test scenarios. The tests all were using a horizon of 7, with a choice of 3 directions at each intersection.

Method	**Mean Time (ms)**	**Standard Deviation (ms)**
Proposed Method	32.6	3.27
Greedy Method	116.84	3.20

sent to the robot can be increased or the number of path options considered can be increased.

The execution time of the proposed method was compared against the greedy implementation. The times for the proposed method include the path generation and one segment of simulation as described in Section 6.2.6. For the greedy algorithm the simulation is implicit in the path generation, therefore for this method only the path generation time is included. As shown in Table 6.1 the proposed method has a significantly shorter run time. In this set of scenarios the path generation was over a horizon of 7 decision intervals each representing 0.1s and there were 3 divisions after each time period. The proposed method would scale very well into larger solution spaces defined by: number of time periods before the time horizon; quantisations in the gantry direction and the number divisions after each time period. This is because the most computationally expensive part of the method (aside from the dense simulation of the first segment) is generating the density map, which is independent of the number of paths at each intersection and scales linearly with gantry quanta and horizon length. In contrast any algorithm that uses the full evaluation for line segments iteratively, such as the greedy algorithm, will scale very badly into bigger solution spaces.

6.3.2 *Experiment with Hardware*

To demonstrate the algorithm outlined above a preliminary test was conducted where the robot shown in Figure 6.6 should place paint in a specified area. The robot body is manoeuvred by the user, moving the robot orthogonally to its sliding axis and the direction of the nozzle. The user can also rotate the object, by rotating the base that it stands on. The algorithm plans the next move at a rate of 10Hz, whilst considering the next 1 second of potential paths available to it. A snapshot of the robots measured velocity and orientation is used for extrapolation into the future. The robot measures its current location and velocity using a camera mounted to the robot, this

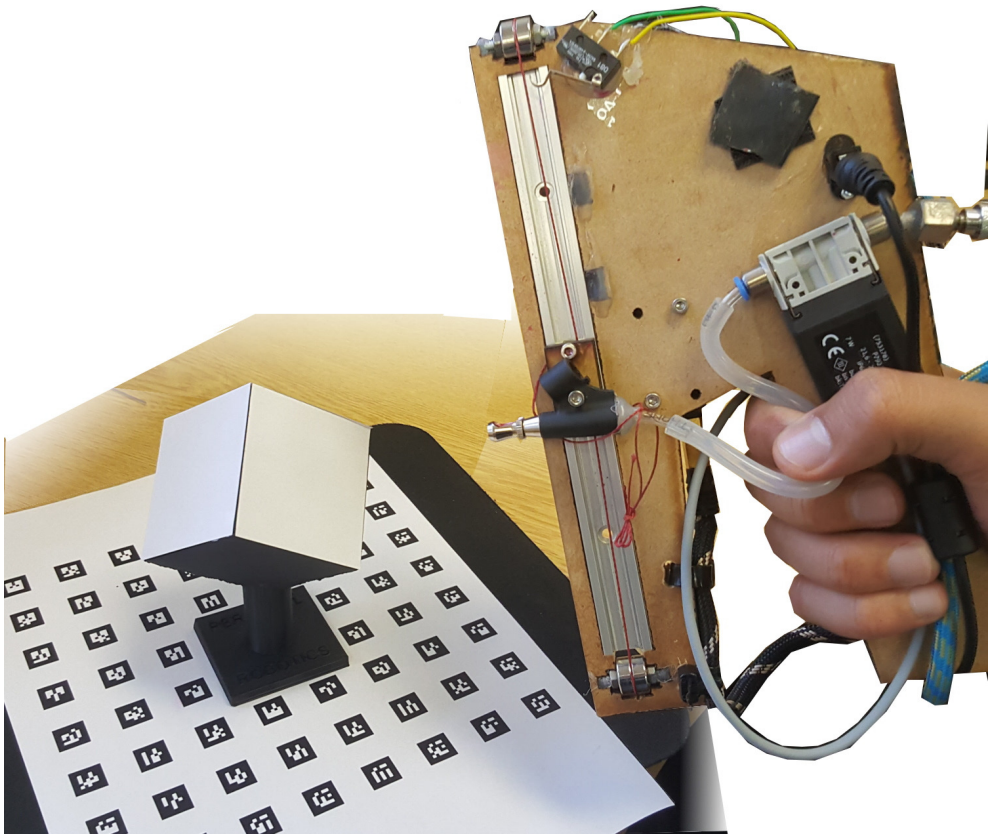


Figure 6.5: The experimental setup. On the right is the robot, it has a single axis of movement which can move the airbrush head up and down. On the left is the object to be painted, it is located on top of an array of visual markers, which are observed by a camera mounted on the robot, see Figure 3.1a for a more detailed view of the hardware.

tracks small markers that are located below the item to be painted, seen in Figure 6.5.

The camera output is fed back to a desktop computer with a mid-range GPU (NVIDIA 960) at 100Hz. The robot is also supplied with a compressed air hose for the airbrush and has a USB connected micro-controller to manage the movement of the airbrush head. The software architecture is using ROS (Robotic Operating System)Quigley et al. (2009) for communication to the robot and between nodes on the desktop. The path calculation is accelerated using OpenCL.

The aim for this demonstration was to paint three small circles on the faces of the 3D printed object shown in Figure 6.7b. The faces were covered in pieces of paper for reusability. Scans of the paper are presented in Figure 6.7c, the green overlays represent the ideal target locations. Figure 6.7d shows the state of the simulation after the demonstration, white sections

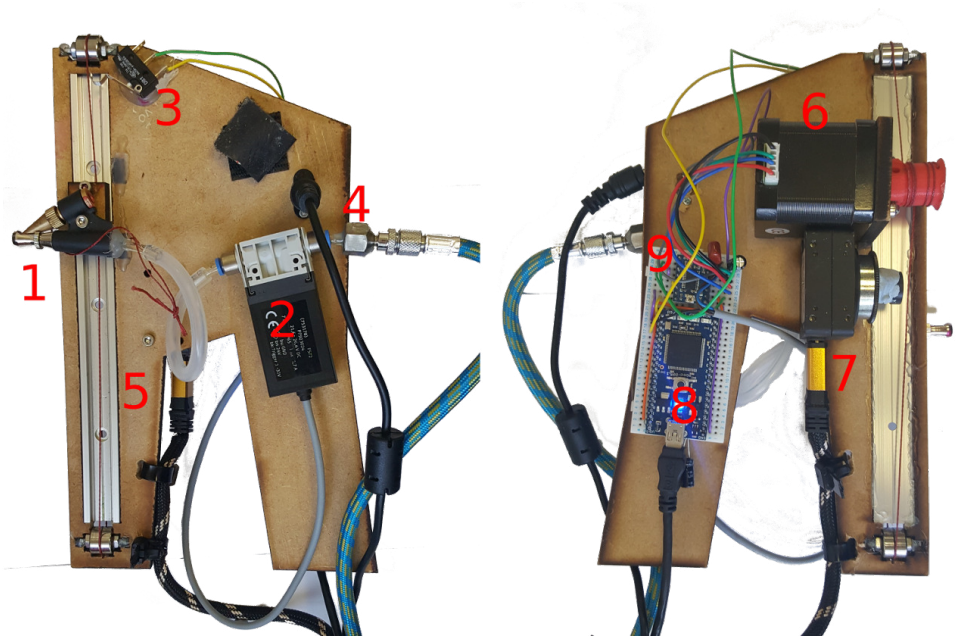


Figure 6.6: The robot used for the hardware experiment. 1: Airbrush nozzle. 2: Festo 1ms air valve. 3: Limit switch. 4: 15PSI air input hose. 5: Igus DryLin low profile slide rail. 6: Nema 17 stepper motor. 7: Point Grey Chameleon 3 USB 3 camera. 8: MBED LPC1768. 9: Pololu stepper motor driver.

are where paint is placed. Ideally the real paining should match that of the simulation.

It can be see from these images that the paint was placed predominantly in the correct location, though there are some anomalies that should be explained. The small amount of misalignment between the target locations and where the paint was deposited is due to a combination of an offset in the optical positioning and perhaps a lack of precision in the registration of the object to the markers on the base. The offset is around 6mm in the worst case. Additionally to the offset, there is some spurious paint placement, this is due to the loss of vision of the markers as the camera becomes very close to the object. To solve this issue the camera should be placed such that it does not lose vision on the markers when close to the target object. Experiments presented later in this thesis use a retro-reflective marker based motion capture system to solve this issue.

6.4 CONCLUSION

In this chapter a method for generating a trajectory for an airbrush nozzle on a *reduced degree of freedom* hand-held robot was outlined. This method had four stages: sampling the swept space for required paint density; building

a directed graph in this space, sampling from the required paint density to generate the edge weights; solving for the path that can maximise the quantity of paint delivered using the Dijkstra method. Finally only a small part of the plan is simulated in a dense manner to update the state of the task, and is sent to the robot for implementation. This allows the robot to plan paths that are likely to be close to optimal, but still be reactive to change in user behaviour on which the plan is based.

The proposed method was shown to have sufficiently low run time to run in a real time system. This was demonstrated by trialling the system on hardware as well as measuring the run time directly. The run time was significantly shorter than a simple greedy algorithm. For a typical scenario the proposed method took 32.6ms to produce a decision for the next time step, whilst considering a time horizon of 700ms.

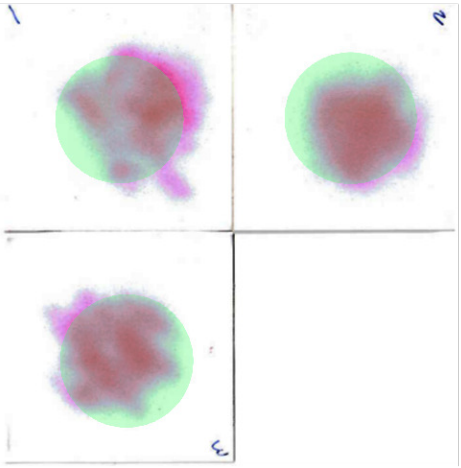
The path selection quality was compared against an exhaustive evaluation of all possible paths, the average path returned was in the top 15% of all paths, though the majority were in the top 5%.

Future work on practical implementations of such a spraying robot should verify the final distribution of the paint on the model in a quantitative manner. This could perhaps be achieved with an analysis of the colour saturation of the paint, or by using an additive to the liquid that can be detected post experiment, such as fluorescent dyes.

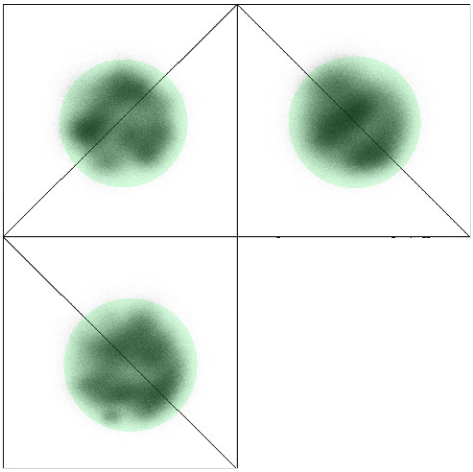
The algorithm presented in this Chapter will be used in Chapter 7 and be compared to a manual operation of the robot, and trigger only assistance.

(a) The object to be painted.

(b) The Object after painting.



(c) A scan of the sides of the model that were painted



(d) The equivalent view from Figure 6.7c taken from the simulation.

Figure 6.7: The object is a 60mm wide cube at an angle. The target is three 34mm circles on each of the top faces, as shown in Figure 6.7c, the green overlay indicates the target areas. Qualitatively it can be seen that the paint is broadly in the correct place, the displacement of each circle is within 6mm of the correct location, this drift is most likely due to registration of the 3D Object and positioning accuracy, anomalies far outside the circles seems to be temporary glitches in the position tracking.

7

TESTING ASSISTANCE IN REDUCED DEGREE OF FREEDOM ROBOTS

This chapter has been produced from the original work published in the following locations:

Elsdon, J., and Demiris, Y. (2018), Augmented Reality for Feedback in a Shared Control Spraying Task, *in* 'IEEE International Conference on Robotics and Automation', pp. 1939–1946.

7.1 INTRODUCTION

This chapter attempts to answer the question: Are *reduced degree of freedom* robots, when paired with suitable algorithms, an effective means of aiding a user in a spraying task?

The primary contribution of this chapter is a user study that aims to evaluate the path planning algorithm developed in Chapter 6. This study will be using version three of the single axis hand-held robot developed in Section 3.1c. This is a *reduced degree of freedom* robot when applied to the task described in this Chapter.

Three assistance modes will be compared: *manual*, *semi-auto* and *auto* modes. These are analogous to the modes that were presented by Gregg-Smith and Mayol-Cuevas (2015). *Manual* mode will leave movement of the end effector and the triggering of the spraying to the user exclusively. *Semi-auto* automates the triggering of the spraying, and *full auto* assists via the algorithm described in Chapter 6.

For a user to use this system it is important that they have access to the current state of the task, and ideally they would have access to some additional visual aids to help them complete the task. For this purpose an AR headset will provide the user with information regarding current paint density, a targeting aid to help them aim, and a visualisation of where paint is still required. This will be introduced in Section 7.2.

It was found that the more sophisticated assistance modes increased the performance, however the majority of the benefits could be had with the simpler trigger assisted semi-auto mode. The participants became split on their

opinion of the full automatic mode, with a sub-set finding it more taxing to use than the alternatives; suggesting that there is some conflict between the users' preferences and the robot's algorithm. It is hypothesised that this conflict is at the *tactical* planning level. This conflict motivated experimentation with explicit allocation of the *tactical* plan, found in Chapter 8.

7.2 USING AUGMENTED REALITY TO CLOSE ACTION PERCEPTION LOOP

The preliminary hardware trial presented in Section 6.3.2 required the user to know in advance what the spraying distribution should be. To make a more realistic system, the user should be able to be informed of the task, and the progress of the task by the robotic system.

Due to the required shared understanding of the task, the robotic system must be able to indicate the current status of the task such that the user can collaborate effectively. This is achieved by extending the robotic system using an augmented reality (AR) headset, specifically the HoloLens by Microsoft. This is a binocular AR headset, and therefore can indicate 3D information to the user. In order to present holograms in the users vision, the headset tracks the users position using a visual odometry system. Whilst this works well for visualisations where positioning is not critical, it does suffer from drift as the Hololens adjusts its internal map or loses trackable features in the scene. The solution to this problem was found using two different methods in Section 4.3, though for experimentation presented in the following section only the *time series method* was used. The hardware setup that is discussed in this chapter is shown in Figure 7.1.

Finding a way to logically provide task specific information to the user of a shared control system is critical as the task cannot be completed without the cooperation of the human user. An outline of a simple colour scheme that represents target areas, completed areas, and overdosed areas in a spraying task in a logical way is detailed in Section 7.2.1.

The benefit of a shared control system is that the parts of the task that require access to rich information or precise timing can be offloaded to the robotic part of the system. Providing some assistance is likely to reduce task load on the user, though knowing exactly how much extra help still adds utility to the system can allow designers to optimise other aspects of the system, such as size or cost. In Section 7.3 three levels of assistance are evaluated for a spraying task, considering the users performance at applying liquid, the speed in which they do the task and their subjective task loading.

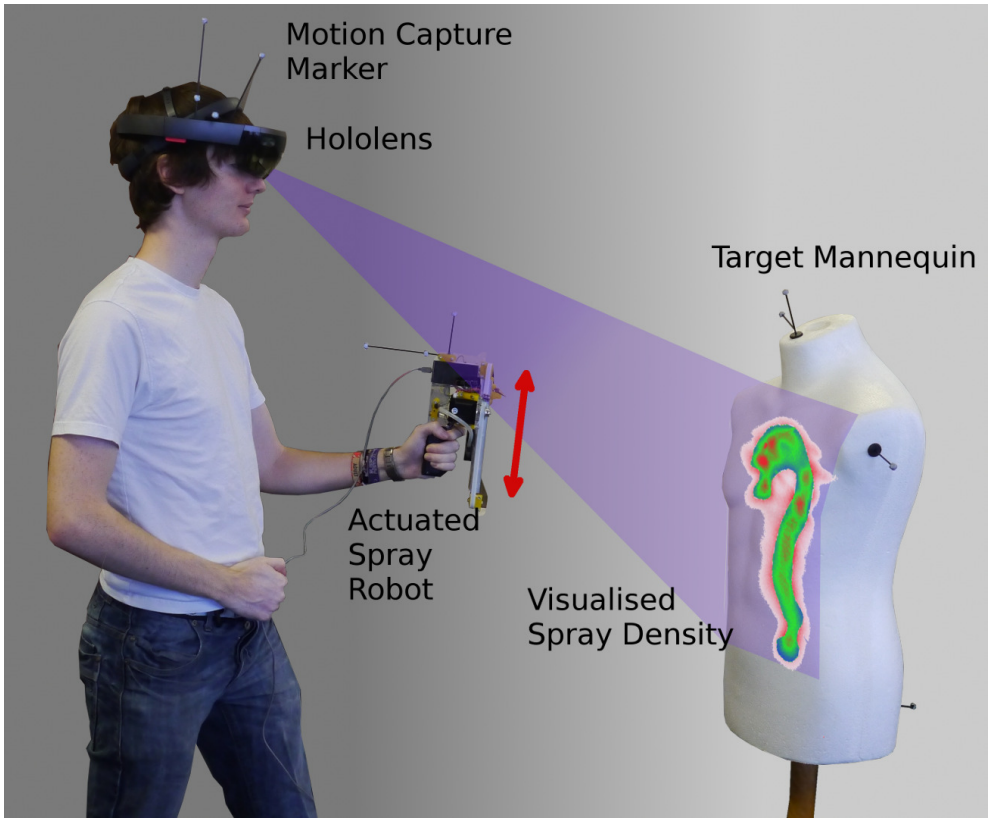


Figure 7.1: The experimental setup used in this work. The Hololens, spraying robot and mannequin all are tracked with retro-reflective markers. The visualisation of the spray density is only visible to the user of the system as it is displayed to them through the Hololens. The spraying robot's head can be actuated up and down to assist with the spraying task.

This is a similar paradigm as presented by Gregg-Smith and Mayol-Cuevas (2015), summarised in Section 2.3.1, though the task detailed here is more realistic. Further the robot that is used in this task is a *reduced degree of freedom* robot rather than a *locally capable* robot as presented Gregg-Smith and Mayol-Cuevas (2015).

7.2.1 *Proposed Visualisation*

Drawing on the work of Kim et al. Kim et al. (2007) it was decided that a 2D representation mapped directly to the surface of the 3D object using augmented reality was the most intuitive method of providing feedback in a spraying task. This task has an additional complication as compared to that work: there is no implication that the paint should be evenly applied everywhere, there will instead be target regions. This means that feedback should be given to the user regarding areas to spray, the progress towards the target dose in those areas, as well as highlighting any overdosed areas. This visualisation should also be natural to understand.

The proposed visualisation is as follows: regions that require spraying, target regions, are blue. As the area becomes filled this will transition to green. If the area becomes overfilled then the area will transition through a gradient to red. This is shown, along with the indication of reward in Figure 7.2. Traditionally green is seen as "good" and red as "bad", therefore the user will see that their job is to make all blue areas green, whilst making as few areas red as possible. Areas that were never intended to receive spray start white, due to this being the colour of the real world model they are spraying. The white will shift through a gradient to red as the area receives more spray. Full red in this case would be reached when the overdosing is in proportion to the typical correct dose within the target regions.

7.3 CONTRASTING DIFFERENT LEVELS OF ROBOTIC ASSISTANCE IN SPRAYING TASK

This section will detail a user trail to investigate the extent to which robotic assistance could help in a 3D spraying task, specifically with a simple and light weight *reduced degree of freedom* hand-held robot. Specifically the perceived task load and painting performance metrics will be investigated, such as completion time and accuracy.

The visualisation that is presented in Section 7.2.1 will be used to ensure there is a shared understanding of the state of the task. This knowledge

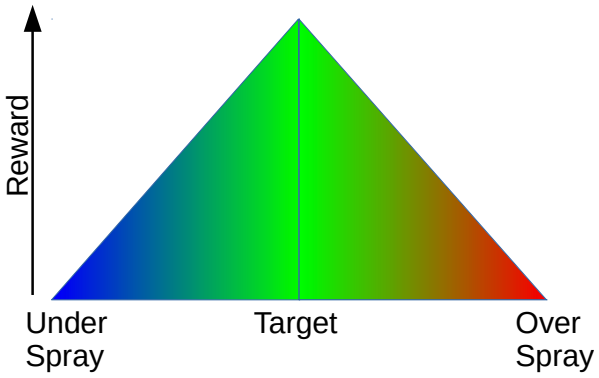


Figure 7.2: The colour scheme for a region that requires spraying. Blue would indicate to the user that they should spray the area more, green that the area is correctly sprayed and red that it has become over sprayed. The vertical height in the image refers to the reward that is applied to the whole spraying job at this location, when the score becomes negative the colour will stay red.

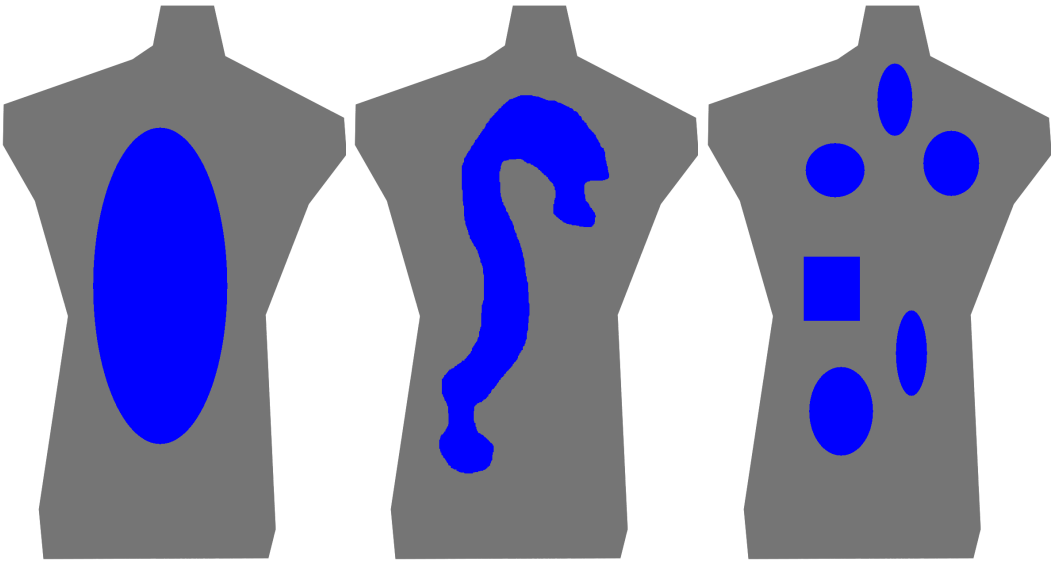
makes it possible to form a shared control behaviour with the robot. In this case the human will be directing the *tactical* level of planning and then sharing the *operational* level of the plan with the robot by providing the locomotion of the robot as well as inherently sharing control of the position of the spray nozzle.

7.3.1 *Task Outline*

The task that the user will be attempting to complete will be to spray virtual liquid onto a mannequin on the zones indicated to them on the augmented reality headset. Three modes will be tested. *"Manual"* gives the user full control of the trigger, which releases the virtual paint from the nozzle, and nozzle remains stationary on the robot. *"Semi automatic"* mode activates the trigger on the users behalf when the simulation, described in Section 6.2.6, finds that spraying will increase the task score, the selected path in this instance will always be the stationary option. This acts similarly to the algorithm presented by Prévost et al. (2016). Finally, *"automatic mode"* will also activate the trigger, and additionally have the capability to slide the nozzle up and down a short gantry mounted on the handheld robot. The movement of the nozzle is calculated in a receding horizon manner using the method presented in Chapter 6.

The user will partake in three painting experiments per assistance level, which are shown in Figure 7.3. After each assistance level they complete the NASA TLX survey Hart and Staveland (1988) using the official NASA

IPad application. They will be wearing the Microsoft HoloLens which will overlay on the mannequin the target areas and their current paint status as described in Section 7.2.1, the setup can be seen in Figure 7.1. The user will communicate to the assistant when they feel that they can no longer make reasonable progress on the task. By the end of the experiment they will have filled the three different target patterns with each of the three assistance levels. The users are allocated one of the six orders that the experiments can be in at the beginning of the experiment. Before each new mode the users were given the opportunity to test the mode and to ask any clarifying questions that they had. Before the experiment began, the participants were invited to read details about how the data would be used and stored, they then confirmed their consent to those terms.



(a) Target 0 (b) Target 1 (c) Target 2

Figure 7.3: The three patterns the users were asked to spray in each mode. a is designed to test area fill, b for detailed strips, and c for small isolated areas. Further the boundary to area ratio increases through the levels

7.3.2 *Results*

There are three key areas of measurement that will be presented here, NASA TLX Hart and Staveland (1988) that indicate task load, task completion time, and accuracy of paint placement. In total there were 18 participants, 4 female and 14 male, aged between 20 and 34.

7.3.2.1 *NASA TLX*

As can be seen from Figure 7.4, there is a marked decrease in the median task load as the level of automation increases from 63.5 in manual to 47.835 in automatic mode. Though the median of the task load decreased as automation increased, the automated modes were not favoured by all, in fact both the highest and the lowest task load score was registered for the automatic mode. Comparing the automatic modes to the manual mode using an independent samples t-test, when considering the hypothesis *"Semi/Fully automatic mode decrease the task loading compared to manual mode"*, one can conclude that there is not enough evidence here to meet a 5% confidence threshold (Fully automatic mode $t(18) = 1.66, p = 0.106$). The factors involved in task load are shown in Figure 7.5; overall there is no significant change in the relative importance of each factor across the experiments.

7.3.2.2 *Completion Time*

Across nearly all of the trials manual was the quickest mode, with little difference between the automatic modes. Though this does require some careful qualification. Due to the nature of an automatic mode not allowing the participant to make a mistake, users often become quite perfectionist, hunting for the smallest area of improvement. In the manual mode however they get nervous that they will ruin what they have achieved and may stop early to avoid incurring negative scores due to overdosed regions. This effect could likely be removed easily by instructing the user the level of coverage the task requires, such that they do not waste their time, and further practice to remove their nerves in manual mode. To help remove this effect in this study, all times referred to are the times taken to get from 10% to 90% of final coverage for that attempt, to remove the thinking time at the beginning and the hunting behaviour observed in the automatic modes at the end. With this correction applied, Figure 7.6 shows that the time taken in the different modes did not vary significantly. Only in round 2 was there any appreciable difference, though this is explainable by the fact the perfectionist attitude of the users is applied per sub-patch, which is not removed by the filtering described above. The sub-patches in question can be seen in Figure 7.3.

7.3.2.3 *Accuracy*

To effectively compare performance across trials the mean squared error (MSE) per pixel is a good measure as it is agnostic to the dose level (which happens to be constant) and the total area required to be covered. As can be

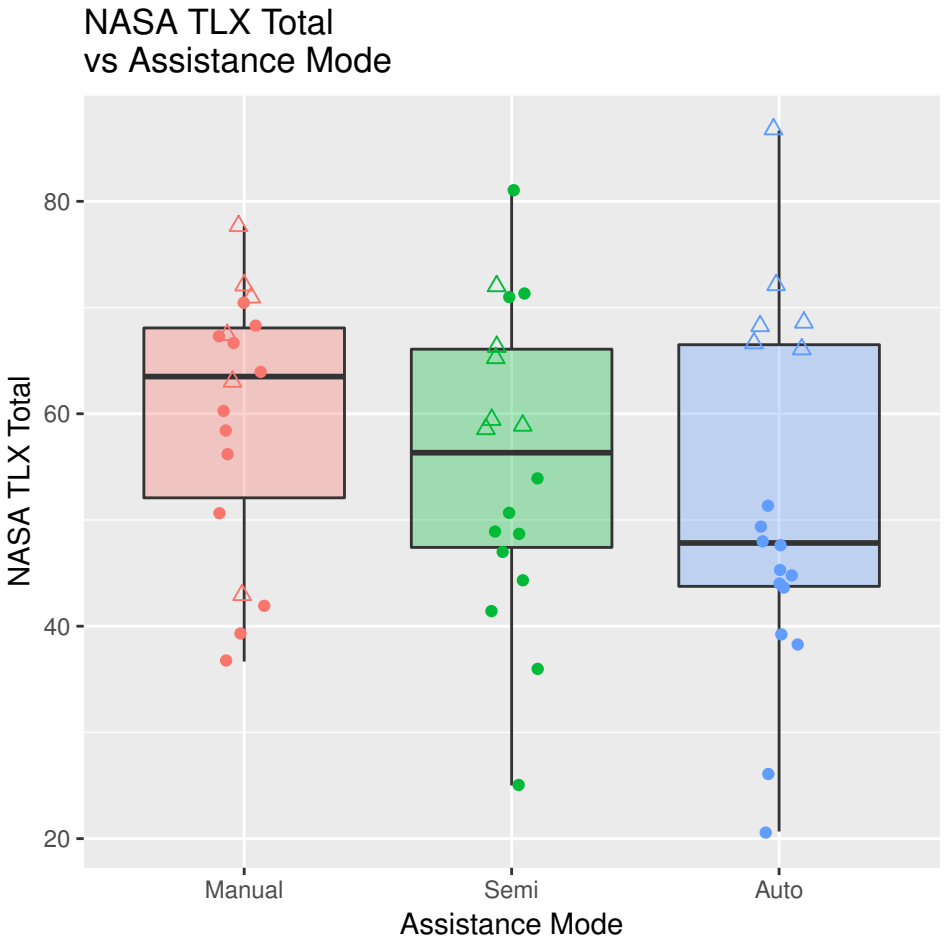


Figure 7.4: The combined TLX scores by mode. It can be seen that increasing the assistance level reduces task load, though there is a larger range of subjective task load at the highest assistance mode. The group that rates *auto* mode as having high task load have been highlighted in the other modes using the triangle point markers. It can be seen that this group becomes more separated as assistance level increases.

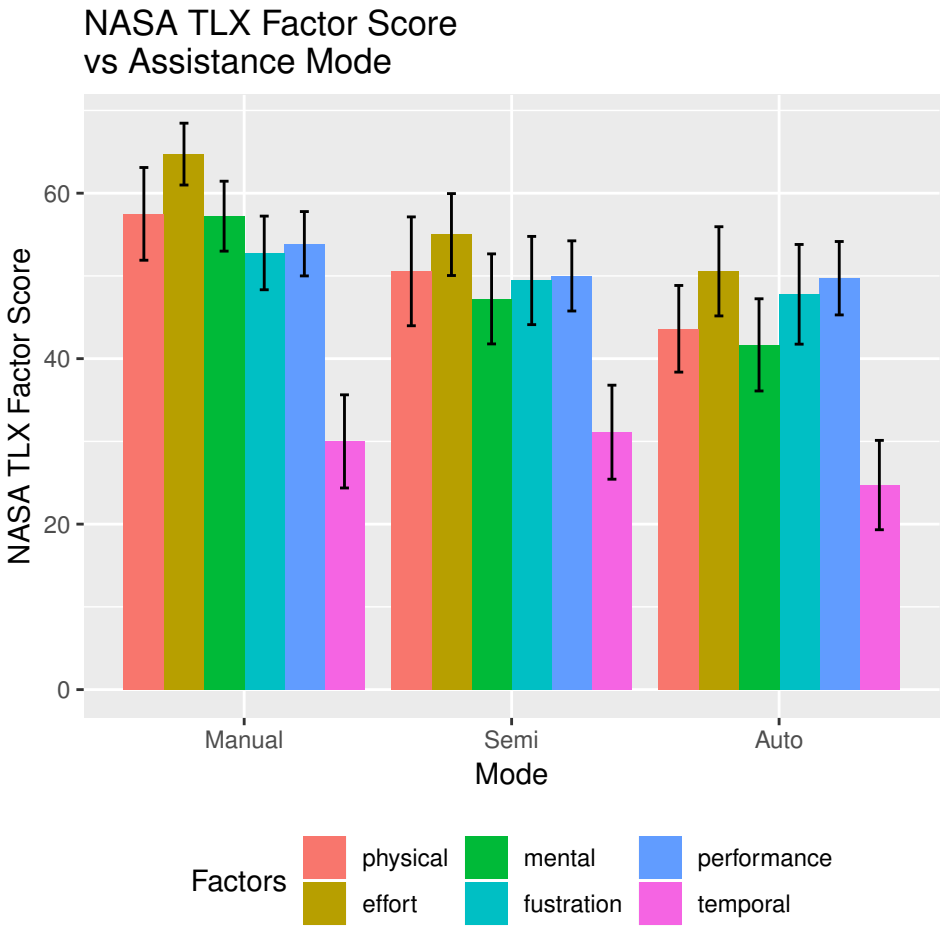


Figure 7.5: The factor weighting by mode. It can be seen that there is little change in the relative importance of each of the factors when changing the assistance mode.

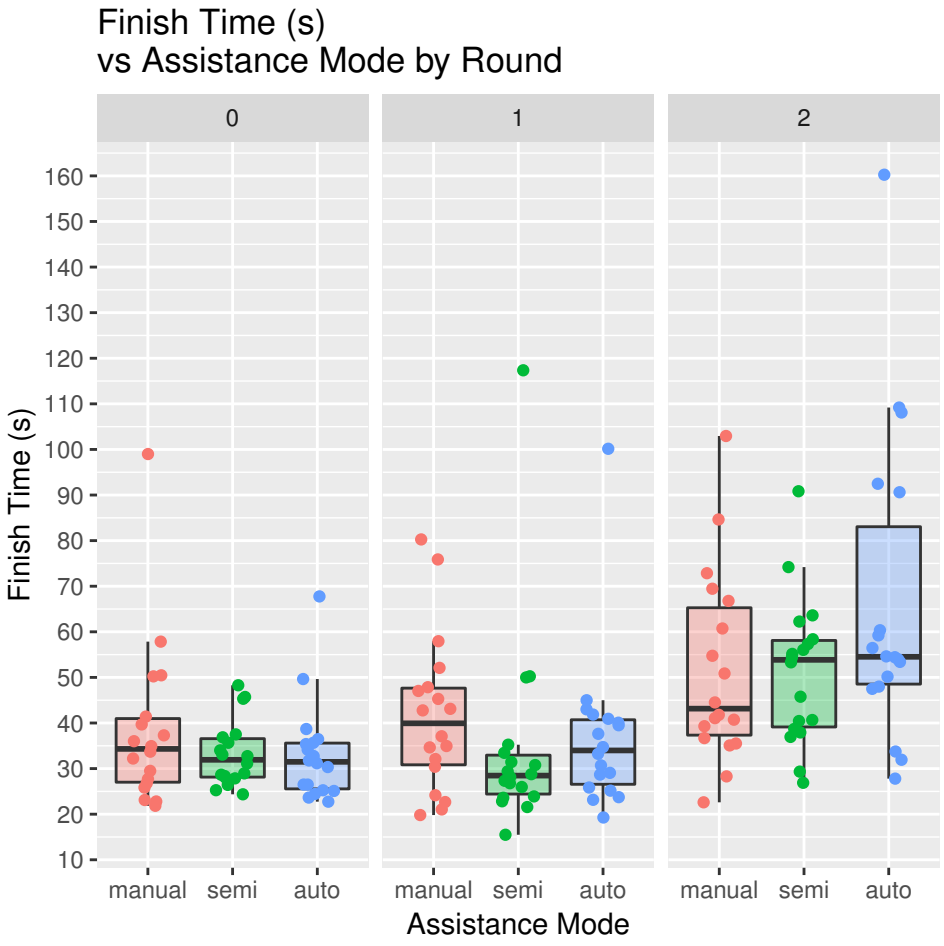


Figure 7.6: The time to complete the spraying task. Time is taken from the interval between 10% and 90% of the maximum score for that run to remove unproductive time at the beginning and end of each run. It can be seen that assistance mode has only a small effect on finish time. Round 2 consists of many small patches, and as such some participants became perfectionist on a per patch basis.

seen from Figure 7.7 the MSE is best in the automatic modes and worst in manual mode. There is between a 33% and 45% reduction in MSE between the automatic modes and manual mode. There is no significant difference between the automatic modes. Figure 7.8 shows a typical paint distribution. It should be noticeable that in manual mode there is significant amounts of paint outside of the intended boundary, whereas the automatic modes have a similar kind of paint distribution. When considering the statement *"Fully/Semi Automatic modes lower the MSE, compared to manual mode"*, the independent samples t-test suggests for both fully and semi automatic that the statement is correct. The results are summarised in Table 7.1, the three values for the t-tests being for each spray round, as shown in Figure 7.3.

Table 7.1: A summary of the independent sample t-test results comparing both Fully and Semi Automatic modes to the Manual Mode. This shows that the increase in accuracy shown in Figure 7.7 are clearly statistically significant.

Round	**mode**	**t(18)**	**p-value**
0	Semi	2.78	0.0086
1	Semi	2.69	0.0110
2	Semi	4.75	$3.62e^{-5}$
0	Auto	2.51	0.016
1	Auto	2.61	0.013
2	Auto	5.08	$1.34e^{-5}$

7.4 CONCLUSION

In this chapter the algorithm detailed in Chapter 6 was used in a user trial to investigate whether the *reduced degrees of freedom* paradigm is acceptable to users, as well as investigating the nature of the *tactical* and *operational* plans. It was found that there is a large advantage in moving the *operational* plan to be controlled by the robot, which has fast response times and analytical access to the state of the task. However it was also found that there is a mix of responses to the assistance that helps move the spray nozzle from the participants. A sub-set of participants reported conflict with the system and rated the system poorly for task load. This seems to be due to the fact that the *tactical* plan is split between the user and the robot in this case. Users reported that "the robot ruined my plan for the task" (paraphrase), this is very similar to what was reported by Gregg-Smith and Mayol-Cuevas (2015). This conflict seems to be due to users not having a firm grasp of how the assistance algorithm is attempting to help them. This idea is called a

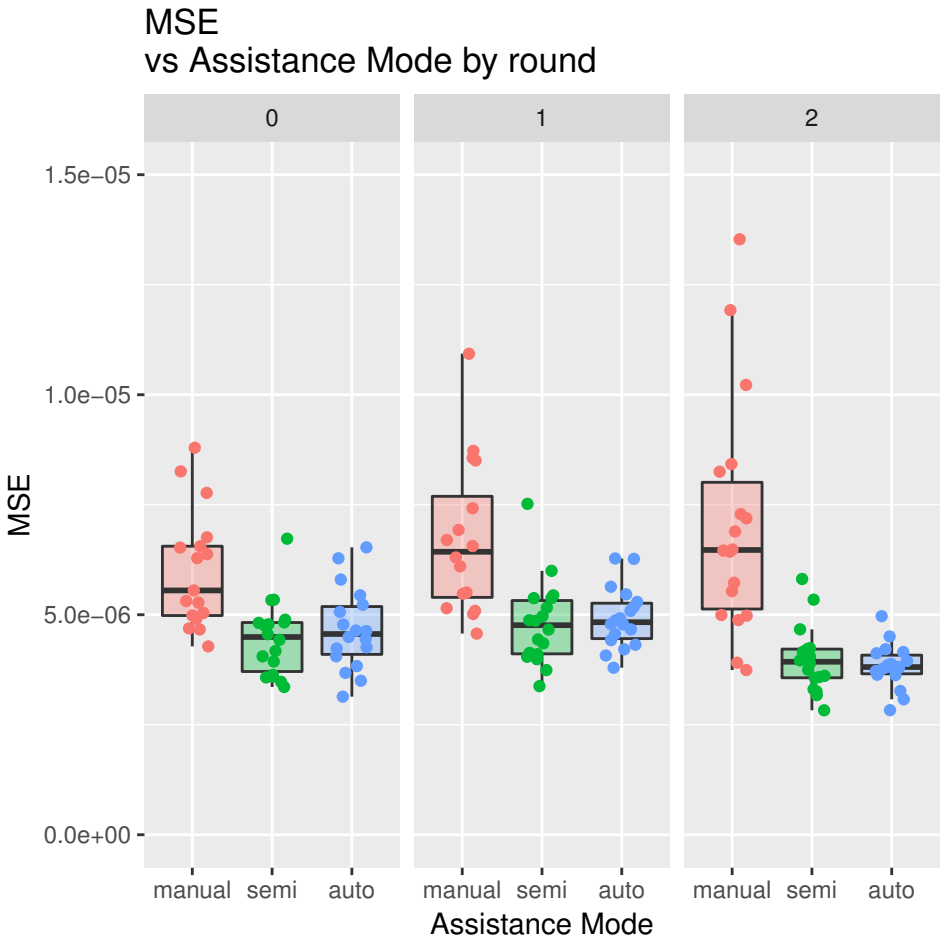


Figure 7.7: The Mean Squared Error (MSE) per pixel in the bounding box surrounding the target patch. It can be seen that the manual attempts are much more error prone, also the user variation is large with the manual mode, and small with the automatic modes. Variation between rounds is also small in the automatic modes. Some outliers in the manual categories are omitted, at around 2e-5ml/pixel.

(a) Typical result of manual spraying (b) Typical result of semi auto spraying (c) Typical result of auto spraying

Figure 7.8: A typical example of the 3 modes used on a given target. Green sections are well painted, blue is in need of paint, and red areas are over painted. It can be seen that in manual mode there is significant over spraying.

mental model of the shared control. This conflict partially motivated the work presented in Zolotas, Elsdon and Demiris (2018), where a user study was constructed to measure if AR visualisations could aid in the forming of a shared model.

Within the context of this thesis, this result also motivates Chapter 8, where a *locally capable* hand held robot is used to more decisively allocate the *tactical* plan to either the user or the robot. Due to the fact that user's and the robot's control of the nozzle in a *reduced degree of freedom* are entangled and cannot be fully separated, such a decisive allocation of the *tactical* plan is not possible without disabling the user's or the robot's control of the position of the spray nozzle entirely (as was the case with the semi-auto mode in this chapter).

8

LOCALLY CAPABLE HAND-HELD ROBOTS

8.1 INTRODUCTION

This Chapter will concern the concept of *locally capable* hand-held robots, these are hand-held robots that can complete the task independently as long as they are in the vicinity of the task. They can decouple user motion of the handle of the robot from the end effector of the robot in all the degrees of freedom that are relevant to the task.

In Chapters 6 and 7 the complementary category, *reduced degree of freedom* hand held robots, were investigated. One of the key lessons learned from that series of work is that there is a sub-set of people that are likely to conflict with the robot's assistance. This conflict is hypothesised to be at the *tactical* layer of planning. Unfortunately a *reduced degree of freedom* hand-held robot cannot take full control of the *tactical* layer of planning. To do so would require a robot specified path to be implemented, and the user would have to comply adequately to allow the end effector to track the prescribed trajectory. As was discovered in Chapter 5, the user's ability to track a specified trajectory is limited.

A *locally capable* hand-held robot allows the user's motion to be decoupled from end effector motion, allowing a trajectory to be tracked that was specified by the robot. The user is only required to comply by moving the robot in such a way that the robot can continue to reach the current planned trajectory.

Thus with a *locally capable* robot the *tactical* plan can be allocated fully to the human or fully to the robot. Further the *operational* level of the plan can also be almost entirely allocated to the robot, as motion of the end effector can be decoupled from the user, and the trigger for the end effector can also be robot controlled. Thus the skill requirement in finely controlled motion and timing that is required for the *operational* level of control can be removed.

The fidelity of visualisations shown to the user will also be investigated. Both high fidelity and lower fidelity visualisations will be used in the user study presented in Section 8.4.

More information to the user could seem like it is a universal good, though overwhelming the user with information is certainly possible. It was shown

that having visualisations present reduced performance in a shared control wheelchair study, Zolotas, Elsdon and Demiris (2018). Therefore it is possible that a system that provides the minimum necessary information could perform better than one with more information.

This Chapter demonstrates that a shared task with a *locally capable* robot is possible with lower fidelity visualisations, though the participants did perform better when the full state of the task is shown to them. Importantly, when the full state is available to the user, the agent that constructs the *tactical* plan is not important. Though in the case where detail of the task is restricted, there are significant performance gains to allowing the robot to automate the *tactical* plan.

8.2 STRATEGY AND TACTICAL PLANNING AND WHO CONTROLS IT

As discussed in reference to the work presented by Michon (1985) in Section 2.2, there are multiple levels at which an agent plans a task. Their taxonomy highlighted three modes of planning: *strategic*, *tactical* and *operational*. In the context of our drawing task, *strategic* planning is the planning of the set of lines to be drawn. This strategy could be the well ordered zigzags typical in automotive painting robots, or a pre-designed piece of vector art, as it will be in this work.

The *tactical* planning in the drawing task is the selection of which stroke from the *strategic* plan to execute next. Finally the *operational* plan is the low level tracking of the selected path and initiating the drawing output at the appropriate time.

All of these modes of planning and action could be primarily solved by either a robot or a user in a shared control system. Choosing which element of planning and action should by handled primarily by the robot or the user is not always straightforward. For this work the *strategic* plan was made manually, and is provided as an input as an SVG vector file. The *operational* plan will be made by the robot, as its end effector will track the currently targeted line. The ownership of the *tactical* plan is a factor that will be tested in this work; both the user and the robot have the opportunity to make the *tactical* plan during the trial presented in Section 8.4.

8.2.1 *Path Selection Framework*

There will be two path selection algorithms, one that prioritises the users input (Section 8.2.3), allowing the user to form the *tactical* plan, and one where

the priority is to follow the robot's *tactical* plan (Section 8.2.4). Both of these algorithms use the same underlying framework, which will be presented in this section.

The input to this module is the strategic plan, which in this case is the set of all vector lines that are to be drawn, as well as the current location of the end effector, and depending on which agent has control of the tactical plan, various bias locations are also provided.

The set of vector lines are loaded into an octree Meagher (1980), which is a data structure that is constructed to represent a three dimensional space. It is formed as a tree structure, where each node represents an axis aligned bounding cube. Each node can be one of two types: an octant node or a leaf node. An octant node is an internal node of the tree which points to 8 children nodes. These represent the sub-division of the bounding cube into 8 smaller cubes that have an edge length half that of their parent. A leaf node is one that has no children, and can contain the data that is to be stored. In the case of this work, the leaf nodes store the end point of a vector line and a pointer to further information about the vector line. This architecture is shown in the diagram in Figure 8.1. The benefit of an octree is that it makes the retrieval of a particular point very efficient, in the order of $O(\log(n))$.

This approach only stores the end points of the vector lines, thus the method presented can only allow the robot to start a line at either one of the ends, and not in the middle of the vector line. When the system selects a line to follow it is removed from the octree. If the line needs to be terminated in an incomplete state, for example if the user moves the robot out of reach of the current line, the end points are recalculated and then are re-entered into the octree. This is detailed in Algorithm 6.

In order to minimise the memory footprint of the octree structure it is constructed in a dynamic manner, when nodes are added or removed the tree structure is either expanded or culled to ensure that there are no empty leaf nodes, or underutilised octant nodes.

The octree structure also makes it efficient to find the N nearest neighbours to a given target point. This makes it a natural answer to the issue of finding the nearest vector from the current end effector position. This approach forms the basis of how the tactical planning is changed from the user to the robot. Therefore a brief summary of the generic solution will be shown here, such that the augmentations to this technique made in later sections are clear.

Algorithm 6: An algorithm showing how the process of selection lines progresses. If a selected line is completed or falls out of the reachable area a new one is selected. Lines that are left incomplete are added back to the octree to be selected later. If a newly selected line is too far away from the current end effector location, a new trajectory is generated that travels to the selected line's location.

```
while !ocrtree.empty() do
    if !line.complete() AND line.reachable() then
        line.stepAlong();
    else
        if !line.reachable() then
            octree.store(line);
        else
            if octree.getClosest().distance > ϵ then
                line = travelLine(octree.getClosest());
            else
                line = octree.getClosest();
return;
```

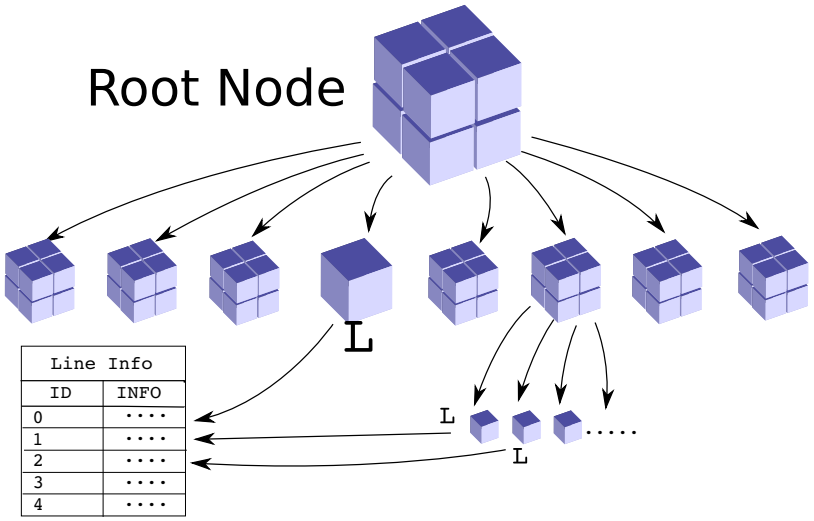


Figure 8.1: A diagram of the octree structure used for storing and looking up line segments. The end points of all of the vector lines are stored in an octree structure along with a pointer to a list of line segments, that contains the full information about the line, such as its completion status the line type (line, spline, circle etc).

8.2.2 *Octree Nearest Neighbour Search*

The method is described by Samet (2008) in comparison to other methods. However though this implementation is considered one of the classical methods of solving the K-nearest neightbors problem within an octree. Samet refers to it as the *best first* method.

If the root node is a leaf node then the point contained within is returned and the method can exit. If it is a octant node it is entered into a priority queue, this serves as an initialisation of the iterative process to follow. A priority queue is a data structure that will allow access to the element with the lowest key value, which in this case is distance to the target point, in order $O(1)$.

Then the front node from the priority queue is recovered. If the node is a leaf node the data contained within is added to the results list. If it is an octant node, then each of its children is added back into the priority queue. For each of the children, if it is a root node the distance key is simply the distance between the target and the contained point, if it is an octant node then the closest possible point is taken. The closest possible point would be the target point itself if the target is contained within the bounding box of the node in question, else the closest possible point will lay on either the faces, edges, or corners of the bounding box.

The process of recovering nodes from the list, adding data points to the results list, and adding children back into the priority queue continues till the return list is of length N. This method is summarised in Algorithm 7.

8.2.3 *A Method Where the User Generates the Tactical Plan*

In this case the *tactical* plan is left mostly to the user, they are deciding which part of the task to tackle next. The line that is close to the centre of the reachable area and close to the current end effector is selected, whilst prioritising lines that are inside the reachable area.

In order to pick the next vector line, the algorithm described in Section 8.2.2 is employed using the current end effector location as the target, which retrieves an ordered list of the N closest vector start/end points. However two modifications are made. Firstly, rather than a simple euclidean distance between the target location and the node (as defined by the getDistance function in Algorithm 7), a weighted sum of euclidean distances is used instead. These two distances are: the distance between the current end effector loca-

Algorithm 7: The priority queue stores a node object, retrievable in order of the smallest associated distance. distance(point1, point2) returns the euclidean distance between points. minDist(point,bounds) returns the smallest euclidean distance between the point and any location in the bounding box.

```
Data: Octree filled with data points, target point
Result: An ordered list of N-nearest neighbours
if root node is leaf then
    returnList.push(node.data);
    return returnList;
else
    priorityQueue.queue(closestDist(target,node),node);
while !priorityQueue.empty() && returnList.length < N do
    node = priorityQueue.get();
    if node.isLeaf then
        returnList.push(node.data);
    else
        for child = node.getChild do
            dist = getDistance(target,child);
            priorityQueue.queue(dist,child);
return returnList;
Function getDistance(target, node):
    if node.isLeaf then
        d = distance(target,node.data)
    else
        d = minDist(target,node.bounds)
    return d
End Function
.
```

tion ($\mathbf{E}$) and the node; and the distance between the centre of the reachable area ($\mathbf{C}$) and the node.

$$\begin{aligned} \text{dist} = (1 - f) \times \text{getDistance}(\mathbf{E}, node) \\ + f \times \text{getDistance}(\mathbf{C}, node) \end{aligned} \tag{8.1}$$

This weighted sum has the effect of changing the behaviour from just tackling the nearest vector next, to biasing the selection towards a vector line that is closer to the centre of the accessible area. This helps the user keep the end effector centred in the reachable area, as well as giving the user more influence over which line will be tackled next. Changing the weighting factor (f) changes how strictly the robot attempts to stay near the middle of the reachable area versus picking the vector line that is the nearest to where the end effector is. This allows a trade off between the user's ability to be selective of the next line and routing efficiency.

Next, a new list is formed by selecting all the vector line end points that fall within the reachable area of the robot. The lists are then compared, and the closest line that is also reachable is selected. If there are no lines to complete within the reachable area the closest point from the first list is selected.

The filtering based on accessible area causes the robot to attend to all the lines that fall within the reachable area before selecting a line that falls outside of the reachable area. This helps the user to have control over with areas are finished first, even if lines outside the reachable area are in fact closer by the above previously defined distance metric.

8.2.4 *A Method Where the Robot Generates the Tactical Plan*

Due to the nature of a handheld robot, some actuation must always be completed by the user. However the user's role can be limited by having the robot calculate the *tactical* plan. The *operational* plan would still need to be a collaboration in this case due to the limited range of the robots end effector.

This *tactical* planning method is similar to the algorithm in Section 8.2.3. The primary difference is that a set of points fixed in the work space are used as biasing points rather than the centre of the reachable area. In this case a set of w points ($\mathbf{W}_i$) are located in a line at the right hand extreme of the work space. Further, the filtering based on accessible area is removed, such that the robot considers the whole task and is not limited to completing reachable areas first.

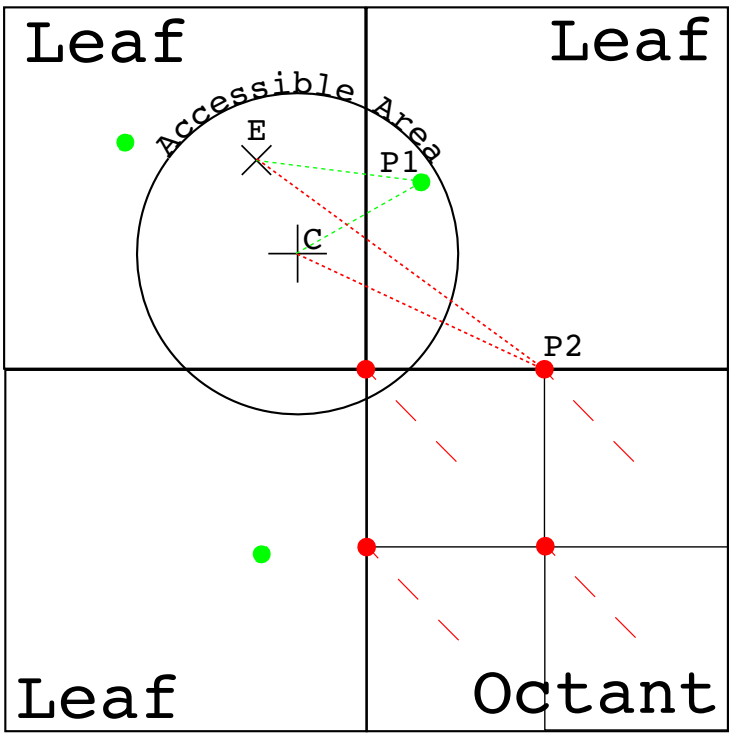


Figure 8.2: This diagram shows how the distance metric is calculated for the user position based behaviour. **E** indicates the current location of the end effector, **C** indicates the centre of the accessible area. The green circles represent leaf node data points, distance to these are calculated directly. The red circles indicate the closest possible point when considering the distance to an octant node. This method uses the weighted sum of the distance between a prospective start/end point, and the centre of the accessible area and the current end effector location. The fine dashed red and green lines here represent two independent calculations for P2 and P1 respectively. The sparse dashed red lines indicate to which octant the red points belong. In this example the point P1 would likely be chosen.

These changes have the effect of making the robot have a tactical plan that consists of starting on the right hand side and completing all the vector lines while moving leftward. Again, the relative weighting can be changed to bias the robot to augment its behaviour. Having a large weight on the fixed points on the right hand side will case the robot to be very strict about moving from right to left, perhaps making many wasteful movements to comply. The opposite weighting will make the robot less strict about moving from right to left, and instead complete local vector lines to a greater extent, even if they are completed somewhat out of order, thus being less wasteful in traversing between lines.

This right to left behaviour is one example of a simple robot driven behaviour. Using the method presented here a number of other behaviours could be designed. For example, if a ring of bias points are set around a design in a circle, the robot will have a *tactical* plan of starting from the outer parts of the design and working inwards.

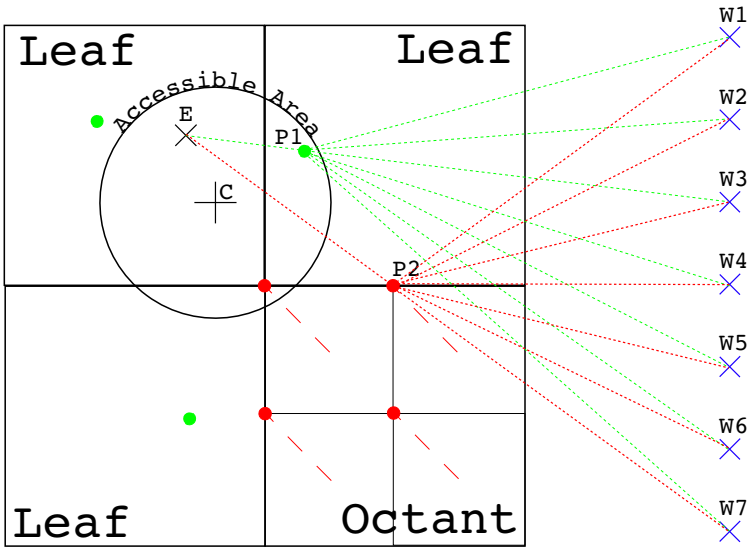


Figure 8.3: This diagram is similar to Figure 8.2, though is demonstrating how the robot driven right-to-left behaviour is generated. The points W_1 to W_7 indicate points that are fixed in the work space. You should note that this calculation does not use the centre point of the work space, C, and therefore has a behaviour that primarily approaches the selection in a global manner. The exception being that the calculation does include the distance to the current end effector location (E), this is such that if lines are very close the algorithm will select them. The user will have very little control over which line will be selected next in this case. In this example point P2 is likely to be chosen, though it depends on the relative weights in the weighted sum.

$$\begin{aligned} \text{dist} = (1-f) \times \text{getDistance}(E, node) \\ + \frac{f}{w} \times \sum_{i=0}^{w} \text{getDistance}(W_i, node) \end{aligned} \tag{8.2}$$

8.3 VISUALISING THE TASK USING AUGMENTED REALITY

A hand-held robotic system is inherently a shared control system, therefore there should to be some exchange of information between the user and the robot. Information from the user to the robot is achieved via measurement of where they are moving the base of the robot, and is implicit in the *tactical* plan algorithms seen in Section 8.2.3 and Section 8.2.4. Information flow from the robot to the user consists of a mix of robotic gesturing and augmented reality feedback. The gesturing happens implicitly due to the movements generated by the methods seen in Section 8.2.3 and Section 8.2.4. For example the user will know which direction the robot intends to cover next, as the end effector will move towards that area. However if the user is to make tactical decisions they must be aware of the *strategic* plan. Further, being aware of the *strategic* plan may help the user comply with the robots *tactical* plan when the robot is in control of that level of planning as in Section 8.2.4.The user could anticipate the types of movement that the robot will attempt next, perhaps allowing a better collaboration. Such a situation could be seen as the user generating a mental model of the robots intentions.

In order to communicate the *strategic* plan an AR (Augmented Reality) headset is used, specifically the Microsoft Hololens. Due to the fact that the robot will always apply heavy assistance at the *operational* level, it is not obvious that the user will need to see the low level detail of the task, perhaps a very simple visualisation would be sufficient for an effective collaboration between the user and the robot. This was one of the conclusions of Gregg-Smith and Mayol-Cuevas (2016*b*), they suggested that as long as the visualisation is adequate to get the robot in the correct vicinity, the quality of the visualisation may not be imperative.

To test this two visualisation styles have been developed. The first is a dense visualisation of the paths to be completed, and the other is a sparse representation consisting of a heatmap of where remaining paths are located. In order to facilitate depth perception and better alignment of the handheld robot with the drawing plane a secondary visualisation has been developed that will be applied in both cases.

8.3.1 *Dense Vector Line Visualisation*

This visualisation is simply the rendering of the lines that should be completed. Lines that still need to be drawn are rendered in red, and those that are completed are rendered in green. This is maintained by a stream of messages sent by the hand-held robot that contains the following information: line ID, start-completion, end-completion and a completed flag. The start and end-completion represent a parameterised progress along the line from the given end. For a straight line this is given as:

$$\mathbf{P}_{sc} = \mathbf{P}_{start} + s(\mathbf{P}_{end} - \mathbf{P}_{start}) \tag{8.3}$$

$$\mathbf{P}_{ec} = \mathbf{P}_{end} - e(\mathbf{P}_{start} - \mathbf{P}_{end}) \tag{8.4}$$

Where $\mathbf{P}_{sc}$ is the current progress from the start, and $\mathbf{P}_{ec}$ is the current progress from the end. $\mathbf{P}_{start}$ and $\mathbf{P}_{end}$ are the original start and end points of the line. s is the start-completion parametrisation, and e is the end-completion parametrisation. There is a similar parametrisation for both cubic Bezier lines and circles, though for brevity they are not included, though the principle is the same.

8.3.2 *Sparse Heatmap Visualisation*

The intention with the sparse heatmap visualisation is to remove the low level information of the task from the user. Due to the nature of the assistance discussed in Section 8.2.3 and Section 8.2.4, detailed actuation of the end effector is handled by the robot in both cases, and the user only needs to make sure lines to be drawn are reachable, and in the case where they are primarily in control they must know where uncompleted elements of the task are. Hence in this section a minimal visualisation that only provides as much information as is necessary is presented. An example of such a heatmap is shown in Figure 8.5.

A heatmap is generated that represents the density of lines available at that location. To accelerate the generation of the heatmap, the octree structure that is described in Section 8.2.3 is utilised. The N-nearest neighbours method, as described in Section 8.2.2 is used without any additional biasing points. The centre of each pixel of the heatmap is used as a search point, with a maximum search distance set to a multiple of the radius of the accessible area of the hand-held robot. This is necessary as the nearest neighbour

Figure 8.4: This shows the Vector Line (VL) visualisation, the upper image is a close up of a letter being drawn (robot omitted for clarity). The lower image is a view of the whole task with the letters "o" and "p" completed by the user. When viewing first hand through the Hololens, the lines appear more prominent than depicted.

can only return the end points of the lines, though the line could have some of its length within the pixel area, even if it starts and ends significantly outside the pixel area. All of the lines in the vicinity of the pixel are then stepped along, and the total length inside the pixel area is accumulated. This stepping approach is used such that all kinds of lines can be used. If only straight lines and circles were used, a closed form solution could be found, and would be significantly quicker.

The pixel size is chosen to be similar to the size of the accessible area of the end effector. Excessive resolution in the heatmap would provide the user with a similar density of information as the vector approach given in Section 8.3.1, and thus would not be informative for this study. Resolution too much lower than the accessible area could mislead the user, as the underlying lines to be drawn could be outside the accessible area of the robot, even if the user had aligned the robot with the pixel in question.

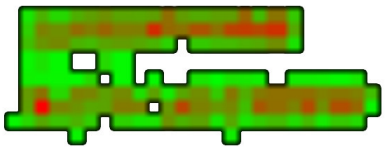


Figure 8.5: This is the heatmap that would be floating in the location of the drawing task when using the heatmap visualisation. The colours indicated the current line density at the location, red being more dense. However the colour does not impart much information as the tasks in the trial were very uniform in their line density.

8.3.3 *Secondary Visualisation: Reachable Area, Orientation and Depth Perception*

Preliminary experimentation exposed that users found the geometry of the robot confusing. They found it hard to judge where the middle of the accessible area was, further the size and shape of this area is unlikely to be obvious to a new user of the robot. A further problem is that despite the fact that the Hololens is a binocular system, perceiving depth can be challenging for some users. This is especially true when the visualisation is free of textured geometry and covers the whole field of view, which is often the case when using the heatmap visualisation discussed in Section 8.3.2.

A visualisation was developed to help solve these issues, this is shown in Figure 8.6, refer to the caption for details of the sub elements.

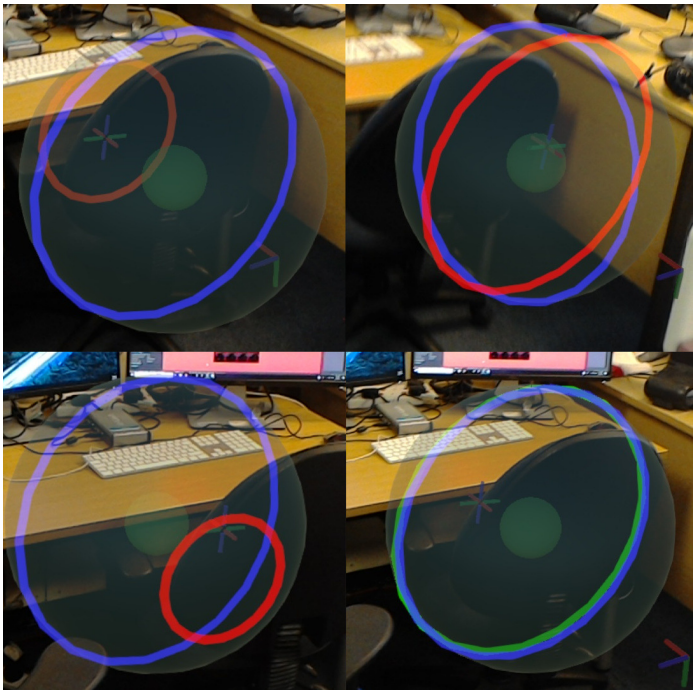


Figure 8.6: This figure shows the secondary visualisation, which is displayed in all of the trials. It is designed to aid the users depth perception, and perception of the reachable area of the robot.The larger translucent blue circle represents the reachable area of the robot. The red/green circle represents the intersection of the reachable area with the work plane, on which all the lines are to be drawn. The blue circle indicates the largest possible intersection of the reachable area. If the intersecting circle is close to the blue circle it is displayed as more green, if the robot is too close, far or angled in respect to the work plane, the circle is more red.

The intersection circle allows the user to know the degree to which the robot can reach the task plane. Therefore it is in the users interest to make this circle large and green, if they want the robot to have maximum freedom in assisting them with the task.

An additional small transparent sphere at the centre of the larger sphere helps the user gauge where the middle of the accessible area is. This is important if the user wants to take advantage of the bias of the user position based method, that prioritises lines near the centre of the accessible area.

The small cross shaped visualisation that represents the current location of the end effector is necessary because the actual end effector can be difficult to see through the visualisations. Both the tactical planning algorithms described in Sections 8.2.3 and 8.2.4 take the current location of the end effector as an input, and therefore for the user to make best use of these methods they should be aware of where the end effector is.

This secondary visualisation can be used to maximise the ability of the robot to assist. If the cross shaped visualisation is inside the smaller sphere, and the intersection circle is green and aligned with the static blue circle, then the robot has the maximum reach on the work space. The user can simply maintain this condition and the task will be completed using either tactical planners.

8.4 EXPERIMENTATION

The aim of the experiment is to come to an answer to two questions:

- Should the tactical plan be made by the user or the robot?
- Does the user benefit from having full knowledge of the state of the task?

8.4.1 *Experimental Design*

Given that there are two factors that are to be tested, visualisation type and *tactical* planning type, there are four possible combinations of factors. Control of the *tactical* planning will be assigned to the user and the robot, this is to investigate which is most accepted by the user, and if there is any performance advantages to either allocation.

The visualisations chosen were described in previous sections, the vector line visualisation is designed to give the user full information of the state of the task. The heatmap visualisation is designed to give minimal inform-

ation. These choices were made to see if performance was the same when using reduced detail visualisations, as was hypothesised by Gregg-Smith and Mayol-Cuevas (2016*a*).

The research questions are investigated with a cross over design trial with 8 arms to investigate system characteristics. Each user performs a drawing task with the robot with each of these four modes. To minimise the amount of change between trials, only one factor will be changed between trails. This means that there are 8 possible orders in which the experiments can take place, the particular order is assigned to the participants at the start of the trial.

The drawing task for each experiment is different, so that the user is discouraged from simply remembering the previous task and emulating it. Remembering the task would render reduction of information supplied to the user in the heatmap visualisation less effective as a test. All of the tasks are simply two word phrases, the order in which each phrase is presented is the same to each user. This means that the phrase-mode pairings are also balanced across the study. Latin words were chosen as the shapes of each letter would be familiar to participants, whilst the whole word is unlikely to have any particular valance with the user, whilst remaining pronounceable. Shape familiarity and pronounceable words are aimed to help the user remember the task, perhaps reducing the effect of the limited field of view on the Hololens headset. All the two word phrases have the same number of characters, and are spread over two lines, aligned on the left. The phrases are:

- Atropa Belladonna
- Abrus Precatorius
- Nicotiana Tabacum
- Argemone Mexicana

8.4.1.1 *Experimental Flow*

At the start of the experiment some limited personal information was collected from each user, such as age and sex, and their informed consent was obtained regarding the purpose of the experiment and the handling of the data.

The different modes were briefly described to the user in written format before the beginning of the testing. Before each round the participants are

told the details of the next trial, this is delivered automatically over the speakers of the Hololens in an attempt to reduce the variation in the perception of each mode that might be induced by manually describing this to the participant.

After each round the participants were asked to fill in a full NASA TLX survey Hart and Staveland (1988), this was using an application loaded on a touch screen tablet. At the very end of the experiment a general survey of their experience was recorded. This included rating from 1-5 their perception of *understanding*, *control* and *efficiency* of both assistance methods. The visualisations were rated on the measures: *understanding*, whether there was *adequate detail* and *comfort*. They were also asked for their preferred combination of modes.

The author also performed three sets of trials to provide a base line, as they are highly familiar with the robot and the task. This will provide an estimate of the fastest achievable completion time.

8.4.2 *Results*

The experiment consisted of 16 participants of which 15 were male, and 1 was female. The participants were drawn from those in and around the Personal Robotics Lab at Imperial College London. They had a an age range of 19 to 32 with an average age of 26.6 years. Most participants had had some experience with similar experiments and with augmented reality more generally, though none of the participants had had experience of the 5 axis version of the robot, or this particular task.

8.4.2.1 *Finish Time*

The time it took participants to complete each mode is summarised in Table 8.1 and Figure 8.7. Due to the drawing speed being fixed, there is a lower limit that is placed on finish time. This value averages as 151.7 seconds across the different task texts. The author also completed the trials as a comparison, they are experienced with the robot and the visualisations. The experienced user could finish in an average of 189.2s with very little variation across different modes. This proves that good performance can be achieved in all modes, and that the information provided to the user is adequate in all cases.

The users performed similarly in both the modes that used Vector lines as the visualisation method, which indicates that when provided with detailed information, the *tactical* planning mode is not important. However in the case where the user is presented with a heatmap, there is a large advantage

Table 8.1: A table showing the mean and standard deviation for both an expert and user trials accross the 4 tested modes, HM = Heatmap and VL = Vector Line are the visualisation modes, RC = Robot Control, UC = User Control are the tactical planning modes.. The expert, the author of this work, repeated the set of experiments 3 times, with all trials on one design, *Atrop Belladonna*. This represents the fastest time a user is likely to get with the different modes. It can be seen that there is little difference between the modes for the expert. For the participants, there is a much larger range of performances. This can be seen graphically in Figure 8.7. The last column shows the number of participants that chose that mode as their preferred one overall.

mode + vis	Expert N = 3		Participants N = 16		
	mean (s)	σ (s)	mean (s)	σ (s)	User Preference
HM + RC	189.2	2.45	348.0	159.6	2
VL + RC	185.9	0.74	262.9	97.9	8
HM + UC	194.2	4.76	452.6	248.6	1
VL + UC	187.4	5.10	274.5	77.82	5

to having the robot control the *tactical* plan. The mean completion time falls from 452.6s using the user control method to 348.0s with the robot control method ($p = 0.018$, Wilcoxon signed rank test).

The difference when changing between the heatmap visualisation and the vector line visualisation is large regardless of the control mode. The mean completion time was reduced by 85s when the robot was in control, and 178.1s when in user control mode, when switching to the vector line visualisations. This suggests that higher fidelity visualisations have a large positive effect performance of the user.

Some users could perform well even when not shown the detail of the task. This seems to be due to the some users leveraging the secondary visualisation to allow the robot to guide them to the nearest line. They did this at times even when they had primary control of the *tactical* plan. This can be seen as the user voluntarily giving the robot tactical control once they have moved to their preferred region of the task. This is a tactic that all of the better participants reported doing, when using the heatmap and user control.

All of the extreme outliers were due to either a misunderstanding of the visualisations, or a conflict in the tactical plan (based on observation and participant reports). For the heatmap modes, this was usually the users not fully understanding that they can use the secondary visualisation to infer where the nearest line is. In the robot control modes the outliers were primarily due to the users misunderstanding the "right to left" behaviour, believing that it

was a local behaviour rather than a global one. This highlights the fact that a mismatch in shared understanding negatively effects performance.

8.4.2.2 *Task Load*

Task load as measured by the NASA TLX survey can be seen as a box plot in Figure 8.8 and each factor can be seen separately in Figure 8.9. You can see that the magnitude of task load for the different modes across all users is in the same order as the finish times, VL+RC, VL+UC, HM+RC, HM + UC from best to worst. It seems that the longer the trial, the more physically tired users became due to the mass of the robot. This can be seen in the ratings of *physical* and *effort* in Figure 8.9. This was also reported by users at the end of the trial. The mix of task load factors across the different trials is broadly similar, with a notable difference is that users rated both Vector Line (VL) trials as less frustrating, and considered their performance to be less of a source of task load. This aligns with the users comments that they found the Vector Line (VL) visualisation much more comfortable, as can be seen in Table 8.2. It was found that the task load was significantly lower when using the vector line visualisation ($p = 0.0048$ for user control, $p = 0.016$ for robot control). There is no significant difference in total task load score between tactical planning methods ($p = 0.49$ for heatmap, $p = 0.44$ for vector line). This is confirmed by the survey responses shown in Table 8.3, as the responses to *Understand* and *Efficient* are similar. The users did report feeling less in *Control*, though this is the intended effect of moving the *tactical* plan from the user to the robot.

Table 8.2: Table showing the user ratings of the different visualisation methods. The ratings were 1 to 5 inclusive, with higher being more positive. You can see that the Vector Line (VL) visualisation is by far the preferred visualisation across all of the questions asked. This is to be expected, as the heatmap was designed to provide minimal information.

Visualisation Property	Vector Line		Heatmap	
	mean	σ	mean	σ
Understand	4.94	0.25	3.31	1.20
Detailed	4.69	0.60	2.75	1.06
Comfortable	4.88	0.34	2.63	0.96

8.4.2.3 *Experimental Flaws*

There were a small number of experimental flaws that may have effected the results.

Table 8.3: Table showing the user ratings of the different control methods. The ratings were 1 to 5 inclusive, with higher being more positive. You can see that the Robot Controlled (RC) method is preferred by a small margin, with the exception that users felt less in control, which is understandable due to the deliberate removal of agency from the user in this mode.

Control Property	Robot Control (RC)		User Control (UC)	
	mean	σ	mean	σ
Understand	4.06	0.68	3.81	1.11
In control	3.00	0.89	3.81	0.91
Efficient	4.13	0.72	3.94	1.00

- Using words as the target drawing encourage particular behaviour, such as completing a whole word before starting another. This caused some conflict when the robot was controlling the *tactical* plan.
- The narrow viewing angle of the Hololens caused users to miss areas, even though they were attempting to comply with the robot *tactical* plan.
- The weight of the robot is likely too much for trials that can take up to 1 hour. Most users complained about this, and this can be seen in the NASA TLX results in Section 8.4.2.2. This may have made other factors be underestimated.

8.5 CONCLUSION

In this chapter the allocation of the *tactical* plan to either the user or the hand-held robot is investigated. The fidelity of visualisations that act as a communication between the robotic system and the user are also analysed.

A user study was conducted to investigate whether the robot or the human should be in control of the *tactical* level plan. It was found that this depends on whether the user has access to high quality task state information. When detailed information was delivered over the AR headset the agent that makes the *tactical* plan does not matter. However if the system limits the quality of information delivered to the user then there is a significant benefit of allowing the robot to take a larger role in the *tactical* planning.

This has implications for the idea presented in Gregg-Smith and Mayol-Cuevas (2016*b*), that visualisations could be of low quality when using a collaborative robot, under the condition that they were sufficient to allow the user to position the robot in the vicinity of the task element.

This is shown to be circumstantial for this task. When the robot was in control of the *tactical* plan this was indeed the case for a subset of users, however performance of the users was increased by having access to the higher fidelity visualisations. This work can modify the previous conclusion: low quality visualisations are sufficient for users to complete a task with a collaborative handheld robot, though they are not necessarily optimal, or comfortable for the user.

The weight of the robot was a large factor and dominated the responses to the NASA TLX survey. Therefore hand-held robot designers should consider minimising weight of the robot as a primary design objective. The primary change in sources of task load were that of *frustration* and *performance*, being reduced when the user had access to the underlying vector visualisation. It was also shown that with adequate experience all of the tested modes perform equally in terms of completion time, both from the author experiments setting his own benchmarks, as well as a subset of the participants performing close to these benchmarks in three out of four of the modes.

Future work on this topic could investigate the effect on users of varying the relative wights in the *tactical* planning algorithms, denoted by f in Equations 8.2 and 8.1. These factors vary how aggressively the *tactical* plan is enforce, in this case "right to left" or "where the user is pointing". In this work these weights were chosen by the author subjectively, attempting to maximise the apparent smoothness of operation whilst enforcing the *tactical* plan.

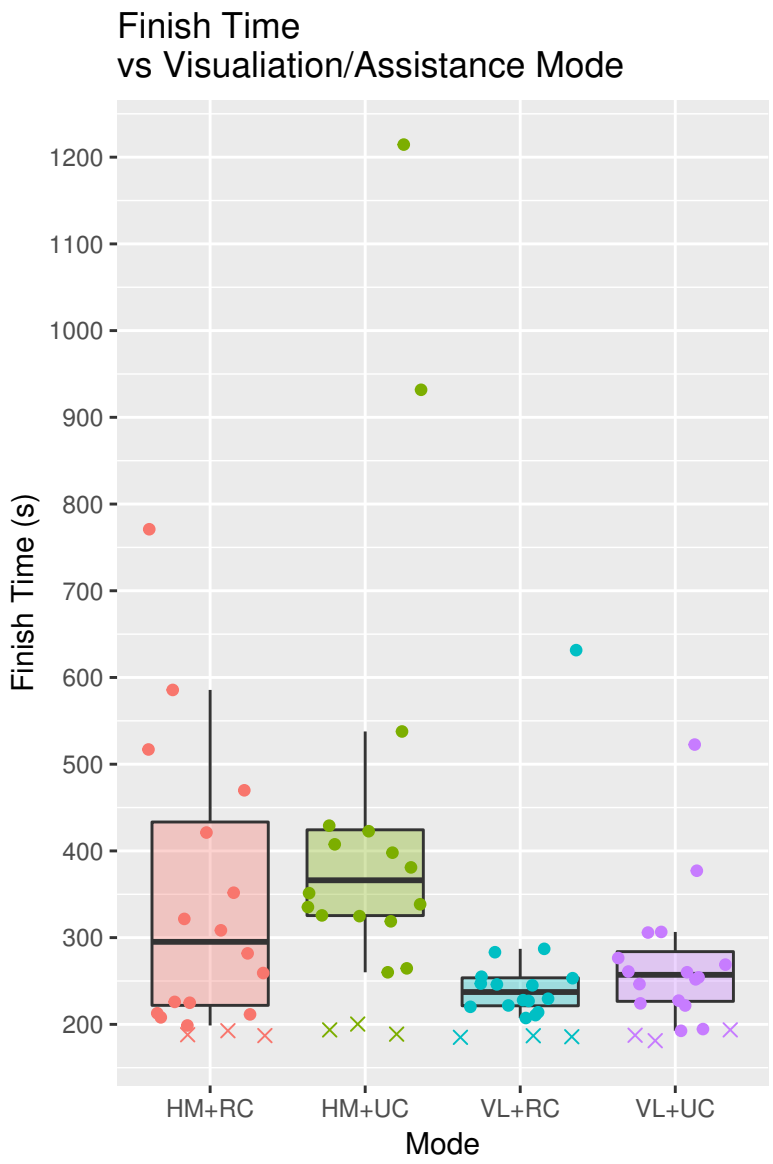


Figure 8.7: A boxplot showing the finish times of users for each of the different modes. It can be seen that the Vector Line (VL) visualisation performs the best out of the visualisations, and that the robot control (RC) also aids performance slightly. Though it should be noted that only user control with the heatmap (HM + UC) limited the performance of the fastest participants . For very experienced users there is almost no performance penalty, even for that mode expert values are displayed by the *X* points in this graph.

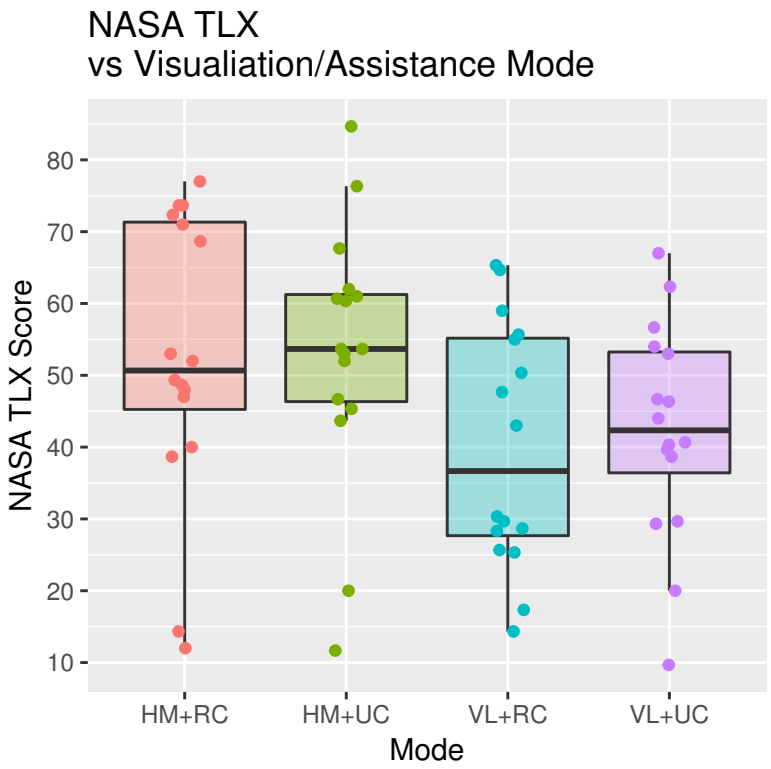


Figure 8.8: A boxplot of the total NASA TLX score given by the users for each of the different modes. HM = Heatmap and VL = Vector Line are the visualisation modes, RC = Robot Control, UC = User Control are the tactical planning modes. It can be seen that there is large disagreement between the users in how much task load is present in this experiment, as all modes have responses accross most of the scale. However it can be seen that there is a indication that the vector line visualisation and robot control of the tactical plan lowers the task load.

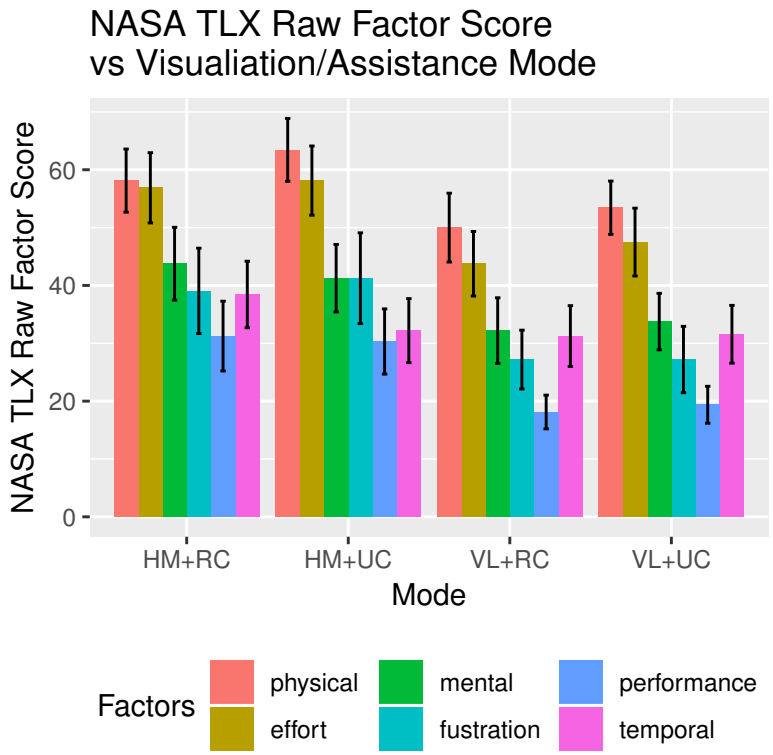


Figure 8.9: This is a bar chart showing the individual factors of the NASA TLX survey for each of the four combinations of modes. It can be seen that there is no large change in the composition of the sources of task load. Physical and effort dominates in all trails primarily due to the weight of the robot being taxing for most participants, further most reported that this physical load was a proxy for their perceived completion time, as they became tired in the longer experiments due to the weight. The most notable difference is that the frustration was much lower in the vector line modes. The error bars are the standard error of the mean (SEM). These are the unweighted NASA TLX factors.

9

CONCLUSIONS AND ANALYSIS

In this work two categories of hand-held robots were investigated, namely *locally capable* and *reduced degree of freedom* hand-held robots. Robotic hardware and software systems were developed for both, as well as user studies dedicated to parsing out some of the properties of these categories. It was found that:

Reduced degree of freedom hand-held robots

- These are robots that have fewer degrees of freedom than is required to fully correct human movements when applied to a particular task.
- These robots can be constructed to be significantly lighter than their more complex *locally capable* counterparts. The single axis robot used was only 443g compared to the five axis robot weighing 1407g. Whilst this is dependent on construction methods used and other specifications of the robot, in general the requirements of a *reduced degree of freedom* robot are less than their counterparts, and should be considered easier to produce in a light weight manner.
- Due to the fact that they cannot track arbitrary trajectories with the end effector without precise adherence to a plan by the user, it is not feasible for the robot to generate the *tactical* plan. The robot must follow the lead of the user to achieve a joint objective.
- In the case where a *reduced degree of freedom* robot is required to track a specific trajectory, the designer should expect errors in the range of 65mm linearly and 10 degrees in rotation for the directions that cannot be actuated by the robot. This applies when the trajectories are "body scale" movements.
- Similarly the inability to track trajectories independently means that the *operational* plan must be shared. This produces opportunity for conflict between the user and the robot. If there is conflict at the *operational* level then it is likely to spoil a user's *tactical* plan also.

Locally capable hand-held robots

- These are robots that have as many or more degrees of freedom than the task requires. The robot can completely decouple user motion from the end effector, if the desired location of the end effector remains within the reachable area. This allows them to track trajectories with some degree of independence from the user.
- If the system is to have the *tactical* plan decided by the robot, in most cases the robot should be *locally capable*. It is also possible to have the user decide the *tactical* plan with *locally capable* robots.
- When a user is supplied with high quality information about the state of the task it is not important which agent controls the *tactical* plan. However when only sparse information is supplied it is beneficial to users for the robot to form the *tactical* plan.
- In a task that is expecting users to move the robot at roughly the body scale, a *locally capable* robot should have the ability to decouple at least 65mm linearly and 10 degrees in rotation. A robot with $\pm$70mm range of motion linearly and $\pm$40 degrees in rotation was tested and users could complete tracking tasks successfully.

9.1 LIMITATIONS

This work presents a series of novel experiments with a number of handheld robots, investigating the interplay of AR, shared control and agency, and hardware design. As with all novel research, there are limitations with the work presented, highlighted below:

- **Weight:** The robots designed all targeted the single handed grip form factor. This was because this is a form factor that is natural to most who have used a power drill or a hot glue gun, and allows used the dexterity associated with the wrist. This is in contrast to the two handed form factor of the robots of Gregg-Smith (2016) , that are held in a manner similar to a leaf-blower.

  However due to the limitations of manufacturing facilities and time for design iteration, the robots were heavier than necessary.

  Especially in the case of the five axis robot used in Chapter 8, robot weight was unanimously indicated to the experimenter as a source of task load after the trial. This was the case even for participants that

were outwardly strong individuals. This factor meant that the subjective measure of task load (NASA TLX) is likely to have a reduced dynamic range in other task factors. Some users expressed that all other factors were not significant in their experience of task load, and that would bring into question their ability to carefully rate other factors, such as frustration, mental load etc.

- **Direct comparison between *reduced degree of freedom* and *locally capable* robots:** Not comparing *reduced degree of freedom* and *locally capable* robots directly was a deliberate design decision within the research. It is expected that the choice to implement one paradigm over the other will not be based on performance, but rather cost, weight or size. As the *locally capable* robots could emulate a *reduced degree of freedom* robot by inhibiting some axis of actuation if necessary. It was decided that more information could be gained studying other factors, such as which agent should own the *tactical* plan, or the effect of visualisation fidelity on collaboration between the user and the robot. These alternative investigations might help future designers of either category of robot. However a future designer may have need for a measure of the performance lost, if any, in moving to a *reduced degree of freedom* robot, such that they can see if it is an acceptable loss for their application. However it is likely that this is strongly dependent on the task at hand.

- **Inconsistency between tasks:** The task that the user is attempting to solve changes with each experiment. The first experiment was a real-world paint spraying experiment presented in Section 6.3.2, followed by a simulated spray painting exercise in Section 7.3 and finally a simulated drawing task in Section 8.4.

  The transition from real-world painting to simulated painting was made for three reasons. Firstly, painting was time consuming and unreliable. The inkjet described in Appendix A and the airbrush nozzle used in Section 6.3.2 had a tendency to require a careful cleaning routine to maintain performance, neglecting this would change the paint distribution enough that assumptions made in the planning algorithms would be incorrect. Secondly, the convenience of simulated paint is that the final distribution can be analysed much more easily. Finally, when performing user studies, as conducted in Section 5.2, 7.3 and 8.4, it is useful to ensure that the participants are not risking getting paint or other liquids on them or their clothes. After all these considerations it made sense to remove the physical spraying element from the work.

The transition from tasks that emulate spray painting to one that was tracking 3D trajectories was made for a different reason. Since exact trajectories can be traced with a *locally capable* robot, pre-planned painting trajectories could be generated with methods such as those discussed by Hegels et al. (2015). Then at run time only the trajectories are important. Finally the additional complication required to simulate and display the paint falling on a surface was not necessary to answering the question that the five axis robot was built to answer: Should the tactical plan be made by the user or the robot?

9.2 FUTURE WORK

This thesis highlights some fruitful avenues for future work.

- Investigate *tactical* plans that are generated by the user, though acted on actively by the robot, rather than a universally applied low level assistance algorithm as applied in this work (Section 8.2.3). For example, if the user could generate a plan *"I want to work from the middle outwards"*, the robot could then attempt to aid the user in that plan.
- Extend the trajectory tracking task in Section 8.4 to a full 3D surface, with changes in orientation also. This could increase complexity to the point that the user would prefer not be overwhelmed with the full state of the task, and perhaps take a more submissive role in the collaboration.
- Explore different task scales and the transition between them. The Micron project (MacLachlan et al., 2012) worked in the 10µm to 1mm range. The robots presented here and by Gregg-Smith (2016) were performing tasks in the 1m range, with expected precision of around 1cm. The robot form factors were optimised for these scales. It would be interesting to design a system that can work smoothly between such scales, and perhaps larger scales. An example application for having such a large dynamic range in task scales could be inspection tasks requiring fine positioning over a larger object (finding fractures in structural supports, inspecting circuit boards that are mounted to structures etc.).
- Applying the ideas presented more directly to real world tasks. An assisted eating system, similar to the Liftware[1] systems, with a locally

1 https://www.liftware.com/

capable robot to completely isolate user tremor could be an effective use of hand-held robotic ideas. Assistance could be adjusted based on the stage of the eating action.

- Refining the one piece parallel linkage presented in Section 3.3.1.1 to more degrees of freedom. A cheaply producible 4,5 and 6 degree of freedom linkage would be useful in cost sensitive robotic applications, such as one-time-use robots. This could be situations where the linkage is thrown away for sanitary reasons, such as if used in medical contexts. The body of the robot could be kept and a new linkage attached.

- Integrating a hand-held robotic system with more convenient tracking systems that can be moved to new locations more easily.

- Investigate multi-tasking with a hand-held robot when trajectory tracking is a key element of the task. For example, the work by Gregg-Smith (2016) used a task that was not demanding on user skill, rather loading the *strategic* planning element of the task.

- Investigate the range of task complexities that could be solved using the robotic gesturing method, as shown in Chapter 8 and in the work by Gregg-Smith and Mayol-Cuevas (2016*b*). If complex tasks could be completed without the need for an augmented reality headset or other cumbersome visualisation methods, system cost could be reduced significantly.

9.3 EPILOGUE

Hand-held robots could radically alter the set of tools available in many industries. Whether they are augmenting the ability of crafts people, or bridging the gap for those with reduced ability to use traditional hand tools, their weight, cost and complexity advantages will make them attractive options for future designers. This thesis should help these designers understand the some of the options open to them regarding the degree of actuation needed, as well as insight into how to interact and produce plans with the user.

AUTHOR'S PUBLICATIONS

Elsdon, J., and Demiris, Y. (2017), Assisted Painting of 3D Structures Using Shared Control with Under-actuated Robots, *in* 'IEEE International Conference on Robotics and Automation', pp. 4891–4897. doi: 10.1109/ICRA.2017.7989566.

- Presents a framework for calculating a suitable path for a hand held robot's spray nozzle, which has only one degree of freedom. This uses a receding horizon approach, where near future scenarios are simulated and selected based on a cost function.
- Chapter 6 is based on this article.

Elsdon, J., and Demiris, Y. (2018), Augmented Reality for Feedback in a Shared Control Spraying Task, *in* 'IEEE International Conference on Robotics and Automation', pp. 1939–1946. doi: 10.1109/ICRA.2018.8461179

- Presents a method for continuous registration of a head mounted augmented reality device.
- Outlines three levels of user assistance and compares the effectiveness of these using user performance and self reported task load data.
- Suggest a visualisation scheme for the user to view the progress of the spraying task using augmented reality.
- Section 7.2 is based on this work.

Zolotas, M , ***Elsdon, J.***, and Demiris, Y. (2018), Head-Mounted Augmented Reality for Explainable Robotic Wheelchair Assistance, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 1823-1829 doi: 10.1109/ICRA.2018.8461179

- An augmented reality system for aiding motorised wheelchair users is presented.
- A number of different visualisations are proposed and user experiments conducted to analyse their usefulness.
- It is found that visualisations that are too close to the user, and require significant head movement are not popular with users.

- Display elements that add extra capability to the user are the most popular, the example presented is a augmented reality rear view display. This was show to reduce head movement in reversing elements of the test stage.
- This work is tangential to this thesis, though is discussed as a alternative case study in Section 2.4.

Elsdon, J., and Demiris, Y. (2018), Augmented Reality Instructions for Shared Control of Hand-held Robotic Systems, *in* 'IEEE International Conference on Robotics and Automation, Workshop: Robotics in Virtual Reality'.

- Peer reviewed submission, followed by a in person workshop lecture.
- Presented and experiment that provided guidelines on a user's ability to control a hand-held robot.
- This data allows designers to know the range of motion that would be required for the end effector on a hand-held robot.
- This work strongly informs the design of the 5 DoF robot presented in Section 3.3.
- This work implies that precise trajectories cannot be tracked by *reduced degree of freedom* hand-held robot, as they will only be as accurate as the human, which this work shows to be poor for many tasks.

Works Not Published at Time of Print

Elsdon, J., and Demiris, Y. (2019), Visualisation and Agency in Hand-Held Robot Collaboration: Who's in Control and What Should we See?

- This work strongly informs Chapter 8.
- Presents a user study that tests two factors: Should the robot or the human have control of the *tactical* plan, and how does the fidelity of the visualisations effect this relationship.
- This work suggests that systems with poor visualisations can effectively facilitate collaborations between a human and a hand-held robot, however performance is significantly reduced and users experience is reported more negatively.

BIBLIOGRAPHY

Abbink, D. A., Carlson, T., Mulder, M., de Winter, J. C. F., Aminravan, F., Gibo, T. L. and Boer, E. R. (2018), 'A Topology of Shared Control Systems—Finding Common Ground in Diversity', *IEEE Transactions on Human-Machine Systems* pp. 1–17. doi: 10.1109/THMS.2018.2791570

Ang, W. T., Riviere, C. N. and Khosla, P. K. (2000), An Active Hand-Held Instrument for Enhanced Microsurgical Accuracy, *in* 'International Conference on Medical Image Computing and Computer-Assisted Intervention', Springer, Berlin, Heidelberg, pp. 878–886. doi: 10.1007/978-3-540-40899-4_91

Atkar, P. N., Greenfield, A., Conner, D. C., Choset, H. and Rizzi, A. A. (2005), 'Uniform Coverage of Automotive Surface Patches', *The International Journal of Robotics Research* **24**(11), 883–898. doi: 10.1177/0278364905059058

Becker, B. C., Maclachlan, R. A., Lobes, L. A., Riviere, C. N. and Riviere, C. N. (2012), Position-Based Virtual Fixtures for Membrane Peeling with a Handheld Micromanipulator., *in* 'IEEE International Conference on Robotics and Automation : ICRA', Vol. 2012, NIH Public Access, pp. 1075–1080. doi: 10.1109/ICRA.2012.6224844

Bouri, M. and Clavel, R. (2010), The Linear Delta: Developments and Applications, *in* 'ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)', [VDE Verlag], pp. 1198–1205.

Chadalavada, R. T., Andreasson, H., Krug, R. and Lilienthal, A. J. (2015), That's on my mind! robot to human intention communication through on-board projection on shared floor space, *in* '2015 European Conference on Mobile Robots (ECMR)', IEEE, pp. 1–6. doi: 10.1109/ECMR.2015.7403771

Chen, H., Fuhlbrigge, T. and Li, X. (2009), 'A review of CAD based robot path planning for spray painting', *Industrial Robot: An International Journal* **36**(1), 45–50. doi: 10.1108/01439910910924666

Clavel, R. (1990), 'Device for the movement and positioning of an element in space'.

Clavel, R. (1991), Conception d'un robot parallèle rapide à 4 degrés de liberté, PhD thesis, École Polytechnique Fédérale de Lausanne. doi: 10.5075/EPFL-THESIS-925

Davison, A. J., Reid, I. D., Molton, N. D. and Stasse, O. (2007), 'MonoSLAM: Real-Time Single Camera SLAM', *IEEE transactions on pattern analysis and machine intelligence* **29**(6), 1052 – 1067. doi: 10.1109/TPAMI.2007.1049

de Saint-Venant, M. (1856), *Mémoire sur la torsion des prismes: Avec des considérations sur leur flexion ainsi que sur l'équilibre intérieur des solides élastiques en général: Et des formules pratiques pour le calcul de leur résistance à divers efforts s' exerçant simultanément*, Imprimerie nationale.

Dijkstra, E. W. (1959), 'A note on two problems in connexion with graphs', *Numerische Mathematik* **1**(1), 269–271. doi: 10.1007/BF01386390

Elsdon, J. and Demiris, Y. (2018*a*), Augmented Reality for Feedback in a Shared Control Spraying Task, *in* '2018 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 1939–1946. doi: 10.1109/ICRA.2018.8461179

Elsdon, J. and Demiris, Y. (2018*b*), Augmented Reality Instructions for Shared Control of Hand-held Robotic Systems, *in* '2018 International Conference on Robotics and Automation (ICRA). Workshop: Robotics in Virtual Reality'.

Galceran, E. and Carreras, M. (2013), 'A survey on coverage path planning for robotics', *Robotics and Autonomous Systems* **61**(12), 1258–1276. doi: 10.1016/J.ROBOT.2013.09.004

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. and Marín-Jiménez, M. (2014), 'Automatic generation and detection of highly reliable fiducial markers under occlusion', *Pattern Recognition* **47**(6), 2280–2292. doi: 10.1016/J.PATCOG.2014.01.005

Gonenc, B., Tran, N., Gehlbach, P., Taylor, R. H. and Iordachita, I. (2016), Robot-assisted retinal vein cannulation with force-based puncture detection: Micron vs. the steady-hand eye robot, *in* '2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)', IEEE, pp. 5107–5111. doi: 10.1109/EMBC.2016.7591876

Gregg-Smith, A. (2016), Cooperative Handheld Robots, PhD thesis, University of Bristol.

Gregg-Smith, A. and Mayol-Cuevas, W. W. (2015), The design and evaluation of a cooperative handheld robot, *in* '2015 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 1968–1975. doi: 10.1109/ICRA.2015.7139456

Gregg-Smith, A. and Mayol-Cuevas, W. W. (2016*a*), Inverse kinematics and design of a novel 6-DoF handheld robot arm, *in* '2016 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 2102–2109. doi: 10.1109/ICRA.2016.7487359

Gregg-Smith, A. and Mayol-Cuevas, W. W. (2016*b*), Investigating spatial guidance for a cooperative handheld robot, *in* '2016 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 3367–3374. doi: 10.1109/ICRA.2016.7487512

Guan, Y., Chen, D., He, K., Liu, Y. and Li, L. (2015), Review on research and application of variable rate spray in agriculture, *in* '2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)', IEEE, pp. 1575–1580. doi: 10.1109/ICIEA.2015.7334360

Haggar, R., Stent, C. and Isaac, S. (1983), 'A prototype hand-held patch sprayer for killing weeds, activated by spectral differences in crop/-weed canopies', *Journal of Agricultural Engineering Research* **28**(4), 349–358. doi: 10.1016/0021-8634(83)90066-5

Hart, S. G. and Staveland, L. E. (1988), 'Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research', *Advances in Psychology* **52**, 139–183.

Hegels, D., Wiederkehr, T. and Müller, H. (2015), 'Simulation based iterative post-optimization of paths of robot guided thermal spraying', *Robotics and Computer-Integrated Manufacturing* **35**, 1–15. doi: 10.1016/j.rcim.2015.02.002

Ho, N. (2013), *Finding Optimal Rotation and Translation Between Corresponding 3D points*, http://nghiaho.com/?page_id=671.

Hughes, A., Malik, A. and Iles, D. C. (2012), *Design of Steel Beams in Torsion*, The Steel Construction Institute.

IEEE Robotics and Automation Society. Standing Committee for Standards Activities., Institute of Electrical and Electronics Engineers. and IEEE-SA Standards Board. (2015), *IEEE standard ontologies for robotics and automation*, IEEE.

Kalman, R. E. (1960), 'A New Approach to Linear Filtering and Prediction Problems', *Journal of Basic Engineering* **82**(1), 35. doi: 10.1115/1.3662552

Kim, D., Yoon, Y., Hwang, S., Lee, G. and Park, J. (2007), Visualizing Spray Paint Deposition in VR Training, *in* '2007 IEEE Virtual Reality Conference', IEEE, pp. 307–308. doi: 10.1109/VR.2007.352515

Kok, M., Hol, J. D. and Schön, T. B. (2017), 'Using Inertial Sensors for Position and Orientation Estimation', *Foundations and Trends in Signal Processing* **11**(2), 1–153. doi: 10.1561/2000000094

Konieczny, J. and Meyer, G. (2009), Airbrush simulation for artwork and computer modeling, *in* 'Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering - NPAR '09', ACM Press, pp. 61–69. doi: 10.1145/1572614.1572625

Larsen, T., Andersen, N., Ravn, O. and Poulsen, N. (1998), Incorporation of time delayed measurements in a discrete-time Kalman filter, *in* 'Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)', Vol. 4, IEEE, pp. 3972–3977. doi: 10.1109/CDC.1998.761918

Lung-Wen Tsai, Walsh, G. and Stamper, R. (1996), Kinematics of a novel three DOF translational platform, *in* 'Proceedings of IEEE International Conference on Robotics and Automation', Vol. 4, IEEE, pp. 3446–3451. doi: 10.1109/ROBOT.1996.509237

MacLachlan, R. A., Becker, B. C., Cuevas Tabarés, J., Podnar, G. W., Lobes, L. A. and Riviere, C. N. (2012), 'Micron: An Actively Stabilized Handheld Tool for Microsurgery', *IEEE Transactions on Robotics* **28**(1), 195–212. doi: 10.1109/TRO.2011.2169634

MacLachlan, R. A., Becker, B. C. and Riviere, C. N. (2009), Control of an active handheld instrument for microsurgery and micromanipulation, *in* 'IEEE International Conference on Robotics and Automation, ICRA', pp. 29–31.

MacLachlan, R. A., Hollis, R. L., Martel, J. N., Lobes, L. A. and Riviere, C. N. (2017), Toward Improved Electromagnetic Tracking for Handheld Robotics, *in* 'Proceedings of the 3rd International Conference on Mechatronics and Robotics Engineering - ICMRE 2017', ACM Press, pp. 75–80. doi: 10.1145/3068796.3068823

Madyastha, V., Ravindra, V., Mallikarjunan, S. and Goyal, A. (2011), Extended Kalman Filter vs. Error State Kalman Filter for Aircraft Attitude Estimation, *in* 'AIAA Guidance, Navigation, and Control Conference', Guidance, Navigation, and Control and Co-located Conferences, American Institute of Aeronautics and Astronautics. doi: doi:10.2514/6.2011-6615

Meagher, D. (1980), Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer, Technical report, Rensselaer Polytechnic Institute.

Merriaux, P., Dupuis, Y., Boutteau, R., Vasseur, P., Savatier, X., Merriaux, P., Dupuis, Y., Boutteau, R., Vasseur, P. and Savatier, X. (2017), 'A Study of Vicon System Positioning Performance', *Sensors* **17**(7), 1591. doi: 10.3390/s17071591

Michon, J. A. (1985), A Critical View of Driver Behavior Models: What Do We Know, What Should We Do?, *in* 'Human Behavior and Traffic Safety', Springer US, Boston, MA, pp. 485–524. doi: 10.1007/978-1-4613-2173-6_19

Mirshekari, E., Ghanbarzadeh, A. and Shirazia, K. H. (2016), 'Structure Comparison and Optimal Design of 6-RUS Parallel Manipulator Based on Kinematic and Dynamic Performances', *Latin American Journal of Solids and Structures* **13**(13), 2414–2438. doi: 10.1590/1679-78252937

Prévost, R., Jacobson, A., Jaroszb, W. and Sorkine-Hornung, O. (2016), 'Large-Scale Spray Painting of Photographs by Interactive Optimization', *Computers & Graphics* **55**, 108–117.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R. and Ng, A. (2009), ROS: an open-source Robot Operating System, *in* 'Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics'.

Roumeliotis, S., Sukhatme, G. and Bekey, G. (1999), Circumventing dynamic modeling: evaluation of the error-state Kalman filter applied to mobile robot localization, *in* 'Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)', Vol. 2, IEEE, pp. 1656–1663. doi: 10.1109/ROBOT.1999.772597

Roumeliotis, S., Sukhatme, G. and Bekey, G. (n.d.), Circumventing dynamic modeling: evaluation of the error-state Kalman filter applied to mobile robot localization, *in* 'Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)', Vol. 2, IEEE, pp. 1656–1663. doi: 10.1109/ROBOT.1999.772597

Samet, H. (2008), 'K-Nearest Neighbor Finding Using MaxNearestDist', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **30**(2), 243–252. doi: 10.1109/TPAMI.2007.1182

Seriani, S., Cortellessa, A., Belfio, S., Sortino, M., Totis, G. and Gallina, P. (2015), 'Automatic path-planning algorithm for realistic decorative robotic painting', *Automation in Construction* **56**, 67–75. doi: 10.1016/J.AUTCON.2015.04.016

Singh, M., Haverinen, H. M., Dhagat, P. and Jabbour, G. E. (2010), 'Inkjet Printing-Process and Its Applications', *Advanced Materials* **22**(6), 673–685. doi: 10.1002/adma.200901141

Sola, J. (2017), Quaternion kinematics for the error-state Kalman filter, Technical report, Institut de Robòtica i Informàtica Industrial.

Tuceryan, M., Genc, Y. and Navab, N. (2002), 'Single-Point Active Alignment Method (SPAAM) for Optical See-Through HMD Calibration for Augmented Reality', *Presence: Teleoperators and Virtual Environments* **11**(3), 259–276. doi: 10.1162/105474602317473213

Tuceryan, M., Greer, D., Whitaker, R., Breen, D., Crampton, C., Rose, E. and Ahlers, K. (1995), 'Calibration requirements and procedures for a monitor-based augmented reality system', *IEEE Transactions on Visualization and Computer Graphics* **1**(3), 255–273. doi: 10.1109/2945.466720

Uhlmann, J. K. (1992), 'Algorithms for Multiple-Target Tracking', *American Scientist* **80**, 128–141.

Uneri, A., Balicki, M. A., Handa, J., Gehlbach, P., Taylor, R. H. and Iordachita, I. (2010), New steady-hand Eye Robot with micro-force sensing for vitreoretinal surgery, *in* '2010 3rd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics', IEEE, pp. 814–819. doi: 10.1109/BIOROB.2010.5625991

Walker, M., Hedayati, H., Lee, J. and Szafir, D. (2018), Communicating Robot Motion Intent with Augmented Reality, *in* 'Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction - HRI '18', ACM Press, pp. 316–324. doi: 10.1145/3171221.3171253

Wells, T. S., MacLachlan, R. A. and Riviere, C. N. (2014), Toward hybrid position/force control for an active handheld micromanipulator, *in* '2014 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 772–777. doi: 10.1109/ICRA.2014.6906942

Wu, L.-C., Lin, I.-C. and Tsai, M.-H. (2016), Augmented reality instruction for object assembly based on markerless tracking, *in* 'Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games - I3D '16', ACM Press, pp. 95–102. doi: 10.1145/2856400.2856416

Xaar (2014), 'User Manual - High Performance Ink Jet Printhead'.

Yanco, H. and Drury, J. (2004), Classifying human-robot interaction: an updated taxonomy, *in* '2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)', Vol. 3, IEEE, pp. 2841–2846. doi: 10.1109/ICSMC.2004.1400763

Yang, S., MacLachlan, R. A. and Riviere, C. N. (2015), 'Manipulator Design and Operation of a Six-Degree-of-Freedom Handheld Tremor-Canceling Microsurgical Instrument', *IEEE/ASME Transactions on Mechatronics* **20**(2), 761–772. doi: 10.1109/TMECH.2014.2320858

Zolotas, M., *Elsdon, J.* and Demiris, Y. (2018), Head-Mounted Augmented Reality for Explainable Robotic Wheelchair Assistance, *in* '2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 1823–1829.

A

CONSTRUCTION OF AN INKJET HAND-HELD ROBOT

In addition to the single axis hand-held robot, presented in Section 3.2, and the five axis hand-held robot, presented in Section 3.3, the author also constructed a robot based on Inkjet technology. Inkjet spraying was a natural candidate for such a robot due to its compact size, lack of moving parts and no need for compressed air.

This thread of research however was not used in any of the experiments described in this thesis, or central to any of the arguments. However, some details of the robot are presented here, as they could be useful for future researchers and designers of hand-held robotic systems for spraying applications.

A.1 SYSTEM DETAILS

Inkjet is a method of liquid dispensing based one small chambers that are compressed with piezo electric elements, or in a variation sometimes known as *bubble-jet* a heater boils some component part of the liquid causing a sudden rise in pressure, and the ejection of a droplet from the nozzle. Singh et al. (2010) provide a review of the applications and technology behind this method of liquid deposition.

In this work a Xaar (2014) print head, model Xaar 128/80, was used. This print head has 128 individually controllable nozzles, the droplet size is 80pL. The availability of many nozzles gives this spraying method effectively one degree of freedom when used in a spraying task, as a selection of nozzles can be used separately to simulate moving a single or group of nozzles along a short axis. However, due to the fact that the print nozzle array is only 17.4 mm long, the degree of freedom is only useful on the small scale.

The driving electronics for the print head are held internally, however to use this print head on a hand-held robot, communication electronics to interface with the print head were developed as commercial options were to large.

The print head receives whether each head should eject a droplet in a serial manner, then all droplets are released using a single signal. For the

purposes of debugging an LED display that used the same communication signals was developed, and is shown in Figure A.1.

An Odroid XU3 single board computer provides the local processing and platform for communicating to a broader system using ROS. This then communicates to an microprocessor over USB, which manages the timing and output of signals to the print head. The microprocessor also samples the user buttons and forwards them to the ROS system. The microprocessor is mounted to a custom Printed Circuit Board (PCB) that also provides additional power supplies, both 5V and 35V, which are derived from the on board lithium polymer batteries.

On the front of the robot there is a camera with a wide angle lens, to be used in identifying regions of interest for the spraying task.

A.2 INKJET ROBOT USAGE AND CONCLUSIONS

This robot was highly compact, however for the purposes of conducting research it had some significant short comings. The most major issue being the fact that maintaining the Inkjet head for intermittent usage was unreliable. Ink would dry in the nozzles and cause them to block. A secondary issue is that improper purging of air from the system would lead to a section of nozzles being blocked. Due to the nature of a hand-held robot, the orientation at any time is not well defined and as such it was not possible to strictly follow the guidelines in the manual regarding ink pressure. The manual recommended having a small negative pressure on the ink supply, as the head can act as a pump. The recommended values are a ink pressure of between -0.1Kpa and -0.4Kpa, or between -1cm and -4cm hydro-static head using typical ink. The method that was used in this robot was to use a thin supply tube to the Inkjet head, such that the friction would provide some resistance to ink flow, however this was unreliable. It lead to both starving the head of ink and unwanted leakage of ink. As such this system was too unreliable to continue experimentation, though with some careful attention to the liquid supply pressure regulation, such a system could be of use to those wanting to emulate the techniques supplied in this work on a smaller scale. Construction of a reliable Inkjet system would be possible with industrial partners who were knowledgeable regarding the ink supply systems for Inkjet.

Figure A.1: The custom LED array used for debugging the XAAR 128 interface.

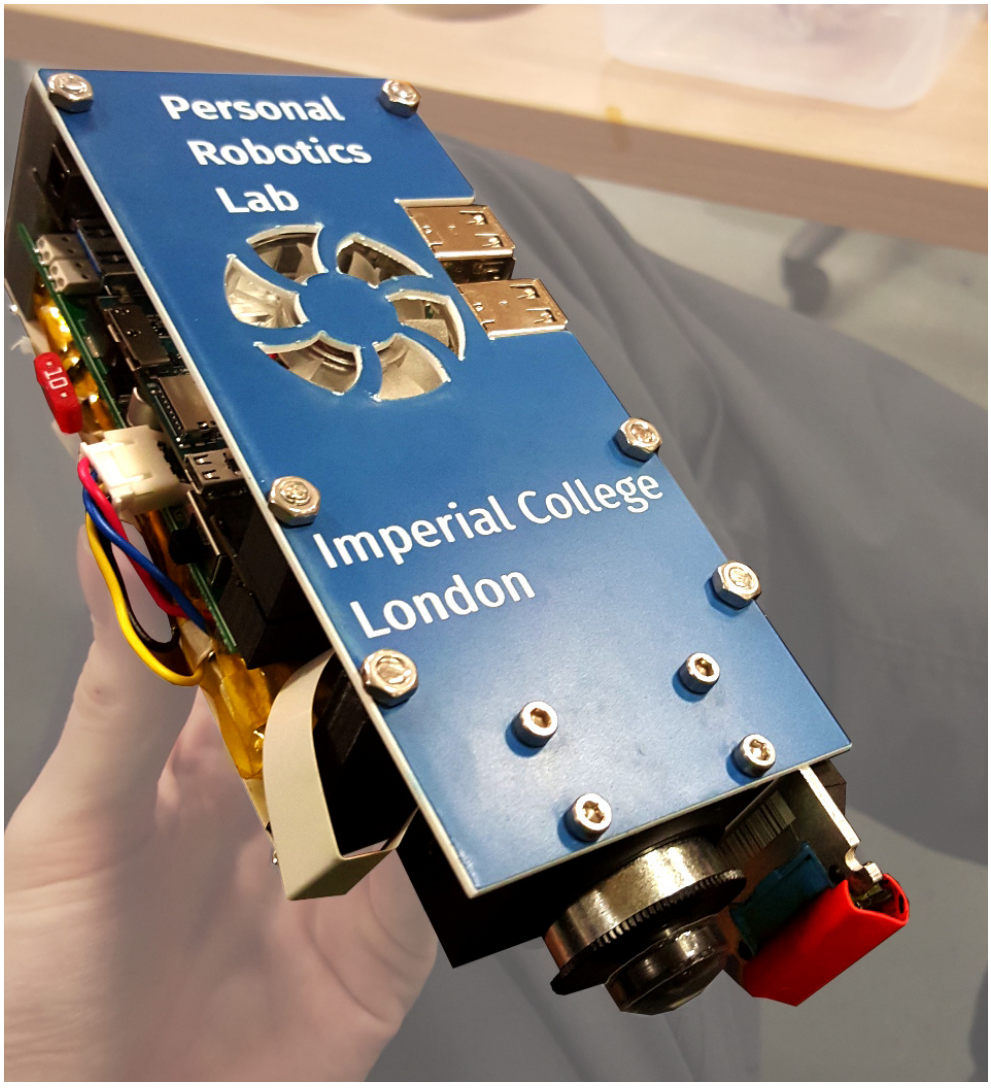


Figure A.2: An diagonal view of the inkjet hand-held robot.

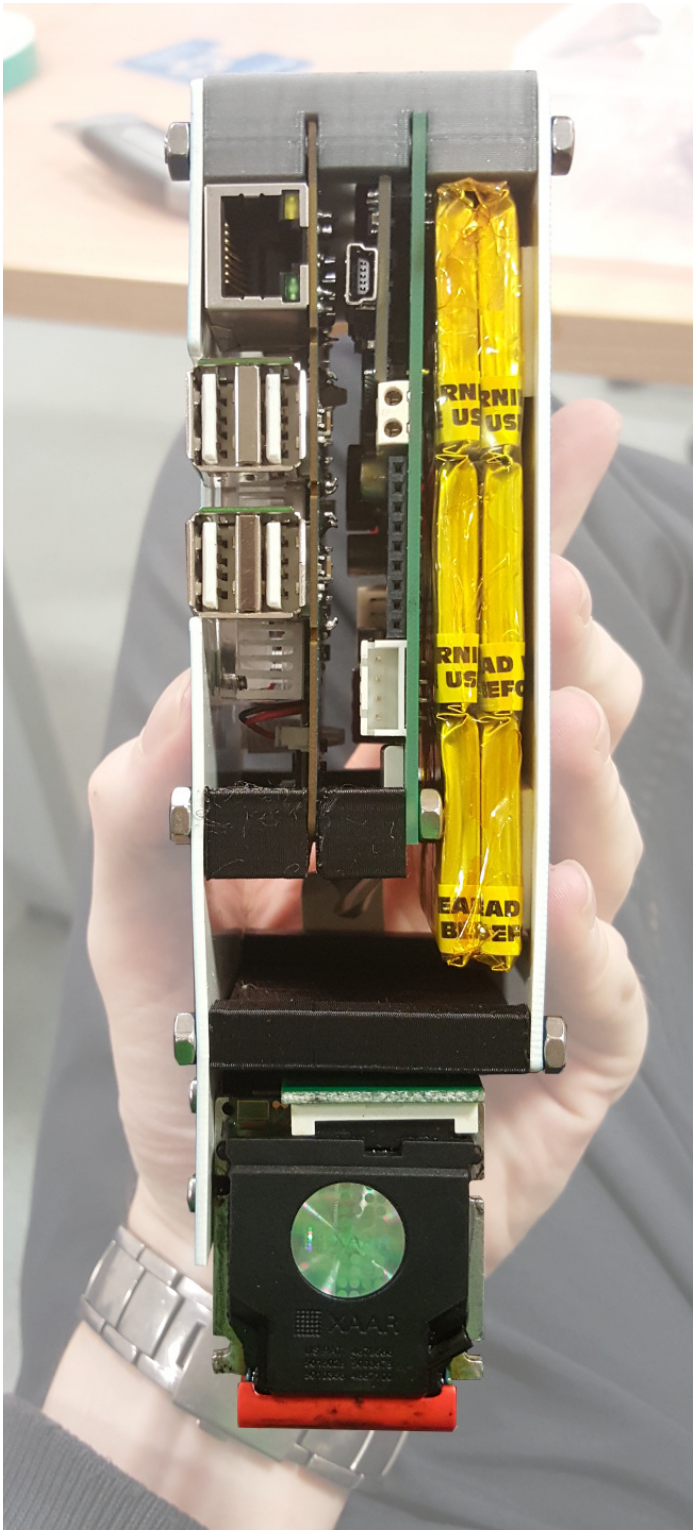


Figure A.3: A side view of the inkjet hand-held robot.

B

QUESTIONNAIRES

This appendix lists the printed questionnaires that were used in the experiments in this work. See the captions for details of which chapter each is relevant to. In the work presented in Chapter 8 the NASA TLX survey was completed on an application installed on a tablet.

Questionnaire for 5DoF Drawing Experiment

This survey will be used to find your opinion on the methods tested in this experiment.

You are candidate:

Prior to the Experiment

1. **Please enter the following personal information.**
   Gender:
   Age:
2. **Do you have any known disabilities?**
3. **Are you left handed or right handed:** □ Left □ Right
4. **Do you use eye correction:** □ Glasses □ Contacts □ None
5. **Do you have any prior experience with similar spraying experiments?**
   - □ Simulated spraying experiments without augmented reality feedback.
   - □ Simulated spraying experiments with augmented reality feedback.
   - □ None
6. **Do you have any prior experience with augmented reality?** □ Yes □ No

---

After the Experiment

0.1 Robot Right to Left Behaviour.

7. **Rate the following (1 – Strongly Disagree, 5 – Strongly Agree):**
   I *understood* how the robot was assisting
   □1 □ 2 □ 3 □ 4 □ 5
   I felt in total *control* of the drawing task
   □1 □ 2 □ 3 □ 4 □ 5
   I felt I could be *efficient* in the drawing task
   □1 □ 2 □ 3 □ 4 □ 5
   With this behaviour I prefered the:
   □Heatmap style visualisation □ The vector line visualisation

0.2 User Location Based Behaviour.

8. **Rate the following (1 – Strongly Disagree, 5 – Strongly Agree):**
   I *understood* how the robot was assisting
   □1 □ 2 □ 3 □ 4 □ 5

Figure B.1: Page 1 of the questionnaire used in Chapter 8

I felt in total *control* of the drawing task

□1 □ 2 □ 3 □ 4 □ 5

I felt I could be *efficient* in the drawing task

□1 □ 2 □ 3 □ 4 □ 5

With this behaviour I preferred the:

□Heatmap style visualisation □ The vector line visualisation

0.3 Heatmap Visualisation

9. Rate the following (1 – Strongly Disagree, 5 – Strongly Agree):

I found the visualisation easy to *understand*

□1 □ 2 □ 3 □ 4 □ 5

The visualisation gave me *adequate detail* to complete the task

□1 □ 2 □ 3 □ 4 □ 5

I felt the visualisation was *comfortable* to look at

□1 □ 2 □ 3 □ 4 □ 5

0.4 Vector Line Visualisation

10. Rate the following (1 – Strongly Disagree, 5 – Strongly Agree):

I found the visualisation easy to *understand*

□1 □ 2 □ 3 □ 4 □ 5

The visualisation gave me *adequate detail* to complete the task

□1 □ 2 □ 3 □ 4 □ 5

I felt the visualisation was *comfortable* to look at

□1 □ 2 □ 3 □ 4 □ 5

0.5 Overall

11. My preferred combination was:

□Heatmap + right-to-left
□ Vector line + right-to-left
□ Heatmap + user-location-based
□ Vector line + user-location-based

General Feedback:

Figure B.2: Page 2 of the questionnaire used in Chapter 8

Spraying Experiment with Augmented Reality Feedback Instructions to Participants

Joshua Elsdon[1] and Yiannis Demiris[1*]

February 17, 2018

1 Data Storage

Data collected in this experiment will be stored with your participant ID only.

2 Safety Considerations

- The experiment requires the use of trailing wires, these represent a trip hazard and you should move around the experimental area with care.
- Whilst wearing the augmented reality headset your vision will be somewhat impaired, so special care must be taken to avoid collisions.
- Some people feel nauseous after wearing the augmented reality headset for a short while, let the demonstrator know and the experiment will be stopped. These effects are short lived.
- The gun is heavy, please consider holding it with two hands.

3 Experiment Guide

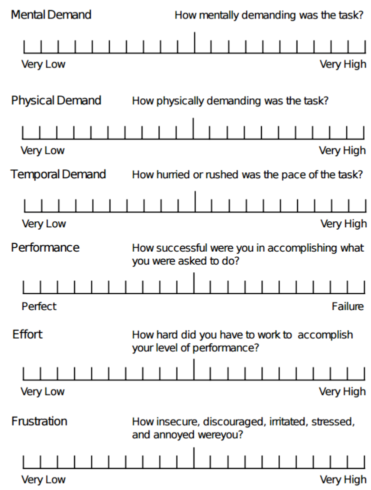
1. You will fill out a short questionnaire about yourself.
2. You will be shown a short video. This is primarily to demonstrate the colour scheme, and the action of the different modes.
3. Your inter-ocular distance will be measured.
4. You will be given the headset, and asked to walk a short distance.
5. You will be given a minute to practise with the gun.
6. You will be presented with 4 spraying experiments per mode (12 total experiments). You should continue until you are satisfied with the spray coverage. Your time is being recorded.
7. You will complete a questionnaire grading the workload of the tasks.
8. End of experiment.

*[1]Joshua Elsdon, je10@ic.ac.uk, and Yiannis Demiris, y.demiris@ic.ac.uk, are with the Department of Electrical and Electronic Engineering, Imperial College London,

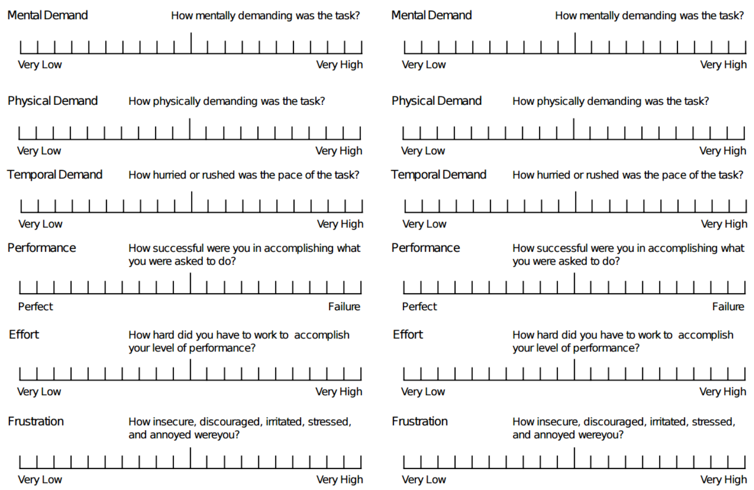
Figure B.3: Page 1 of the form given to users for experiments in Chapter 7

TITLE	ENDPOINTS	DESCRIPTIONS
MENTAL DEMAND	Low/High	How much mental and perceptual activity was required (e.g. thinking, deciding, calculating, remembering, looking, searching, etc.)? Was the task easy or demanding, simple or complex, exacting or forgiving?
PHYSICAL DEMAND	Low/High	How much physical activity was required (e.g. pushing, pulling, turning, controlling, activating, etc.)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?
TEMPORAL DEMAND	Low/High	How much time pressure did you feel due to the rate or pace at which the task or task elements occurred? Was the pace slow and leisurely or rapid and frantic?
PERFORMANCE	Good/Poor	How successful do you think you were in accomplishing the goals of the task set by the experimenter? How satisfied were you with your performance in accomplishing these goals?
EFFORT	Low/High	How hard did you have to work (mentally and physically) to accomplish your level of performance?
FRUSTRATION LEVEL	Low/High	How insecure, discouraged, irritated, stressed, and annoyed versus secure, gratified, content, relaxed, and complacent did you feel during the task?

(a) **Definitions**



(b) **Manual**



(c) **Semi-Auto** (d) **Full-Auto**

Figure 1: Surveys to be completed at the end of each task.

Figure B.4: Page 2 of the form given to users for experiments in Chapter 7

C

SYSTEM SCHEMATICS

This appendix lists the high level schematics for the three hand-held robots presented in this work, as well as some electrical schematics that were used to construct PCBs for the Inkjet hand-held robot. Analysis of these is outside the scope of the thesis, and are presented here for completeness.

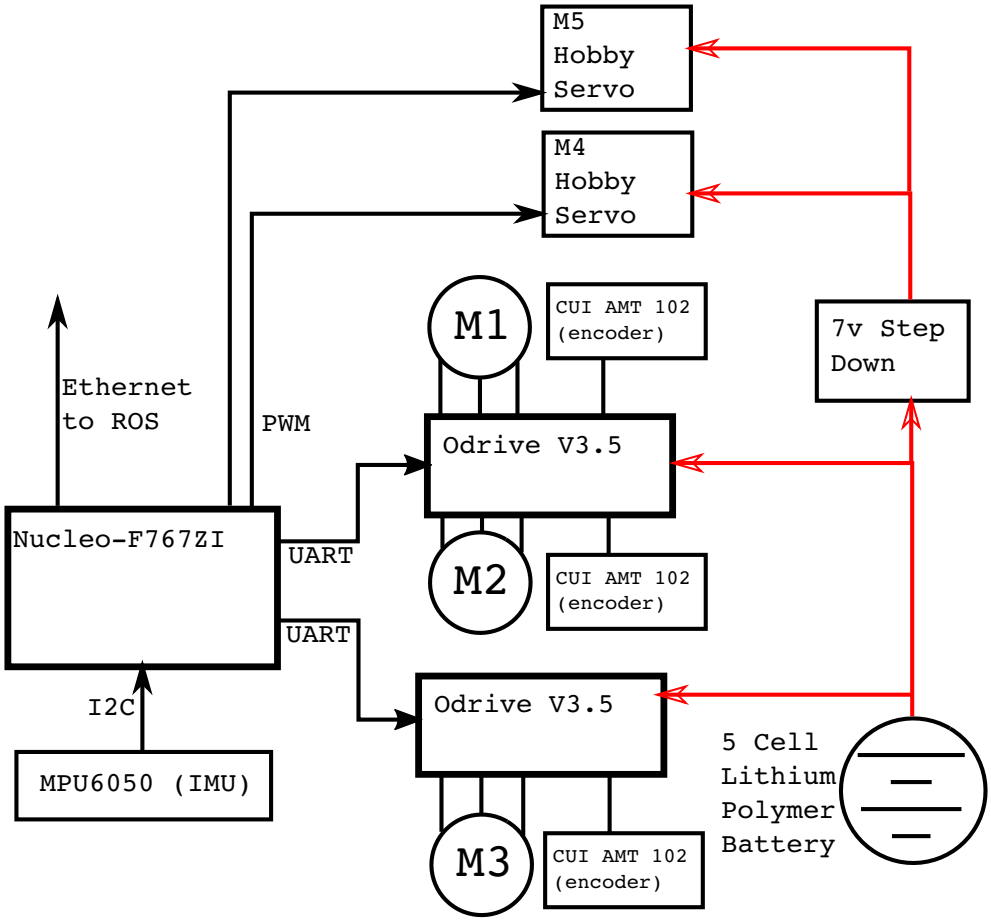


Figure C.1: High level electrical schematic of the 5 axis hand-held robot. Red arrows indicate power supply connections.

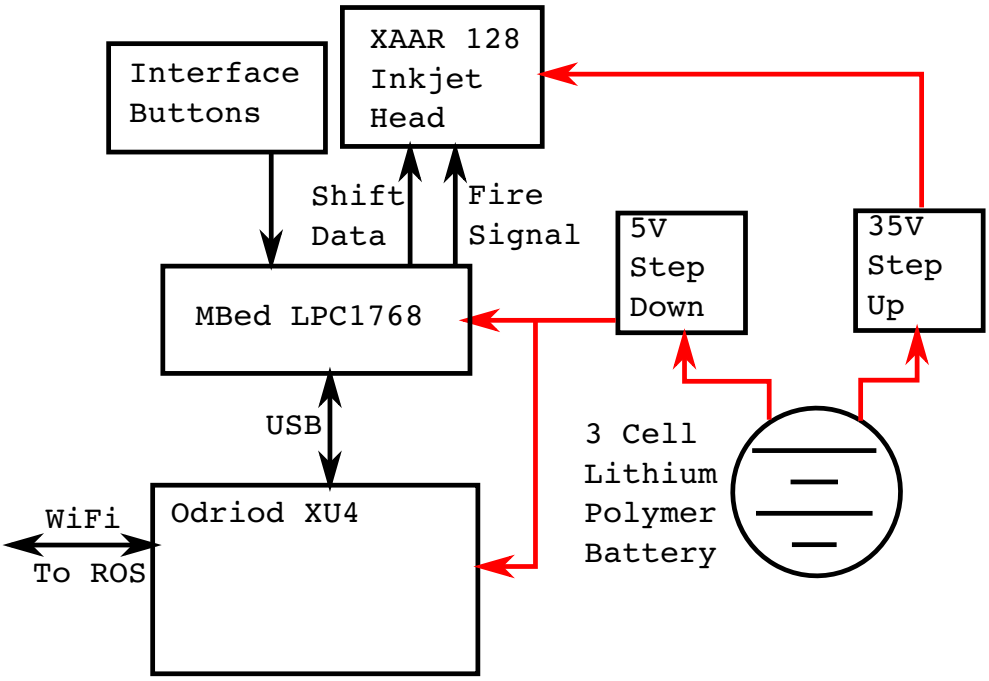


Figure C.2: High level electrical schematic of the Inkjet based hand-held robot. Red arrows indicate power supply connections.

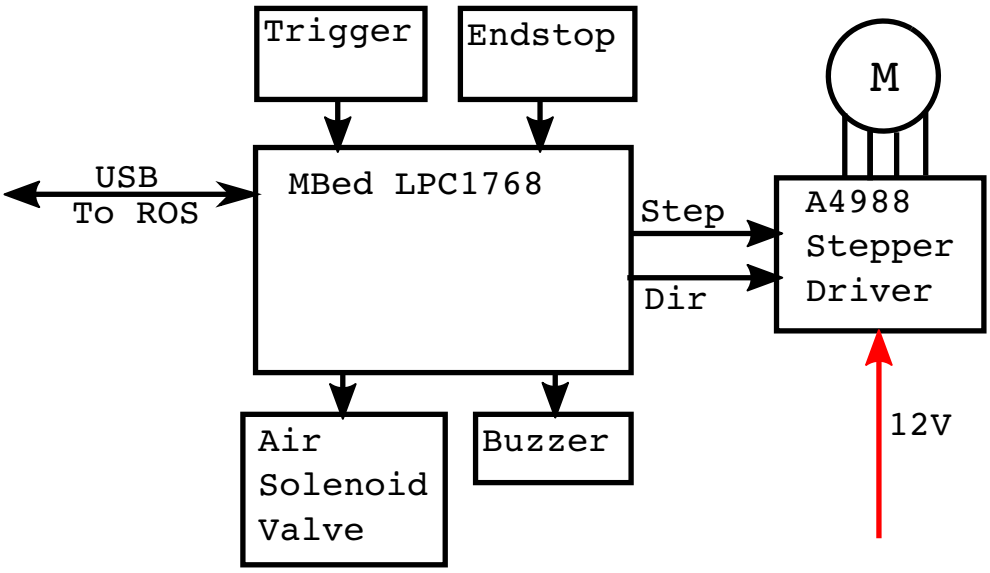


Figure C.3: High level electrical schematic of the single axis hand-held robot. Red arrows indicate power supply connections.

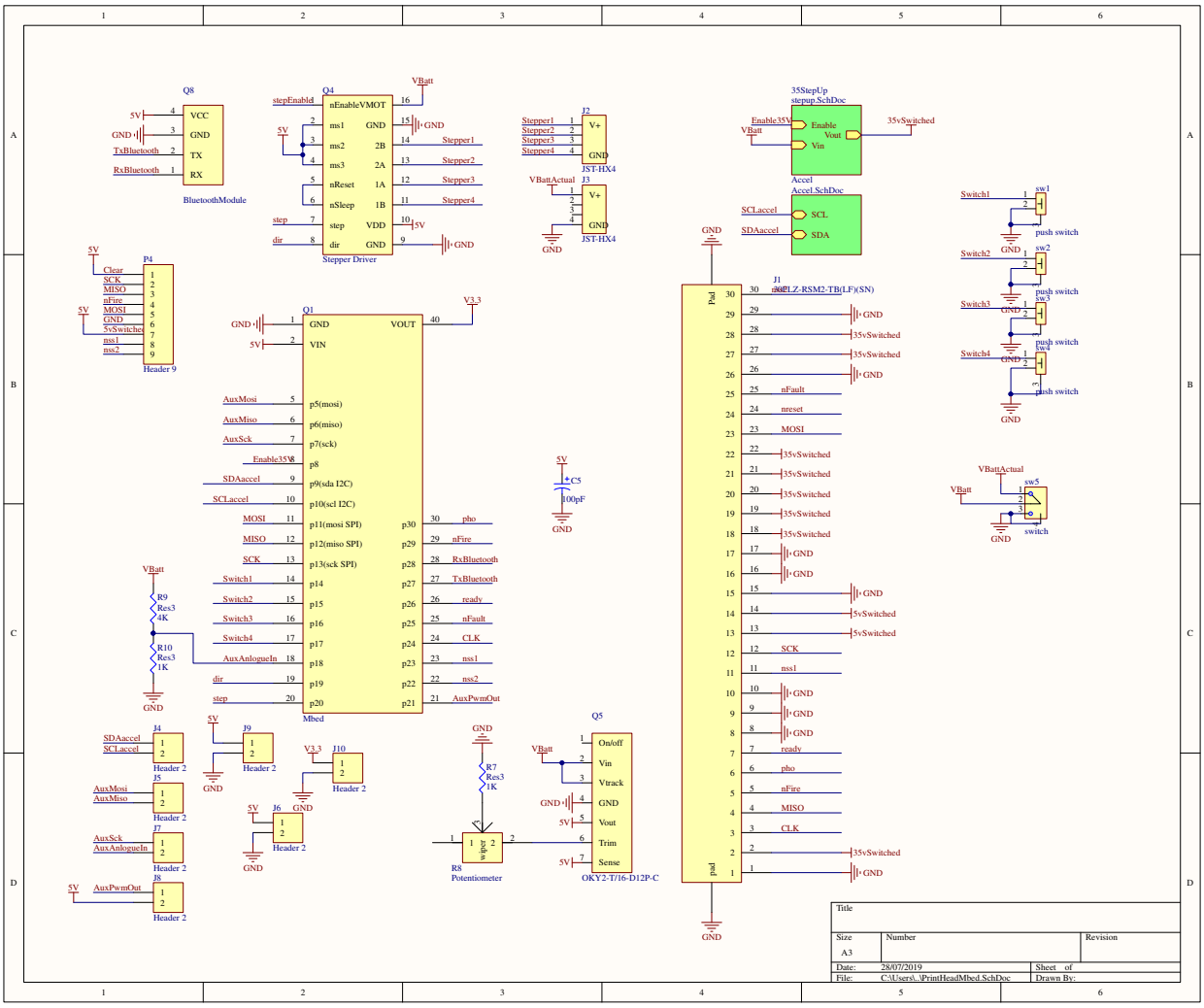


Figure C.4: This schematic shows the electrical connections used to define the custom PCB for the Inkjet hand-held robot. The 35v power supply block can be seen in Figure C.5. The accelerometer block was left un-populated.

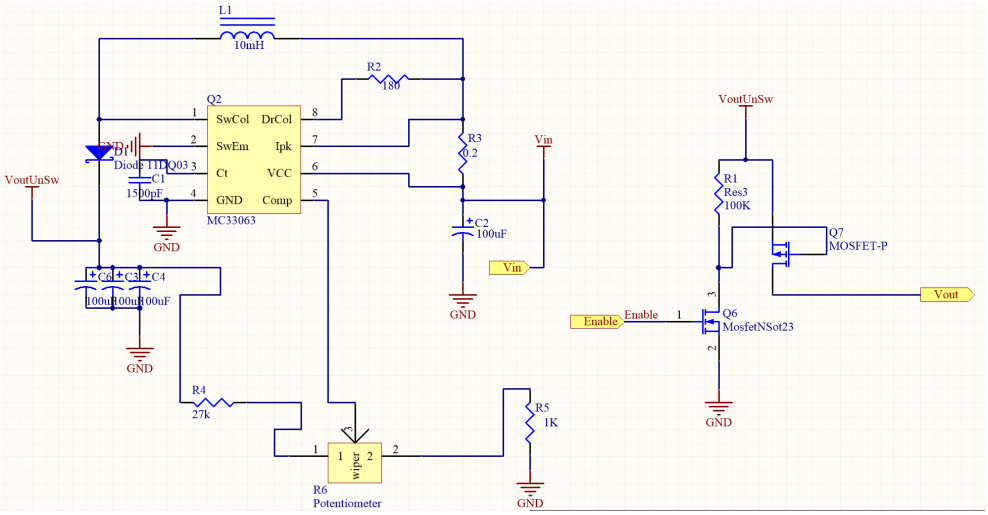


Figure C.5: The electrical schematics for the 35v power supply used for the Inkjet hand-held robot. This generates the 35v needed to drive the piezo electric elements. This is a sub-schematic of the system presented in Figure C.4.

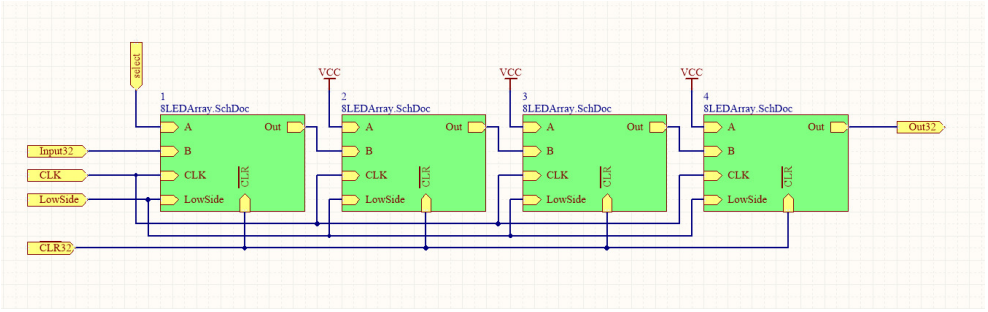


Figure C.6: The electrical schematics for the shift register visual debugger for the Inkjet robot. The individual shift register sections can be seen in Figure C.7. Two copies of this design are joined together to simulate the 128 nozzles of the XAAR128 inkjet head. The output of the first copy feeds the input of the second copy.

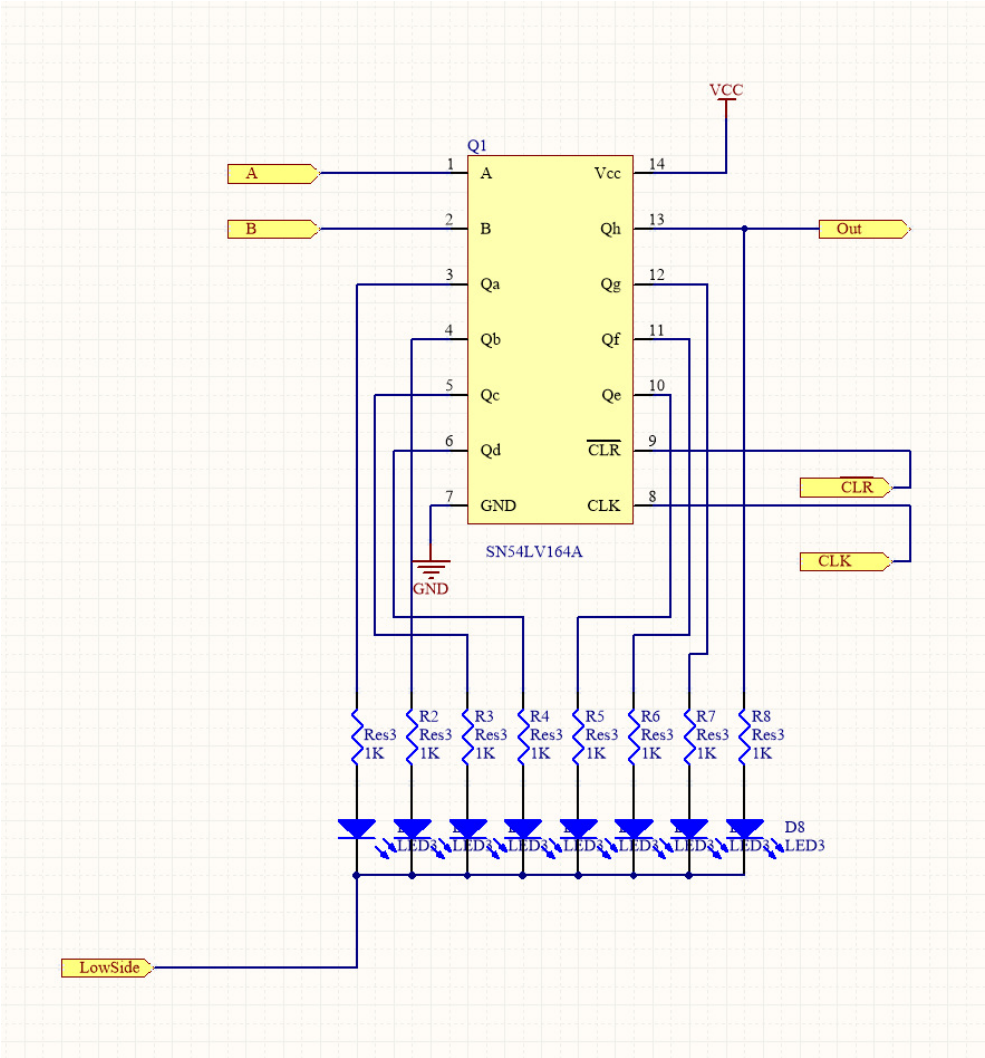


Figure C.7: The sub-schematic for eight lanes of the shift register driven LED debugging tool. This is a sub-schematic within the design presented in Figure C.6.