UNIVERSITY OF YORK

MASTERS THESIS

Augmented Reality Debugging System for Robot Swarms

Author:
Alistair JEWERS

Supervisor:
Dr. Alan MILLARD

A thesis submitted in fulfillment of the requirements for the degree of Master of Engineering

in the

Department of Electronic Engineering

April 3, 2017

Declaration of Authorship

I, Alistair JEWERS, declare that this thesis titled, "Augmented Reality Debugging System for Robot Swarms" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

University of york

Abstract

Faculty Name

Department of Electronic Engineering

Master of Engineering

Augmented Reality Debugging System for Robot Swarms

by Alistair JEWERS

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor...

Contents

List of Figures

List of Tables

For/Dedicated to/To my...

Chapter 1

Introduction

1.1 Background

Recent years have seen rapid development in robotics technology due the constantly increasing availability of computing power, reductions in the cost of hardware such as digital sensors and actuators, and developments in the application of artificial intelligence to robot control. This has lead to robots being used to perform increasingly complex tasks and solve ever more complex problems. Many new areas of robotics research have emerged as a result, as researchers strive to find new and better ways to apply this technology, entering into problem domains once thought to be impossible for robots. Whole new robotics paradigms have been created as the standard model of a single, complex, expensive robot has been questioned, opening the door for cooperative robots, multi-robot systems, and more specifically swarm robotics.

Studies into the self-organising behaviour of social insect colonies, and the development of mathematical models based on these behaviours, led to the development of a field of research referred to as Swarm Intelligence (SI). The aim of these models is to determine how large numbers of individual agents are able to solve problems collectively, with each agent using only local information, and without any centralised control. Swarm Robotics developed from a desire to apply these concepts in practice to real world problem solving. Dorigo et al. describe swarm robotics as '*the study of how to design groups of robots that operate without relying on any external infrastructure or on any form of centralized control ...* [where] *the collective behaviour of the robots results from local interactions between the robots and between the robots and the environment*[1]'. Swarm robotics has since emerged as a promising area of research for solving problems which would be infeasibly difficult or expensive for a conventional robotics approach.

1.2 Project Context

Developing and debugging robotics behaviours has always been a challenging task. Whilst traditional software is run in a purely digital environment with a tightly controlled set of inputs and outputs to and from the physical world, robots must interact constantly with the physical world in order to satisfy their intended purpose. Robots are therefore subject to a much wider array of inputs and outputs, and are subject to a huge number of changing variables within their environment at any given time. This makes detecting, reproducing and correcting specific faults significantly harder than in traditional software. One of the main difficulties comes from the layers of abstraction between the real world, the robot, and the human developer. There is a potential disconnect between the robot's interpretation of the world and the reality of the world itself. Inaccuracies in this interpretation can be caused by any number of issues, including sensor hardware problems as well as software bugs. This can cause erroneous behaviour that might be wrongly attributed to a bug in the robot's behavioural code or decision making, rather than its perception. This issue can be compounded by the fact that the human operator's knowledge of the robot's interpretation of the world might also be inaccurate or incomplete. Figure 1.1 shows these different layers of information abstraction when dealing with a robotic system. The arrow highlighted in red shows where many of the difficulties in debugging a robot's behaviour occur. Retrieving human readable information from a robot in a timely manner whilst it is running is often non-trivial, and what the robot sees and what the human operator thinks the robot sees may differ significantly.

This problem is made significantly more complex when working with multi-robot systems, and especially swarm robotics. Introducing multiple robots multiplies the number of potential variables and increases the amount of information required to describe the system, hence both the number of points where a bug may be occurring and the amount of information the operator needs in order to locate it are also increased. The decentralised nature of swarm robotics systems further exacerbates this problem through the lack of a single, central control point, where information for the whole system can be retrieved.

1.3 Project Concept

This project focuses on mitigating the problems discussed in the previous section, thus improving the timeliness with which bugs identified in a swarm robotics system can be located and fixed, by improving the operator's access to system information. This means collecting information from multiple sources and presenting it all in one place, in a human readable manner, in real time. The information sources to be used include the individual robots themselves, as well as a live camera feed of the robots' environment.

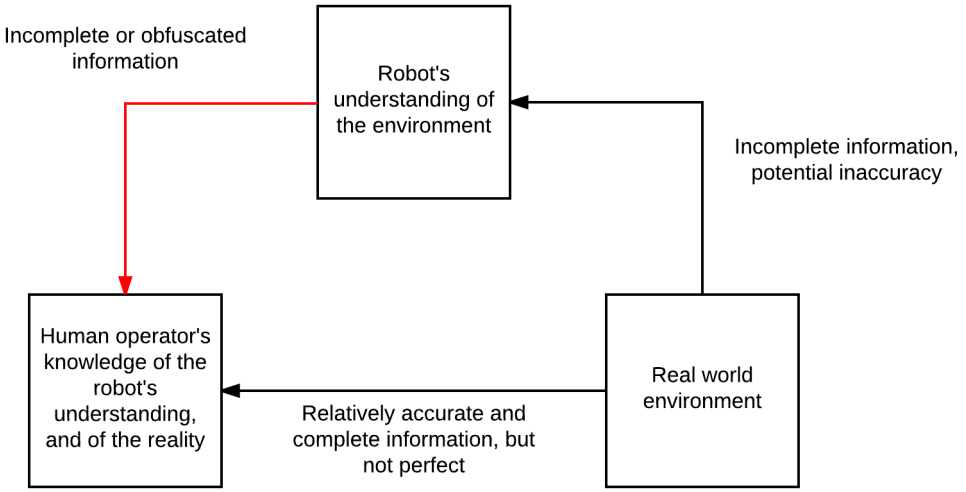

FIGURE 1.1: Layers of information abstraction in robotics debugging.

This project attempts to achieve this by creating a software application and associated wireless data transaction format to present a user with a single, coherent, and highly readable interface through which they can view relevant information about the swarm and it's constituent robots in real time. This includes the use of a video based tracking system to monitor robot positions, and provide the user with a view of the robots' environment. This can then be augmented with graphical representations of relevant elements of the data retrieved from the robots, such as sensor readings. The robots will communicate data to the computer running the application wirelessly. The wireless protocol to be used initially will be WiFi, with Bluetooth to be considered as a possible extension. The initial target robot platform is the widely used *e-puck* robot [!!EPUCK REF!!], equipped with a Linux extension board and WiFi adapter, hence the choice of WiFi as the first wireless protocol to support. The e-puck platform is discussed in greater detail in section . The diagram in Figure gives a logical representation of the proposed system architecture, in terms of its components, including the e-puck robots, tracking camera, and the application's host computer. This report describes the design, implementation and testing of this system, and includes details of the steps undertaken to evaluate its effectiveness. Some portions of this report appeared previously in a similar form in an *Initial Report*, and are included here for completeness, with minor alterations.

1.4 Aim and Objectives

Given the descriptions of the project context and concept in sections 1.2 and 1.3, the project aim can be formalised, and a set of objectives determined. This project aims to understand the needs of a swarm robotics researcher or system developer when

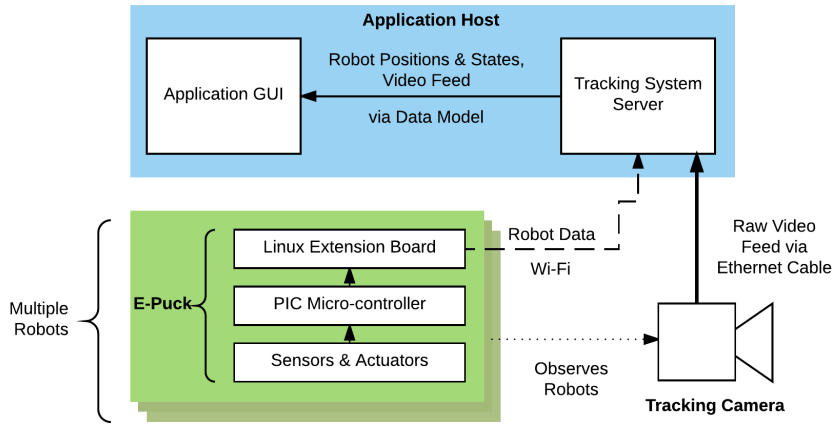

FIGURE 1.2: The proposed system architecture.

attempting to debug their system, and create a computer application which allows a user to monitor the state and behaviour of a robot swarm system in real time, thus improving the ease and efficiency of this debugging process. The objectives required to achieve this aim are as follows:

- Utilize existing fiducial marker based tracking technology to track the position of individual robots within a swarm over time.
- Develop code for presenting the user with a live video feed of a robot swarm, augmented with relevant and spatially situated information relating to the robots, using the data obtained from the tracking system.
- Develop code to allow multiple robots to communicate information regarding their internal state, sensor readings and decision making to the main application wirelessly via a network.
- Develop a data model that allows the application to store the information it receives from the robots, and update it as new information arrives.
- Develop code to ascertain higher level data related to the robots, such as recent movement history or state trasition history, and add this data to the model.
- Design and implement a user interface which presents the data model to the user in a human readable manner, and performs data fusion on the information provided by the robots and the tracking information.
- Develop the user interface in such a way as to allow the user to filter out information that is not currently relevant, and to contrast and compare information related to specific robots.
- Design and implement the system in a modular way so as to allow for relatively simple integration with other swarm robotic platforms and tracking systems in future extensions.

1.5 Functional Specification

When developing software of any kind it is common practice to define a software specification prior to starting development. This specification describes the functionality required in the software in order for it to satisfy its purpose. The specification presented here is separated into core and secondary requirements. Core requirements are considered essential to the satisfactory delivery of the software. Secondary requirements will be satisfied where possible, given the time constraints of the project.

Core Requirements:

1. Must be comprised of a PC application.
2. Must be capable of receiving data related to the state of multiple robots.
3. Must be capable of receiving positional data for the same set of robots.
4. Must be capable of receiving a live video feed of the robots in their environment.
5. Must collate received data and present it to the user in a combined graphical form.
6. Must present auxiliary, non-spatial data to the user in textual or other forms.
7. Must update in approximately real time.
8. Must at minimum support the e-puck robot platform.

Secondary Requirements:

1. Should use a modularised structure.
2. Should exchange data between the robot platform and the application using a platform-agnostic, extensible protocol.
3. Should provide a basis for interoperability with a number of robotics platforms.
4. Should allow the user to configure the displayed data.
5. Should employ a model-view-controller (MVC) software architecture.
6. Could provide the user with ways to configure and display custom data types.
7. Could allow the user to compare data on two or more specific individual robots.
8. Could calculate and display swarm-level meta-data and statistics.
9. Could generate log files of robot activity over a user defined period.

1.6 Report Structure

This report begins with a survey of the existing literature relevant to the project topic. Individual pieces of research and writing with relevance to some area of the project are highlighted. The project plan is then outlined, with details of the tasks to be undertaken and the time allocated for each, as well as the risk assessment made at the start of the project. This is followed by a short statement on ethics. The hardware to be used in the project is then examined in detail, with information about the target robot platform, the e-puck, and details of the camera and tracking system. The results of an initial survey of some potential users of the system is then presented, and the effect of these results on the implementation are summarised. The design process is then described in detail, including both the structural design of the software architecture and the design of the user interface. The next section gives details of the implementation of the software. This is followed by a description of the testing processes used to verify the software and the results, and details of how the system's effectiveness was evaluated. The penultimate section focuses on possible future work that could be carried out to improve the system. Finally the conclusion looks at the system as a whole, discusses its effectiveness, shortcomings, and its place in the wider field of swarm robotics.

Chapter 2

Literature Survey

2.1 Overview

Although a relatively young field, Swarm Robotics has already generated a substantial body of research and literature. This section presents an overview of that literature, and highlights specific pieces of research identified as relevant to this project, with the aim of providing the reader with the base of knowledge required to better understand the project. This research informed the project direction significantly, and formed the basis for many of the design and implementation decisions made later. The literature covered in this section can be separated into several broad topics, each informing a different element of the project work.

Firstly an understanding of the fundamental concepts of Swarm Robotics, and to a lesser extent Swarm Intelligence, was deemed key to producing an application that is useful in practice, and will help a reader to better understand the purpose and aims of the project. An overview of the core concepts as well as some key publications are presented in Section 2.2. A deep understanding of the technical details of specific swarm systems, such as specific behavioural algorithms or implementation details, is not a priority for understanding this project, as the application aims to be more broadly applicable to a wide range of swarm systems. Emphasis was instead placed on understanding the general classification of swarm robotic systems, relevant problem domains, and recurring concepts, so that the software might better serve researchers in the field.

This project focuses on a piece of software which forms an interface between a human operator and a robot swarm. A relevant area of current research is therefore Human-Swarm Interaction (HSI). This topic focuses specifically on the different roles humans take whilst interacting with robot swarms, and contains research into the best practices for facilitating this interaction given different aims, and different types of user (Developer, researcher, end user, etc.). The two key challenges of HSI are control - how best to allow a human operator to direct the behaviour of a decentralised swarm - and monitoring - how to retrieve data from a swarm and present it in a useful, human readable manner. This project focuses on the latter problem. An

overview of the relevant Human-Swarm Interaction literature is presented in Section 2.3.

Recent advances in virtual-reality (VR) and augmented-reality (AR) technologies have led to an increased interest in using these technologies in conjunction with robotics. AR specifically presents a powerful tool for human-robotic interaction (HRI), including HSI, as a digitally augmented space can be readily understood by both humans and robots. Research relating to the use of AR with robotic systems is summarized in Section 2.5.

A number of systems exist which utilize a range of the concepts previously discussed in the context of multi-robot systems, and this work is summarised in section 2.6. This includes other real time, graphical debugging systems which bear similarities to the aims of this project .

2.2 Swarm Intelligence and Swarm Robotics

Sahin [2] presents a summary of the key concepts of swarm robotics, and attempts to offer a coherent description of the topic. He notes that a key difference from other multi-robot systems is the lack of centralised control, and the idea that desired behaviour should emerge from simple local interactions between robots, and between the robots and their environment. He also notes some of the key motivators behind Swarm Robotics research, stating that a swarm robotics system would ideally have "*robustness*", "*flexibility*" and "*scalability*" [2]. Robustness refers to the swarm's ability to continue to function should one or more individual swarm members suffer a failure of some kind. Flexibility refers to the swarm's ability to adapt to changes in the environment without the need for re-programming. Scalability describes the idea that a swarm should be functional at a range of sizes, and that ideally the number of robots in the swarm could be increased or decreased depending on the demands of the task. Sahin [2] goes on to describe several classes of application where Swarm Robotics systems might be well suited. Tasks that cover a region could benefit from a swarm's ability to distribute physically in a space according to need. Dangerous tasks could benefit from the relative dispensability of individual robots in the swarm; should one be damaged or destroyed the swarm could continue to function, and it would be less costly that the loss of a single, complex, expensive robot. Tasks requiring scalability are good candidates, as discussed before, and tasks that require redundancy are also highlighted, as swarm systems should have the ability to degrade gracefully, rather than suffering a single catastrophic failure. Through this generalisation of the application areass, insight can be gained into the kinds of work swarm robotics researchers are likely to be doing, and this should inform the design of the application. This paper [2] provides a coherent, succinct overview of the field, and although it is now over a decade old the concepts covered remain relevant.

The book '*Swarm Intelligence: From Natural to Artificial Systems*' written by Bonabeau, Dorigo and Theraulaz [3] provides in its introductory chapter a good overview of the biological concepts and animal behaviours which inspire the field of swarm intelligence. The later chapters provide a detailed look at several of these behaviours, and how mathematical models and algorithms can be derived from them. Although more detailed than this project requires, an understanding of these behaviours and models can offer insight into what information the application might need to expose to the user to allow them to validate the correct operation of a system based on these concepts.

2.3 Human Swarm Interaction

In their paper '*Human Interaction with Robot Swarms: A Survey*' [4] Kolling at el. begin by noting the lack of research into methods for interfacing humans and robot swarms. They suggest that real-world applications for swarm robotics systems are now within reach, and that discovering effective methods for allowing humans to control and/or supervise swarms is a key barrier to realising these systems. The paper [4] provides a detailed analysis of human swarm interaction from a number of different perspectives. Of relevance to this project is the statement on page 15 that "*Proper supervision of a semiautonomous swarm requires the human operator to be able to observe the state and motion of the swarm, as well as predict its future state to within some reasonable accuracy*" [4]. Considering that swarm supervision and swarm debugging are highly comparable tasks - both involve observing the swarm whilst performing its task and determining the validity of the behaviour observed - this statement lends credence to the aims of this project. The proposed application would allow the state of the swarm, including the internal state of individual robots, to be observed simultaneously with the physical positions and motions of the robots within their environment. The paper [4] goes on to suggest that by observing the swarm over time the human operator will be able to provide '*appropriate control inputs*'. In the case of this application, rather than providing control input, the human operator will be seeking to identify faults, and provide appropriate corrections to the system, however the concept of state visualisation remains relevant.

Rule and Forlizzi [5] present a thorough examination of the complexities of human robot interaction (HRI) when dealing with multi-robot (and multi-user) systems. Much of the paper focusses on control methods, which are not directly applicable to this project, however Section 2.4 titled *Salience of Information* discusses the task of designing interfaces for displaying information about multi-robot systems to a human operator in a manner which is both information dense and rapidly understandable. The authors note that the use of colour has been shown to improve interface readability [6], and that the brain has been shown to process text faster than images [7], hence complex icons should be avoided. These ideas should be incorporated into the

design of the application user interface for this project. A range of different designs could be explored, including finding a balance between the amount of information displayed graphically, and the amount displayed textually, and deciding whether to use colour to differentiate between individual robots, or to differentiate between different types of data, or a combination of both.

2.4 Debugging Robotics

Collet and MacDonald [8] describe in detail the difficulties in debugging robotics systems. The authors identify that the difficulties in developing and debugging robotics applications when compared to traditional software arise from either the environment of the robot - which will often be "*uncontrolled*" and "*dynamic*" - or from the mobile nature of the robot. Because the environment a given robot operates in is a real world space, the level of control that can be exerted over it by the researcher or operator is inherently limited [8]. The environment may therefore change over time, exhibit imperfections, and include other time-varying elements. A robot is a physical actor and will likely experience dynamic change in its sensor readings and its relationship to the environment over time. This is especially true for mobile robots, whose position and orientation will change over time. The behaviour of the robot often largely depends on these highly variable factors, and therefore replicating a given behaviour exactly becomes almost impossible. The authors go on to state that difficulties in debugging often arise from "*the programmer's lack of understanding of the robot's world view* [8]". It can therefore be extrapolated that for a multi robot system such as a robot swarm this problem would be exacerbated. Each robot will have its own perception of the environment, which will differ based on differences in the robots positions and orientations as well as variations in the instrumentation of each robot. For a multi-robot system the programmer is required to have an understanding of not just one but multiple world views, adding yet more potential for error and inaccuracy, and further obscuring bugs or behavioural issues that the programmer is trying to diagnose. This paper [8] and further analysis of the problem in the swarm context suggest that developers working on these systems will need specific tools which mitigate these issues in order to develop effectively for swarm robotic platforms. This need forms the mandate for this project. Collet and MacDonald [8] also present an augmented reality based software tool for this purpose, which is discussed in Section 2.5.

2.5 AR and Robotics

Augmented reality presents a powerful tool for use with robotics, and specifically for debugging robotics, as it allows information gathered by a robot about an environment to be superimposed onto that environment in a way which can be inherently understood by humans. Milgram et al. [9] discuss the different communication formats used to interface between humans and robots, grouping them into "*continuous*" and "*discrete*" formats. For any communication involving a spatial or temporal component, the process of converting to and from a discrete format in order to transmit this information is an unnecessary burden. Both humans and robots use the continuous spatial dimensions, and humans have an inherent, instinctive understanding of physical things expressed in three dimensions. The authors therefore identify that [9] augmented reality provides an excellent means of supporting the communication of spatial information. Their paper focuses on the combination of stereoscopic displays and computer generated graphics to allow for more intuitive control of robotic systems. In the case of this project the concept is reversed; robots reporting spatial information for validation by a user should do so in a format which is inherently continuous such as AR, rather than one that is discrete such as text-based numerical output. This should in theory reduce the time required for a human to process the information. The authors note that [9] the ideal system utilises both discrete and continuous formats where appropriate to best communicate the required information, and is ergonomically designed to allow the user to make use of both easily and intuitively.

Like much of the literature surveyed, this paper [9] is focused primarily on robot control, rather than observation and monitoring. In spite of this much of the content of the paper remains applicable. Since the paper was written, just over twenty five years ago, major advancements have been made in virtually every area mentioned, including the quality and precision of robotic systems, their cognitive, perceptive and decision making abilities, augmented reality technologies and robotic autonomy. Because of this, some of the contents of the paper have fallen out of date. Specifically, the assumption that robots lack the level of autonomy required to survey their environment and then form and execute a series of steps to carry out a relatively high level task, such as "find and go to object Q"[9], is no longer necessarily true. A number of modern robots possess sufficient sensing capabilities, processing power and cognitive programming to perform such tasks based on high level commands. This does not however de-value the AR methods discussed within, and given the increased complexity and sensing capabilities of modern robots AR based methods of interaction actually show more potential than ever. The paper however makes no mention of the potential for an AR system to report data from a robot's sensors visually, which may be attributed to the technological limitations of the time rather than an oversight by the authors.

FIGURE 2.1: Sonar sensor data visualisation in Collet and MacDonald's system [8].

Collet and MacDonald [8] suggest that augmented reality tools can address and mitigate some of the robotics debugging issues discussed in section 2.4 by superimposing graphical representations of the robot's understanding of the environment on top of a live view of the environment itself [8]. Hence the programmer is able to see how the robot has interpreted the environment, and identify inconsistencies. The authors describe the image of the real world environment as the "*ground truth*" against which the robot's view can be compared and contrasted [8]. Figure 2.1 shows a visual example of this technique, where the data received from the robot's sonar sensors is converted to spatially situated 3D shapes and superimposed over the live image, and can therefore be verified visually by the user.

The application developed during this project closely follows this paradigm; allowing the user to identify bugs by comparing the robot's knowledge of its environment and its decision making factors (collectively referred to as its state) with a view of the environment, in real time. The application aims to apply this concept specifically to swarm robotics systems, and therefore must allow the user to compare the states of multiple robots with the environment simultaneously. From the perspective of each robot in the swarm, the other robots will form part of the environment, therefore the application must take this into account when displaying the information. Because of the large increase in information from a single-robot system to a multi-robot one, it becomes important that the application provide a way for the user to filter what information is displayed, allowing them to focus on the primary aspect under test. Filtering also allows the user to compare and contrast specific robots against one another by filtering out information related to other robots, or by displaying in more detail information related to the robots of interest.

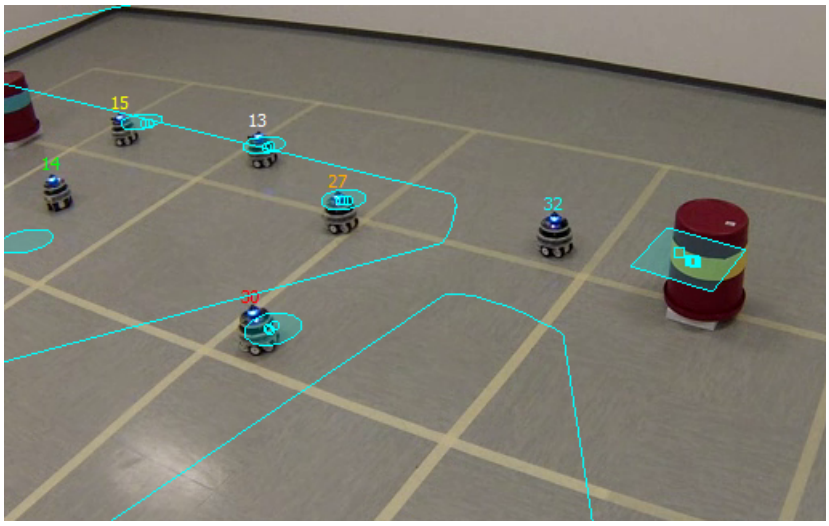

FIGURE 2.2: Example of spatially situated data overlayed on a live image in [11].

2.6 Similar Work

Ghiringhelli et al. [10] present a system for augmenting a video feed of an environment containing a number of robots with real time information obtained from each of the robots. This is similar in concept to the system described by Collet and MacDonald [8], but is designed specifically to target a multi-robot system. The authors identify the ability to overlay spatial information exposed by the robots on to the video feed in real time, in the form of situated graphical representations, as the most important debugging feature of the system. Figure 2.2 shows a spatially situated overlay of data exposed by robot thirty two, in the authors system [10]. A viewer is able to immediately verify the validity of the robot's world view from this image by comparing the blue overlay to the image beneath. Each robot features a coloured LED blinking a unique coded pattern to enable tracking, and the system uses homography techniques to map between the robots' frame of reference and the camera's. The project proposed in this report intends to use a simpler approach, with position and orientation tracking achieved through the use of the AruCo [11] marker-based tracking system, and a birds-eye view position for the camera to simplify mapping by effectively reducing the space to a 2D approximation.

Chapter 3

Project Plan

This project was completed between Monday 16th January and Thursday 18th May, 2017. A well defined break down of the tasks required to complete this project, and an organised plan for completing these tasks was instrumental in ensuring that this project was completed in the available time. However, as with almost all modern software development, accurately predicting the time required to implement every piece of code was a virtually impossible task, as the development process led to the discovery of unforeseen issues and a deeper understanding of the problem constraints. Hence wherever possible an '*agile*' methodology and approach was employed, including frequent re-assessment of the remaining work and feasibility of individual features. This is discussed in more detail in section 3.4.

3.1 Work Breakdown

At the start of the project the software development and software testing work was divided into the logical tasks shown in tables 3.1 and 3.2 respectively. The timings given for each development task are approximate, and based on prior experience with software work. Other tasks, not related specifically to the software development, are listed in table .

TABLE 3.1: Development tasks.

Task	**Objective**	**Approximate Time**
Read and Understand Existing Code	To understand existing code related to the tracking camera and networking on the e-pucks.	14 Days (Alongside other development)
Establish Development Environment and Toolchain	To enable organised development by establishing a tool set and workflow.	3 Days

Learn to Re-Program e-puck Robots	To understand the cross compilation process for the e-pucks.	2 Days
Outline Software Architecture	To design a coherent code structure in order for code to remain organised and modular.	2 Days
Design General User Interface	To create a high level design of the basic UI and implement a skeleton framework of this UI.	3 Days
Incorporate Tracking Camera Code	To incorporate existing low-level code for acquiring images from the tracking camera and performing tag detection.	2 Days
Implement Tracking Camera Controller	To implement code to create a layer of abstraction between the application code and the tracking code.	2 Days
Implement Wireless Data Receive	To implement code to allow the application to receive data wirelessly.	3 Days
Determine Robot Data Types	To establish an initial set of data types that will be supported by default, and a packet format for these.	2 Days
Design and Implement Data Model	To design the back end data model of the application and implement it in code.	6 Days
Implement Mapping Received Data to Model	To implement code to store received robot and tracking data in the application data model.	3 days
Implement Basic Visualiser	To implement code for displaying the video feed and augmenting it with basic geometric primitives.	5 Days
Design UI Data Representation	To establish a design for the representation of the different data types.	2 Days
Implement Graphical and Textual Data Visualisation	To implement code to convert the data in the data model into relevant visualisations.	10 Days
Implement Data Visualisation Filtering	To implement code to allow the user to filter out unnecessary visualisation elements.	5 Days

Implement Robot Data Comparison	To implement code to allow the user to compare the data of specific robots.	3 Days

TABLE 3.2: Testing tasks.

Task	**Objective**	**Approximate Time**
Continuous Integration Testing	To continually test newly implemented features with the system as a whole during the implementation process.	Throughout development
Manual Verification Testing	To verify the correct operation of the software through manual testing. Specific focus on the UI.	10 Days
Verification Fixes and Changes	To make the necessary changes to correct issues identified in the verification testing.	5 Days
Final Fixes and Changes	Some leeway time is available to make any final changes or fixes based on the results of the user evaluation sessions.	Remaining time

TABLE 3.3: Other tasks.

Task	**Objective**
Initial Report	To produce an initial report in the early stages of the project, outlining the preliminary research completed and the project plan at this stage.
Create Initial User Survey	To create a survey to be answered by potential users of the system such as robotics researchers to gauge interest levels for the proposed system and specific individual features.
Distribute Initial User Survey and Collate Results	Distribute the survey and collate and analyse the responses.

Create a System Evaluation and User Testing Plan	To devise a plan for evaluating the effectiveness of the system including a detailed description of the user testing procedure.
User Evaluation Sessions	To evaluate the system by allowing a number of users to use it in a structured evaluation session.

3.2 Timing

3.3 Risk Analysis and Mitigation

3.4 Application of Agile Methodologies

Chapter 4

Statement of Ethics

The ethics considerations affecting this project were minimal, as the work did not involve other humans, animals or potentially unethical practice, and the system does not have direct applicability to military or defence work, or other potentially unethical uses. The system has some potential for indirect use in military or defence work - for example in debugging a military swarm system - however this is an extremely unlikely scenario, and the potential is no greater than for any other robotics software tool. The evaluation process for the project did involve human participants and data collection, and the steps taken to ensure this participation was ethically sound are discussed in section 4.1.

4.1 Human Participant Consideration

4.1.1 Initial User Survey

The initial user survey was a short questionnaire created to gauge the interest of potential users in the system as a whole and individual features. Participants provided responses voluntarily, and the responses were anonymous by default. Participants were optionally given the chance to add their name and email address, so that they might be contacted regarding the later user testing if they were interested. This data was stored in a password protected fashion. This identifying data is not presented as part of this report, and the response data is presented in an aggregated fashion, ensuring anonymity.

4.1.2 User Testing and Evaluation Sessions

The final user testing and evaluation involved an observed testing session, where participants used the system in context whilst under observation, and a final questionnaire. Participants took part voluntarily, and consented to both the observation process and to the use of their questionnaire responses in an anonymous fashion. The data collected does not include any personal or private information, and is

purely related to each participants experience and opinion of the system. Once again the questionnaire data is presented in an aggregated fashion to preserve anonymity. Anecdotal information obtained through observation does not include any identifying information about the participant, and is also fully anonymous.

Chapter 5

E-Puck Robot Platform

5.1 Overview

The e-puck robotic platform, created by a team at the *Ecole Polytechnique Fédérale de Lausanne* [12], is a small, relatively inexpensive, multi-purpose robotic platform designed for for education and research pursuits regarding robotics and multi-robot systems. The platform is widely used in swarm robotics research, featuring in a number of publications. The e-puck was chosen as the first target platform for this system for a number of reasons. Firstly this was one of the platforms available in suitable numbers in the York Robotics Laboratory (YRL) at the University of York, where the practical work for this project was carried out. Secondly the platform's wide use in swarm robotics research helps to show the broad applicability of the system, and better demonstrates its value when compared to a less widely used or bespoke platform. Finally the platform's extensible design meant that it could be equipped with a Linux extension board, a configuration frequently used at the YRL. This made the use of WiFi for wireless data transfer feasible, and was a large part of the reason for the e-pucks choice. This section provides details of the e-puck robot's hardware, including processor, sensors and actuators, as well as details of the configuration used for this project, including the Linux extension board. Figure 5.1 shows the e-puck robot [TAKE NEW PICTURE].

5.2 Processor

The e-puck features a dsPIC 30F6014A microprocessor, designed and manufactured by Microchip Technology Inc. This is a general purpose 16-bit CPU, with relatively low performance by modern standards. The processor features 68 I/O pins, which are connected to the e-pucks various peripherals. Due to the use of the more powerful Linux extension board discussed in section 5.5, the primary use of the PIC processor in the e-puck configuration for this project is to interface with the e-puck's hardware. This involves receiving commands from the Linux extension board and sending appropriate control signals to the robot's actuators, as well as reading the

FIGURE 5.1: The e-puck robot.

robot's sensors and passing the retrieved sensor data back to the Linux extension board.

5.2.1 Firmware

Prior to the start of this project the YRL had already developed a library of low level code allowing the PIC to function in the hardware interface role as described above, controlled through the UART serial port. This firmware code was used as-is on the e-puck PIC controllers throughout the project.

5.3 Actuators

The e-puck robot features two wheels, independently actuated by two step motors. The wheels have a diameter of approximately 41mm. The motors can rotate the wheels at an approximate maximum speed of 1000 steps per second in either direction, where 1000 steps is one full revolution. The robot also features a ring of 8 red LEDs around the edge of the main circuit board.

5.4 Sensors

The e-puck robot features a number of different sensor sets, of which only some are used in this project. A set of 8 IR proximity sensors are arranged around the circumference of the robot, with four positioned on the forward hemisphere, two positioned at right angles to the forward direction (one on either side) and two more positioned on the backward hemisphere at roughly 45 degrees either side of the backward direction. Figure 5.2 shows this layout. The IR sensors can be used in two modes -

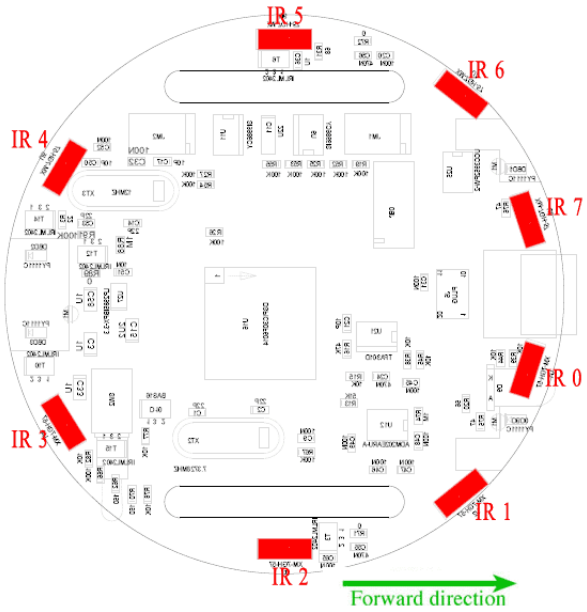

FIGURE 5.2: The layout of the IR sensors on the e-puck robot.

active and passive. In active mode the sensor emits an IR pulse and measures the IR strength of the reflection, whereas in passive mode the sensor simply samples the IR strength without emitting a pulse. The passive mode can therefore be used to get a 'background' IR reading, which can be compared to the active reading to improve accuracy. The IR sensor is of particular interest to this project as it is a frequently used tool when working with robots, especially robot swarms.

The robot also features three microphones, a 3 axis accelerometer and a camera. Due to the bandwidth required to use the microphones and camera they were considered a low priority for this project. [ACCELEROMETER?]

5.5 Linux Extension Board

For this project the e-pucks were fitted with an extension board featuring a 32-bit ARM9 processor running a modified Linux operating system [13], developed by Wenguo Liu, and Alan F.T. Winfield. In this configuration the ARM processor, an Atmel AT91, takes charge of the high level robot control logic, as well as any intensive data processing operations. The dsPIC processor is then used to control the low level actuator and sensor control, running in parallel with the ARM processor and communicating via UART. The extension board provides a USB port, and for this project a WiFi adapter was connected to each robot. The controller code running on each robot could then make use of the standard IP network layer protocol, and the standard transport protocols TCP and UDP.

Chapter 6

Video Tracking System

6.1 Overview

In order to implement the augmented reality visualisation element of the system, and satisfy the related objectives, a live video feed of the swarm was needed. A method for tracking the positions of each individual robot in the swarm based on images from this feed was also required. Prior to the start of the project the YRL already had infrastructure in place for this kind of task, in the form of a machine vision camera placed above an 'arena', and software for processing the output of this camera using the 'ARuCo[11]' fiducial marker based tracking system. Figure 6.1 shows the arrangement of the machine vision camera used for robot tracking, and the robot arena. It was determined that incorporating this existing infrastructure into the system was the quickest way to get this required aspect of the system working, allowing work to focus on the novel aspects sooner.

6.2 Camera

[WHAT CAMERA IS USED? DETAILS.]

6.3 ARuCo Tracking System

Developed by a team from the Computing and Numerical Analysis Department at Cordoba University in Spain, the ARuCo tag generation and detection system [11] is a powerful fiducial marker creation and tracking tool. It comprises an algorithm for producing a 'dictionary' of square, black and white, coded markers which can be printed and attached to objects and surfaces, and a method for automatically detecting these markers in a given image. The stated applications include augmented reality and robot localisation. One of the main benefits of this system over other fiducial marker systems is the execution speed. By first using edge-detection methods to find the outlines of markers in the image, the system can eliminate a large

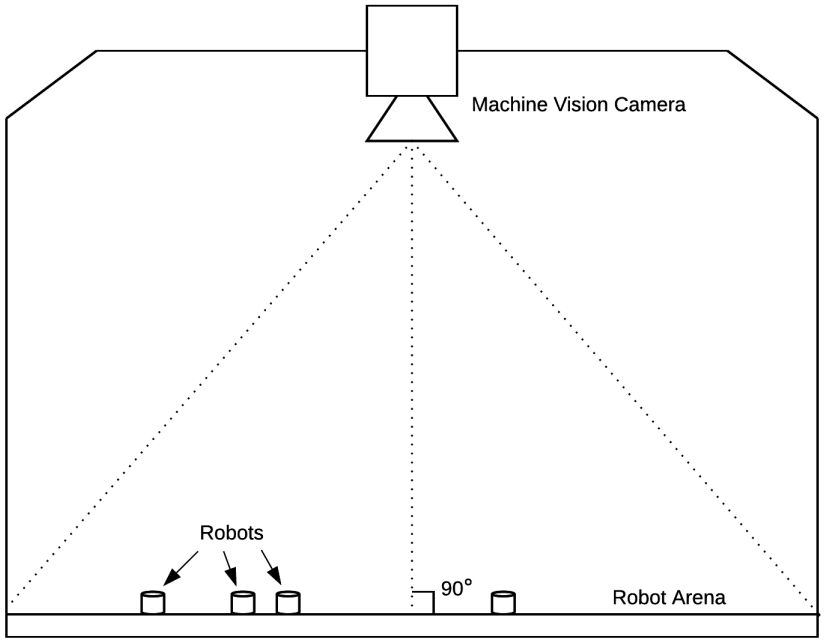

FIGURE 6.1: Arrangement of the tracking camera over the robot arena.

portion of the image before applying the more complex processing to identify and differentiate individual tags [11]. This makes it possible for the ARuCo system to be run in real time, even with relatively modest computational power.

Appendix A

Frequently Asked Questions

A.1 How do I change the colors of links?

The color of links can be changed to your liking using:

`\hypersetup{urlcolor=red}`, or

`\hypersetup{citecolor=green}`, or

`\hypersetup{allcolor=blue}`.

If you want to completely hide the links, you can use:

`\hypersetup{allcolors=.}`, or even better:

`\hypersetup{hidelinks}`.

If you want to have obvious links in the PDF but not the printed text, use:

`\hypersetup{colorlinks=false}`.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Bibliography

[1] M. Dorigo, M. Birattari, and M. Brambilla. "Swarm robotics". In: *Scholarpedia* 9.1 (2014), p. 1463.

[2] Erol Sahin. "Swarm Robotics: From Sources of Inspiration to Domains of Application". In: *Swarm Robotics WS 2004*. Ed. by E. Sahin amd W. M. Spears. Berlin: Springer, 2005, pp. 10–20.

[3] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: From natural to artificial systems*. Oxford University Press, 1999. ISBN: 0-19-513159-2.

[4] A. Kolling et al. "Human Interaction With Robot Swarms: A Survey". In: *IEEE Transactions on Human-Machine Systems* 46.1 (2016), pp. 9–26.

[5] A. Rule and J. Forlizzi. "Designing interfaces for multi-user, multi-robot systems". In: *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction* (2012), pp. 97–104.

[6] R. E. Christ. "Research for evaluating visual display codes: An emphasis on colour coding". In: *Information design: The design and evaluation of signs and printed materials* (1984), pp. 209–228.

[7] C. Carney and J. L. Campbell. *In vehicle display icons and other information elements: Literature review*. Tech. rep. U.S. Department of Transportation, Federal Highway Administration, 1998, pp. 209–228.

[8] T. H. J. Collet and A. MacDonald. "An augmented reality Debugging system for mobile robot software engineers". In: *Proceedings 2006 IEEE International Conference on Robotics and Automation* (2006), pp. 3954–3959.

[9] P. Milgram, D. Zhai S. Drascic, and J. Grodski. "Applications of augmented reality for human-robot communication". In: *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems '93, IROS '93.* 3 (1993), pp. 1467–1472.

[10] F. Ghirighelli et al. "Interactive Augmented Reality for understanding and analyzing multi-robot systems". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2014), pp. 1195–1201.

[11] S. Garrido-Jurado et al. "Automatic generation and detection of highly reliable fiducial markers under occlusion". In: *Pattern Recognition* 47.6 (2014), pp. 2280–2292.

[12] F. Mondada et al. "The e-puck, a Robot Designed for Education in Engineering". In: *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions* 1.1 (2009), pp. 59–65.

[13] W. Liu and A. F. T Winfield. "Open-hardware e-puck Linux extension board for experimental swarm robotics research". In: *Microprocessors and Microsystems* 35.1 (2011), pp. 60–67.