Honours Project Report

Rescue Robot Project
(Human Robot Interaction)

By Zwivhuya Tshitovha

Supervised by
Audrey Mbogho

Co-Supervised by
Anet Potgieter and Stephen Marais

<table>
<thead>
<tr>
<th>Category</th>
<th>Min</th>
<th>Max</th>
<th>Chosen</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>Total marks</td>
<td>80</td>
<td>80</td>
<td></td>
</tr>
</tbody>
</table>

Department of Computer Science
University of Cape Town
2011
# Table of Contents

Table of Contents ........................................................................................................... i
Table of Figures ............................................................................................................... iii
Abstract .......................................................................................................................... v

1 Introduction .................................................................................................................. 1
  1.1 Structure of report ................................................................................................. 2

2 Background .................................................................................................................. 3
  2.1 Project Aim for Human Robot Interaction ......................................................... 3
  2.2 User Centered Design ......................................................................................... 3
  2.3 Participatory design .............................................................................................. 3
  2.4 Design principles ................................................................................................. 3
    2.4.1 Affordance .................................................................................................. 4
    2.4.2 Constraints ................................................................................................. 4
    2.4.3 Visibility .................................................................................................... 4
    2.4.4 Mappings .................................................................................................. 4
    2.4.5 Providing Feedback .................................................................................. 4
    2.4.6 Managing Complexity ............................................................................. 4
    2.4.7 Dealing with errors .................................................................................... 4
    2.4.8 Recognition vs. Recall .............................................................................. 4
    2.4.9 Role integrity ............................................................................................. 5
    2.4.10 Principle of least astonishment ............................................................... 5
    2.4.11 Modes ....................................................................................................... 5
    2.4.12 Fitts’ Law ................................................................................................. 5
    2.4.13 Visual Weight ............................................................................................ 5
  2.5 Human-robot interaction ...................................................................................... 6
  2.6 Related Work ........................................................................................................ 6
  2.7 Conclusion ............................................................................................................ 7

3 Requirements and design .......................................................................................... 8
  3.1 System overview .................................................................................................. 8
  3.2 Software .............................................................................................................. 8
    3.2.1 Robot vision ............................................................................................... 8
    3.2.2 Robot control ............................................................................................ 8
  3.3 System Requirements and Design ..................................................................... 9
    3.3.1 Iteration 1 .................................................................................................. 9
    3.3.2 Second Iteration ......................................................................................... 13
    3.3.3 Final Iteration ........................................................................................... 19
  3.4 System constraints .............................................................................................. 27
4 System development and implementation ................................................................. 28
  4.1 Introduction ........................................................................................................ 28
  4.2 Bosch SDK features used .................................................................................. 28
      4.2.1 Video ........................................................................................................... 28
      4.2.2 Audio input and output ............................................................................. 28
  4.3 Robot control server ........................................................................................... 28
  4.4 Robot vision ......................................................................................................... 28
  4.5 Adherence to Design principles on the GUI ...................................................... 29
      4.5.1 GUI controls ............................................................................................... 29
      4.5.2 Grouped controls ....................................................................................... 30
      4.5.3 Input Constraints on controls .................................................................... 30
      4.5.4 GUI controls Mappings to actual representation ....................................... 30
      4.5.5 GUI user Feedback .................................................................................... 31
      4.5.6 Dealing with errors on the GUI ................................................................. 32
  4.6 Features .............................................................................................................. 32
      4.6.1 Camera view Window .................................................................................. 32
      4.6.2 Navigation tab ............................................................................................ 34
      4.6.3 Robot info log tab ....................................................................................... 36
      4.6.4 Environmental info log tab ........................................................................ 37
      4.6.5 Manipulator tab ........................................................................................... 38
      4.6.6 Robot setup tab ........................................................................................... 44
      4.6.7 Connection settings tab .............................................................................. 45
  5 Evaluation and Results ........................................................................................... 46
  5.1 Introduction ......................................................................................................... 46
  5.2 System Evaluation ............................................................................................... 46
      5.2.1 Bad design characteristics ......................................................................... 46
      5.2.2 Accuracy ..................................................................................................... 46
      5.2.3 Validation Testing ....................................................................................... 47
      5.2.4 Stability ...................................................................................................... 47
      5.2.5 Stress testing ............................................................................................... 47
      5.2.6 Restore testing ............................................................................................ 47
  5.3 User Evaluation ................................................................................................... 47
      5.3.1 Methodology ............................................................................................... 47
      5.3.2 Testing .......................................................................................................... 48
      5.3.3 Questionnaire Results ................................................................................ 50
      5.3.4 Conclusion .................................................................................................. 52
### Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Some disastrous areas that have been hit by earthquakes [15].</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>System with the project break down</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>iRobot PackBot EOD control unit [7]</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>Toin Pelican interface [7].</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>IUT Microbot user interface [7] and INEL’s interface [2].</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>Paper Prototype- Camera view window.</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>Paper information control Prototype- Navigation Tab on window.</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>Paper Prototype- Warning tab on information control window.</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>Paper Prototype- Status tab on the information control window.</td>
<td>13</td>
</tr>
<tr>
<td>10</td>
<td>Paper Prototype- Arm tab on the information control window.</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>Microsoft word- camera view</td>
<td>14</td>
</tr>
<tr>
<td>12</td>
<td>Microsoft word -Navigation Tab</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>Microsoft word –Warning Tab</td>
<td>14</td>
</tr>
<tr>
<td>14</td>
<td>Microsoft word –Status Tab</td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>Microsoft word –Arm Tab</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>High level prototype – Camera view Window.</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>High level prototype – Navigation tab.</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>High level prototype – Robot info tab.</td>
<td>17</td>
</tr>
<tr>
<td>19</td>
<td>High level prototype – Warnings tab.</td>
<td>18</td>
</tr>
<tr>
<td>20</td>
<td>High level prototype – Manipulator tab.</td>
<td>18</td>
</tr>
<tr>
<td>21</td>
<td>High level prototype – Status tab.</td>
<td>19</td>
</tr>
<tr>
<td>22</td>
<td>High level prototype – Cam Connection settings tab.</td>
<td>19</td>
</tr>
<tr>
<td>23</td>
<td>use case diagram for CameraView Window</td>
<td>23</td>
</tr>
<tr>
<td>24</td>
<td>Use case Diagram for Information Control window</td>
<td>24</td>
</tr>
<tr>
<td>25</td>
<td>Class Diagram for CameraView Window</td>
<td>25</td>
</tr>
<tr>
<td>26</td>
<td>Class diagram for Information control Window</td>
<td>26</td>
</tr>
<tr>
<td>27</td>
<td>Raised Buttons and Rollover Buttons</td>
<td>29</td>
</tr>
<tr>
<td>28</td>
<td>on/off button</td>
<td>29</td>
</tr>
<tr>
<td>29</td>
<td>Carbon dioxide concentration, microphone level and speaker level bars</td>
<td>30</td>
</tr>
<tr>
<td>30</td>
<td>Input constraints</td>
<td>30</td>
</tr>
<tr>
<td>31</td>
<td>Mapping between the actual robot and the 2d model</td>
<td>31</td>
</tr>
<tr>
<td>32</td>
<td>Roll, Pitch and Yaw orientation representation</td>
<td>31</td>
</tr>
</tbody>
</table>
Figure 33 Before and After the Home Button is clicked. ................................................................. 31
Figure 34 Feedback from saving connection settings ................................................................. 31
Figure 36 Color code for percentage values for signal and battery life .................................. 32
Figure 35 Camera View Window ................................................................................................. 32
Figure 37 Navigation tab features ............................................................................................. 34
Figure 38 Cartesian plane of the gauge and XY—Cartesian plane of the gauge showing the position of 0km/h and 20km/h respectively ................................................................. 35
Figure 39 Speed Gauge needle on the first quadrant of the XY—Cartesian ...................................... 35
Figure 40 Robot info log tab on the information control window ................................................. 36
Figure 41 Environmental info log tab .......................................................................................... 37
Figure 42 Manipulator Tab features ......................................................................................... 38
Figure 43 Driving and home position respectively ...................................................................... 38
Figure 45 3D shaped Robot image in a 2D xy—plane ................................................................ 39
Figure 46 The steps to be taken while drawing a 3D cube on a 2D surface ................................. 40
Figure 47 Reference point of measuring angle of robot arm on the robot ................................. 40
Figure 48 Reference Point for measuring finger1 and finger2 angle ........................................ 42
Figure 49 Roll, pitch and yaw rest position respectively and the reference point for measuring the angles  ......................................................................................................................... 42
Figure 50 The reference point for measuring the angle that the wrist has rotated ............................ 43
Figure 51 Robot Setup Tab ......................................................................................................... 44
Figure 52 Connection settings Tab ............................................................................................ 45
Figure 53 Percentage of new users who were able to accomplish a certain task .......................... 48
Figure 54 Comparison of usability and system design quality ratings between the 3 types of users ................................................................. 51
Figure 55 Total usability and system design rating for all users .................................................. 52
Abstract

This paper discusses the development of a User interface of a rescue robot from the design stages to the release stages. This interface will allow operators to remotely communicate with a rescue robot being developed in the Duncan McMillan Robotics and Agents Lab in University of Cape Town.

Each and every day humans live in fear of natural disasters and accidents happening in their lives. In those situations victims may be trapped in places that are dangerous and inaccessible by rescue teams because rescue team members also have to conscious of their own safety. In situations like those, a rescue robot comes to play and goes in those confined environments to save the victims.

Robots that are currently being built are not fully independent and need to be operated by a user. To operate the robot a user has to be presented with a usable, user friendly, user-centered and efficient interface in order to effectively communicate/ control the rescue robot and save lives.

Because this involves the communication between humans and robots, the development of this system is primarily a Human Robot Interaction (HRI) problem. To solve this problem an iterative and incremental approach was adopted. This entailed involving users from the first stages of design until the system is complete. This involved three iterations of the system design. Each design was improved upon feedback obtained from the users in the iterations. User testing that was done proved that a user centered and usable system can help rescue teams save lives efficiently and this justifies further research.
1 Introduction

Urban search-and-rescue (USAR) involves the location, rescue (extrication), and initial medical stabilization of victims trapped in confined spaces. Structural collapse is most often the cause of victims being trapped, but victims may also be trapped in transportation accidents, mines and collapsed trenches [20]. Some of the locations where the victims are located are dangerous and inaccessible for the rescue team. This is a huge challenge to the rescue team members as they have to help everyone in danger but also be conscious of their own safety. A more feasible solution to this problem is allowing machines such as robots to be part of the rescue team. This idea gave rise to RoboCup-Rescue project.

The trigger for the RoboCup-Rescue project was the Great Hanshi-Awaji earthquake which hit Kobe City on the 17th of January 1995 causing more than 6500 casualties, destroying more than 80,000 wooden houses and directly or indirectly affecting more than 1 million people. The damage of all infrastructures was evaluated at more than 1 billion US dollars [15].

The intention of the RoboCup Rescue project is to promote research and development in this socially significant domain at various levels involving multi-agent team work coordination, physical robotic agents for search and rescue, information infrastructures, personal digital assistants, a standard simulator and decision support systems, evaluation benchmarks for rescue strategies and robotic systems that are all integrated into a comprehensive system in future [15].

Robots can assist human teams in USAR tasks by travelling into dangerous, small, wrecked and unfamiliar areas to search for victims. Robots need to climb on and maneuver around rubble piles that are deemed too dangerous or hazardous for human or animal searchers to investigate. In difficult USAR environments, it is not yet possible to create a fully independent robot to completely take the place of a human rescue worker. In fact, most USAR robots that are sent into disaster zones are teleoperated. For this application, operators must have a good awareness of their surroundings, yet it is difficult to obtain situation awareness [15]. This means the operator has to be able to communicate with the robot efficiently. An efficient communication between the robot and the operator is made possible by a well-defined user interface. Figure 1 shows examples of disastrous areas that have been hit by earthquakes. Those areas in Figure 1 make it impossible for rescue teams to access and assist trapped victims.

![Figure 1 Some disastrous areas that have been hit by earthquakes [15].](image-url)
1.1 Structure of report
This report is broken into a number of sections.

1. Background Chapter – This section explains the aim of the project, what user-centred design is, what participatory is and the design principles to be considered when implementing the system. This chapter also discusses the field of Human Robot Interaction and related work that has been done in this field.

2. Requirements and Design – This section discusses the process of gathering what features are required on the system from the users and also explains what design techniques were used in order to create the system. It also has a detailed account of how these techniques were applied and how the system was designed.

3. System Development and Implementation – In this section it is explained in detail how the system was eventually implemented including the features of the system and how they work.

4. Evaluation and Results – This section discusses the testing and evaluation techniques used on the resulting system. It also explains how these techniques were applied and what conclusions were drawn.

5. Conclusions and Future Research – This section will explain what was learnt through this project including the conclusions drawn. It will also explain where this research can be taken from here.
2 Background

2.1 Project Aim for Human Robot Interaction
The aim of the project is to develop a user-friendly interface that is intuitive through iterative design and does not subject the user to sensory overload. The Human Robot Interaction has to take a minimalistic approach and only display the selected video streams and what is core to operating the robot. The user interface has to only bring in other data if requested or if required i.e. motor overload. Intelligence in the system is needed that is to think for the user to simplify the operation of the robot.

In order to build a usable and efficient interface User Centered Design techniques and design principles have to be used effectively. The system will be developed using an iterative and incremental approach. The initial designs will be made by the MSc students who are creating the hardware. After the MSc students have designed the system this will be tested on different users were feedback will be collected. The feedback that is collected from all the user testing will serve as input requirements for the next iteration.

The system being built is targeted to users with an engineering background and who have the skills of driving a rescue robot around. This is because the names of the controls that will be used will assume the users understand the meaning.

2.2 User Centered Design
User-centered Design (UCD) is a design approach which is based on research of user characteristics and needs through their involvement [3]. This also includes having the user participate in the design process not just understanding their needs. The user-centered approach is based on early focus on users and tasks, empirical measurement and iterative design.

Early focus on users and tasks involves directly studying cognitive, behavioural, anthropomorphic and attitudinal characteristics of users. Empirical measurement involves measuring reactions of users to scenarios, manuals, simulations and prototypes and their performance. User actions are observed, recorded and analyzed. Iterative design requires that problems be fixed during user testing and more tests should be carried out [10].

Some methods that support UCD are usability testing, usability engineering, heuristic evaluation, discount evaluation and participatory design. Usability testing aims to achieve the following goals: Improvement in the product’s usability, involving real users in the testing, giving the users real tasks to accomplish, enabling testers to observe and record the actions of the participants and enabling testers analyze the data obtained and make changes accordingly [20]. In the case of a rescue robot the users are well-defined thus it is much easier to carry out the user centered design techniques.

2.3 Participatory design
A Participatory Design (PD) workshop is one in which developers, business representatives and users work together to design a solution [10]. PD workshops give users a voice in the design process, thus increasing the probability of a usable design. It enables technical and non-technical participants to participate equally and provide an opportunity for developers to meet, work with and understand their users. It also provides a forum for identifying issues, provide an opportunity to get or enhance user buy-in.

2.4 Design principles
Software design principles are a set of guidelines to be followed to avoid implementing a bad design. The three bad characteristics of bad design to be avoided are rigidity, fragility and immobility [20]. If a software design is rigid it is very difficult to change because many parts of the software are connected together while a fragile system breaks when changes are made to some part of the system. An immobile system makes it difficult to use the system in another application because it cannot be separated from the
application it’s being used on. The following design principles have to be considered when designing a usable system.

2.4.1 Affordance
Affordance is the design aspect of an object which suggest how the object should be used; that is a visual clue to its function and use. Affordance is a very important property of a user interface because this decreases the number of human errors that can happen due to wrong perception of an object.

2.4.2 Constraints
Constraints restrict the allowed behaviour or interaction with an object by the user. There are different types of constraints such as physical, semantic, and cultural constraints. Physical constraints are the physical properties of an object which constrain the possible operations. Logical constraints are constraints that are attached to elements in the design prior to mapping or fitting. This helps in adapting the design’s performance to expected worst-case conditions. Cultural constraints are constraints which have evolved through artificial conventions that govern acceptable social behaviour. These cultural conventions have to be learned, but once learned apply to a wide variety of circumstances [10].

2.4.3 Visibility
Making things visible makes the operation of a device understandable, can act as a reminder of what can and cannot be done and makes the state of the system clear. Having Good visibility leads to objects that are easier to understand, use, and remember and quick to learn. The Principle of Visibility is that it should be obvious what a control is used for [10].

2.4.4 Mappings
A mapping is the relationship between two things. A mapping is the relationship between the controls in an interface and what they are intended to be used for. If the mapping is easy; then users will also find it easy to learn and remember, and therefore easier to use the control. A device is easy to use when there is visibility to the set of possible actions and where the controls and displays exploit natural mappings [10].

2.4.5 Providing Feedback
Feedback is the act of sending information back to the user about what has actually happened after the use has made some actions or about the state of the system. This can either be visually or literally. The Principle of Feedback is that it should be obvious when a control has been used [10].

2.4.6 Managing Complexity
Today’s devices and computer systems are commonly developed with many, many, features. However, the increase in controls and features makes it more difficult to make all the controls visible. This then makes it harder for the user to understand the device, learn how to use it and memorize functions.

2.4.7 Dealing with errors
If an error is possible, someone will make it. Therefore, known errors should be dealt with in advance. This is because complex devices and software will always require some instruction and there are users that will use that object without reading the manual first and they are prone to making errors. Thus possibility of errors should be minimized and the errors must be made as “cost-free” as possible. Errors must also be made easy to detect by the users and it must be made easy to reverse the effects of an error [10].

2.4.8 Recognition vs. Recall
Recognition and recall are two ways that the brain is able to retrieve information that has been stored in the memory. Recognition needs a cue before firing, while recall does not require any specific cues to start the impulses. Recognition is how the user responds to a sensory cue by the user first looking at something and then the mind looking to see if what he is seeing in front matches anything that has been stored. If
you notice a match is found this means that the information has been recognized. Recall is a way that information is obtained from memory without having a cue to prompt the response. The information has to first be drafted without any assistance. Recall is harder than recognition because there are no "hints" in place to aid in finding the answer [10].

2.4.9 Role integrity
User interfaces should not mislead the user about the capabilities the computer or machine can actually provide by specifying tasks on the user interface that the computer can’t actually provide this also includes hidden limits [10].

2.4.10 Principle of least astonishment
The Principle of Least Astonishment states that the result of performing some operation should be obvious, consistent, and predictable, based upon the name of the operation and other clues [10].

2.4.11 Modes
Modes are a functionality of an interface that allows the same interface to behave with different behaviours. If users are not aware of the mode change the interface may appear to be nondeterministic and unpredictable which then leads to an unusable interface in the eyes of the user [10].

2.4.12 Fitts’ Law
Paul Fitts observed that the action of pointing to or tapping a target object could be measured and predicted mathematically. In Fitts’ Law description of pointing, the parameters of interest are:

- Time to move to the target.
- The movement distance from the starting position to the centre of target.
- The width of the target.

Fitts discovered that it takes longer to hit targets which are further away and smaller. With this analogy, he derived the following equation, which is now known as Fitts’ Law:

\[ MT = a + b \log_2(2A/W) \]  

Where \( MT \) represents the movement time to hit the target, \( a \) and \( b \) are empirically determined constants. \( A \) represents the amplitude, which is the distance of the centre of the target from the starting location and \( W \) is the target width [10].

2.4.13 Visual Weight
Visual weight refers to the idea that distinct elements in a design have varying heaviness relative to each other. The main visual weight factors to be discussed are Color, Contrast and size. In order that symmetry and balance in designs is accomplished objects must appear equal in visual weight. If attention to certain objects needs to be emphasized to a certain object the balance must purposely be removed from the design so that the focus moves to the desired object of interest [19].

2.4.13.1 Color
Properties of color such as saturation, brightness/darkness, and hue affect an object’s visual weight relative to others on an interface. With reference to those properties more saturated colours have more attention than unsaturated colours and darker colours also take more of a stand than their lighter counterparts on light-coloured backgrounds [19].

2.4.13.2 Contrast
Controls that have more contrast between them and their background will be more noticeable than those with little contrast to their background colour therefore the use of contrast can maintain visual hierarchy and readability[19].
2.4.13.3 Size
In the physical world, an object that’s bigger than another of the same type will naturally be heavier and will take up more physical space therefore it is natural that bigger objects have higher visual weights than smaller ones[19].

2.5 Human-robot interaction
Human-robot interaction (HRI) is the interdisciplinary study of interaction dynamics between humans and robots [4]. Human computer interaction (HCI) as a field has made great strides towards understanding and improving interaction between users and computer-based technologies [4]. HCI plays a vital role in HRI together with human factors and are both part of research resource in HRI. Robots are a special and unique case of HCI because people perceive them as different from any other computer technology, because of their mobility and their ability to make unique decisions. People’s mental models of autonomous robots are often more anthropomorphic than are their models of other systems [8].

Being mobile brings a challenge to the robot as it must interact with different people, objects and environments. Similarly this brings a challenge to the robot operator as they control the robot remotely in those situations because a complex feedback system is required for effective interaction with the robot. There exists direct communication between a robot and a human; the human is either an operator or a victim who is familiar or non-familiar with the rescue robot [7].

2.6 Related Work
Adams [1] defined situation areas that should be taken into consideration while designing a robot interface. He said that human decision-making, situation awareness, vigilance, workload levels, and human error are the areas to be considered in the design. Vigilance represents sustained attention. A vigilance task refers to situations where an operator is required to detect signals over a long period of time, and the signals are intermittent, unpredictable, and infrequent [1]. The term workload may refer to mental or cognitive workload as well as physical and temporal workload. High mental workload may result in the operator making incorrect decisions that can lead to disastrous situations [1]. Low mental workload may result in the operator not properly monitoring the system and therefore losing situational awareness.

Situational awareness refers to the operator’s ability to properly understand the robotic team’s activities in their current environment [1]. In robotic systems, the operator’s situational awareness is limited by the sensor information provided by the robots. The operator should be presented with enough information to build a sufficiently complete mental model of the robot’s external state and internal state [7]. In general, HRI design ought to provide proper information in a specific situation and operator task distribution [1].

According to Adams [1] human error is an action that fails to meet some implicit or explicit standard of the actor or of an observer. This is an inappropriate or undesirable human decision or behaviour that reduces, or has the potential of reducing effectiveness, safety, or system performance. Baker et al [2] added that both efficiency and robot modality should also be considered for an effective design of interfaces for human-robot interaction in an USAR application. Efficiency means that it must take an operator minimum time and effort to accomplish desired job or task. There must be modest movement of hands, eyes and, equally importantly, there must be focus of attention [7]. Robot modality is providing operator assistance in determining the most appropriate level of robotic autonomy at any given time [2].

Kadous et al [7] also included efficiency, familiarity and responsiveness which they considered important and relevant design principles for a rescue robot interface. Familiarity means that concepts that the operator is familiar with should be adopted and unfamiliar concepts minimised or avoided. For more intuitive presentation to the operator information should be fused [7]. Responsiveness is ensuring that the operator is always having feedback as to the success or failure of actions [7].
Design principles are not always universal to all kinds of applications. Fong et al [5] developed a personal user interface to enable extravehicular activity (EVA) crew members and mobile robots to collaborate and jointly perform tasks in the field. During interface development the design principles and heuristics they used were single-click interaction, design for a small screen, effective use of colour, consistency, simplicity and task-specific modes.

A good robot interface is one which has as few numbers as possible and requires that the operator has as little movement as possible required in the hands, eyes and, equally importantly, focus of attention while using the interface. Video feeds are one of the most important areas that users tend to focus on while driving a robot.

Currently some of the teams that participate in the robot USAR competition in 2009 and 2010 designed more usable and effective user interfaces and are adopting some user interfaces that have been discussed in this paper. Those are teams like team AriAnA [9], team League [18], team Survivor [6] and team CASualty[Milstein]. Some teams were still suffering from the issues raised such as team iRAP_Pro [21], team Jacobs [17], team SHINOBi [12] and team SEU-RedSun[22].

**2.7 Conclusion**

For interfaces to be considered usable and effective both UCD techniques and design principles have to be incorporated in the design process. Some principles to be considered during robot interface design are human decision-making, efficiency, familiarity, responsiveness, vigilance, situation awareness, robot modality, workload levels, and human error. The three characteristics of bad design to be avoided are rigidity, fragility and immobility of a system. The robot interface to be designed will both be user centred and efficient. This will be accomplished by using UCD techniques and design principles for the interface.
3 Requirements and design

3.1 System overview

Figure 2 illustrates the different components of the project and how those different components communicate with each other. Jonathan Dowling is responsible for robot vision and Jaco Colyn is responsible for the robot control. I am responsible for the Graphical User Interface (GUI).

3.2 Software

3.2.1 Robot vision
The robot vision system is used to detect different hazmat signs. This is developed by taking an input stream of video from the robots on-board cameras and then processing this data. The robot contains a Bosch video recorder which transmits the video stream through the robot’s on board wireless communications system. This video stream is then read to the remote computer system through the provided Bosch SDK. Once the video is received, it is processed in order to detect the various hazmat signs. The connection of the cameras to the Bosch box is illustrated in Appendix A.1.2 and more information on the Bosch SDK is discussed in Appendix B. The image processing is done with the help of the OpenCV software system. The general procedure is to transform the image’s data into a simpler representation and then apply a machine learning algorithm onto the transformed data to detect features.

3.2.2 Robot control
The robot control interface is developed in a modular approach in C++ as to let separate components, to be swapped out individually and to be tested and improved upon. The robotic control interface sends all
commands to the robot via TCP/IP socket messages passed over the wireless network, where the primary control system on the robot interprets these messages and forwards them to their respective sub-systems on the robot. The individual robot system messages, such as “drive forward at half-speed”, are sent to the various subsystems on the robot via various communications standards, such as serially through the RS-232 communications standard. Commands entered by the user controlling the robot are received via a game controller which is plugged into the base station PC/Laptop and the other commands are from GUI. The base station, which is a regular computer system, receives inputs from the controllers and processes them in the robot control interface. More information on the hardware devices being accessed by the robot control on the robot are discussed further in Appendix A. Appendix A also illustrates how the devices communicate with the user PC.

3.3 System Requirements and Design
A user centered approach was used in order to determine what the users need the system to provide. Users were involved in the first stages of design so that the probability of a usable interface increases as the user will have an idea of what the system provides. A Participatory Design Methodology was used to voice users in the system design process. The system was developed using an iterative and incremental approach. Three prototypes were made before the final system was developed. For each iteration; the design made in the previous iteration served as input to the requirement gathering of the next iteration.

3.3.1 Iteration 1

3.3.1.1 Requirement gathering
To gather initial system functional requirements a participatory workshop was done. The workshop was attended by eight MSc students in the Robotics and Agents lab. Their studies range from electrical engineering, electro-mechanical engineering and mechatronics. Those students will be the operators (users) of the system. The facilitator of the workshop was Tshitovha Zwivhuya who is involved in the development of the GUI of the robot.

3.3.1.2 Results and Findings for iteration 1
Participants were given an opportunity to think about what defines a usable interface and they came up with four properties which are:

- Learnability: How easy is it for them to accomplish basic tasks the first time they encounter the design?
- Efficiency: Once they have learned the design, how quickly can they perform tasks?
- Memorability: When they return to the design after a period of not using it, how easily can they re-establish proficiency?
- Errors: How many errors do they make, how severe are these errors, and how easily can they recover from the errors?
- Satisfaction: How pleasant is it for them to use the design?

Users were given the chance to raise current issues with the current robot interfaces. This was done by mentioning and discussing bad robot designs that have been built. The following interfaces were identified for having bad designs.
IRobot PackBot EOD control unit in Figure 3 is driven by two 6 degrees of freedom pucks. Depending on the task at hand in some cases the left puck drives the robot whilst the right puck controls the camera and in other cases the reverse is true. Sometimes a twist of the puck to the left rotates the flippers forward; in others a roll of the puck to the right rotates the flippers forward. Learning and memorization of this interface appears to have been a problem, an extensive “cheat sheet” had to be glued onto the control unit. The users said that the iRobot interface is difficult to understand and the controls of the robot are very confusing. Furthermore the interface also suffers from both high operator workload and inefficiency and is vulnerable to human error because it is difficult to learn.

The Toin Pelican interface in Figure 4 has 4 separate windows in one screen and different cameras are on different windows with separate controls. The users said that the Toin Pelican user interface suffers from both inefficiency and high operator effort with low throughput. The introduction of multiple windows to the interface introduces inefficiency as the operator has to move his/her head, eyes and focus of attention often in order to accomplish a task which causes the user to lose focus very quickly.
Both the IUT Microbot user interface and INEEL’s interface in Figure 5 have all required information on one window. The users said that those interfaces suffer from information overload, inefficiency, high operator overload and high operator effort with low throughput. In both cases the video feed is too small to place focus during navigation; this means that those interfaces fail to give what the users mostly require. This makes the two user interfaces difficult to navigate and use.

3.3.1.3 Requirement specifications and definition
Based on Results and Findings for iteration 1, participants identified usability goals that the system must meet. The following design goals were identified:

- Minimal windows must be used for the whole GUI
- The GUI must be easy to understand
- The controls of the GUI must be easy to understand
- Only useful information is shown
- GUI must not suffer from information overload

Use case Scenarios were defined that specified the tasks the user must be able to achieve on the GUI. This can also be referred to as the functional requirements. The following scenarios were identified:

- The user must be able to view the environment the robot is moving in using different cameras
- The user must be able to view the following information from the robot:
  - Carbon dioxide level,
  - Robot wrist position,
  - Gripping force,
  - Environmental heat,
  - Speed of robot,
  - Whether the robot is driving uphill, downhill or on a straight line and what the incline angle is,
  - Whether the robot is tilted or in level and at what angle,
  - The direction the robot is moving,
  - The significant events that have occurred both from the environment and the robot itself,
  - The battery life,
  - The signal strength of the wireless network and
  - The route that the robot has taken.
- The user must be able to give commands to the robot which are:
  - Set the force threshold of the gripper,
  - Set light brightness of all lights on the robot,
  - Set the wrist position threshold.
  - Switch on and off (cameras, speaker, fan, laser, microphone and carbon dioxide sensor)

3.3.1.4 First design
Based on the design goals and scenarios defined in the first iteration paper prototype GUI was made. The pictures of the paper prototype are shown below in Figure 6-10. Refer to Appendix A for the location of devices on the robot.

Figure 6 shows the camera view window paper prototype. This window is mainly used to access the camera streams from all the different 5 cameras.
Figure 6 Paper Prototype- Camera view window.

Figure 7 shows the Navigation tab on the information control window paper prototype. This window is mainly used to access information from the robot. The navigation tab shows a generated route map.

Figure 7 Paper information control Prototype- Navigation Tab on window.

Figure 8 shows the Warning tab on the information control window paper prototype. The warnings tab shows a log of significant events.

Figure 8 Paper Prototype- Warning tab on information control window.

Figure 9 shows the status tab on the information control window paper prototype. The status tab shows states of devices on the robot.

Figure 9 Paper Prototype- Status tab on information control window.
Figure 9 Paper Prototype- Status tab on the information control window.

Figure 10 shows the **arm** tab on the **information control** window paper prototype. The **arm** tab shows state of whole arm including gripper of the robot.

Figure 10 Paper Prototype- Arm tab on the information control window.

3.3.2 Second Iteration

3.3.2.1 Requirement gathering

After the first paper prototype was made a second paper prototype was constructed using Microsoft Word so that users can have an idea of what the interface will look like. Therefore the output of the first iteration (paper prototype) was the input for the requirement gathering for the second iteration. The pictures of this design are shown in Figure 11-15. Users were given the chance to identify any additions they needed to be done on the Microsoft Word paper prototype. Those additions serve as requirements for this iteration. The MSc students who participated in the participatory design workshop carried out this task.
Figure 11 shows **camera view** window Microsoft word paper prototype. The figure shows labels of places where additions where required. Extra additions are discussed in detail in the results and findings of iteration 2. The labels given on the sides are for controls that the users had concerns about.

![Figure 11 Microsoft word- Camera view.](image)

Figure 11 Microsoft word- Camera view.

Figure 12 shows the **navigation** tab on the **information control** window Microsoft Word paper prototype.

![Figure 12 Microsoft word -Navigation Tab.](image)

Figure 12 Microsoft word -Navigation Tab.

Figure 13 shows the **warnings** tab on the **information control** window Microsoft Word paper.

![Figure 13 Microsoft word –Warning Tab.](image)
Figure 14 shows the **status** tab on the **information control** window Microsoft Word paper prototype.

![Figure 14 Microsoft word –Status Tab.](image)

Figure 15 shows the **arm** tab on the **information control** window Microsoft Word paper prototype.

![Figure 15 Microsoft word –Arm Tab.](image)

### 3.3.2.2 Results and Findings for iteration 2

From the Microsoft Word design the following concerns were raised by the users. Refer to Figure 11-15 for the design.

#### 3.3.2.2.1 Camera view Window

Refer to Figure 11 for the design.
- The **signal strength** (A) must be shown as signal bars with percentages.
- The **mission timer** (B) must have a **reset** button that will enable the user to reset the timer to 00:00:00.

#### 3.3.2.2.2 Warnings Tab

Refer to 12 for the design.
- The **events**(A) are recorded in such a way the first event recorded is on top. This must change so the first event recorded is at the bottom.
Position and value fields must be added in order to give more information about an event. Position is the GPS position of where the event occurred and value is the “value” of the events.

A carbon dioxide concentration history graph must be added to give the user a graphical idea of the trend that the carbon dioxide concentration is taking and be able to analyse the trends.

Robot events and environmental events should be separated. This requires adding an extra tab.

### 3.3.2.3 Status tab

Refer to Figure 13 for the design.

- The on/off buttons (A) have to be tabbed, grouping cameras separately and the others together.
- The on/off buttons have to be made bigger to make it easier to click while using a touch screen.

### 3.3.2.4 Arm Tab

Refer to Figure 14 for the design.

- Radio buttons (A) are confusing with regard to selection. Users suggested the usage of selection boxes instead of a radio buttons.
- Gripping force (B) must be represented as a force gauge because it is easier to use and understand.
- The name of this tab (D) is too specific and should generalize. The name of the tab was changed to manipulator tab.
- Values on the wrist gauge (C) should be from -180 to 180.
- The following information also needs to be displayed: The speed at which the robot is gripping an object, the type of object the robot is gripping, the status of the manipulator either stopped, in progress, fully opened or fully closed, the distance between the fingers measured in centimetres and three circles that represent roll, pitch and yaw positions of the robot.

### 3.3.2.3 Second design

Based on the Results and Findings for iteration 2 a high level prototype GUI was implemented. The screenshots of the high level prototype are shown and described in Figure 16-21. The labels given on the sides are for controls that the users gave concerns about on the Results and Findings for iteration 2.

Figure 16 shows the high level prototype of the camera view window with additions discussed on Figure 11 and results and findings of iteration 2.

![Figure 16 High level prototype – Camera view Window.](image-url)
Figure 17 shows the high level prototype of the navigation tab on the information control window.

![Navigation Tab](image1)

Figure 17 High level prototype – Navigation tab

Figure 18 shows the high level prototype of the extra tab added, which is the robot info log on the information control window and the additions made in Figure 13 and results and findings of iteration 2. This tab logs events that occur on the robot itself.

![Robot Info Tab](image2)

Figure 18 High level prototype – Robot info tab.

Figure 19 shows the high level prototype warnings tab on the information control window and the additions made in Figure 13 and results and findings of iteration 2.
Figure 19 High level prototype – Warnings tab.

Figure 20 shows the high level prototype of the **manipulator** tab which was called the **arm** tab previously on the **information control** window and the additions made in Figure 14.

Figure 20 High level prototype – Manipulator tab.

Figure 21 shows the high level prototype of the **status** tab on the **information control** window and the additions made in Figure 15 and results and findings of iteration 2.

Additions

- **A**- Position field
- **B**- Value field
- **C**- $CO_2$ History graph

Additions

- **A**- Finger position
- **A**- Manipulator status
- **B**- Gripper status
- **C**- Set wrist position
- **D**- Gripper type selection combo box
- **E**- Operation mode selection combo box
- **F**- Object mode selection combo box
- **G**- Set wrist position
- **H**- Wrist gauge
- **I**- Set gripping speed numeric up and down
- **J**- Set gripping force gauge
- **K**- Manipulator tab changed
- **L**- Yaw
- **M**- Roll
- **N**- Pitch
Figure 21 High level prototype – Status tab.

Figure 22 shows the high level prototype of the **cam connection** settings on the **information control** window which is an extra tab for manually configuring camera ip addresses.

![Figure 22 High level prototype – Cam Connection settings tab.](image)

3.3.3 Final Iteration

The following requirement gathering was based on the high level software design that was developed from the second iteration. The GUI that was developed did not have any functionality but showed a wide range of feature to be provided by the system, which then gives a horizontal prototype. Requirement gathering was done by users who participated on the participatory design workshop (PD workshop
participants) and also by users who are new to the system and did not design the system (non-PD workshop participants).

3.3.3.1 **PD workshop participants**

3.3.3.1.1 Requirements Gathering from PD workshop participants

The PD workshop participants had to jointly give their comments and problems with the design and give their opinions on high level prototype.

3.3.3.1.2 Results and Findings from PD workshop participants for final iteration

- The tab names are very small fonts and it’s difficult to read on the screen.
- **Camera view window:** There must be a push to talk button so that the operators must be able to communicate with victims. (see Figure 16)
- **Navigation tab:** A speedometer (0-20km/h) must be displayed on this tab.
- **Warnings tab:** The carbon dioxide concentration history graph is required to be real time moving graph instead of static. The graph must also show time stamps. The concentration of carbon dioxide from human can be way more than 400ppm therefore the scale must increase since is used mainly to detect carbon dioxide form humans. (see Figure 18)
- **Manipulator tab:** The gauges that represent roll, pitch, and yaw are not intuitive at first glance. Instead of manually drawing the robot, rendered pictures of the robot should be used and rotated. Those gauge circles are also quite small. (see Figure 19)
- **Manipulator tab:** Because a touch panel will be used the controls used to set the wrist and force threshold are too small to touch. Since the wrist rotates continuously the gauge used to set the threshold should move from -180 over to 180. The wrist gauge needle must not change colour. The force gauge only needs to change colour on the needle not the numbers. (see Figure 19)
- **Status tab:** The on/off buttons must not be tabbed because the operator must be able to see all the on/off buttons at once. (see Figure 20)
- **Status tab:** A speaker volume control should be added next to the mic volume control. (see Figure 20)
- The GUI colours are not appealing. Having the GUI colours blue and black it’s too much on the eyes and the other reason being that standard robot interfaces are normally not that colourful. The standard colours that should be used are white and grey. Red and green must only be used for notification or emphasis on something.

3.3.3.2 **Non-PD workshop participants**

3.3.3.2.1 Requirements Gathering from Non-PD workshop participants

The second requirement gathering that used the high level prototype was done by users that did not participate in the PD workshop. Those users where required to accomplish 13 predefined tasks on the GUI that would help in identifying the usability of the design. The 13 task are as follows:

- **Task 0:** Connect to all cameras
- **Task 1:** You are driving the robot and you would like to change the main camera view form live camera to front camera.
- **Task 2:** The robot has entered a very hazardous area and you would like to immediately halt the movement of the robot.
- **Task 3:** You are driving the robot in a mission, how would you make the timer restart timing from 00:00:00.
- **Task 4:** You would like to save the route that the robot has taken so far.
- **Task 5:** You would like to save events that have occurred in the robot’s surrounding.
- **Task 6:** You would like to save events that have occurred in the robot itself.
Task 7: Switch off front camera.
Task 8: Switch on the fan.
Task 9: Set the gripping force threshold to 5.
Task 10: Select automatic gripping.
Task 11: Select that you want to grip wood.
Task 12: What do the three small circles in the manipulator tab represent?

After accomplishing all the tasks the users were required to comment on the look and feel of the GUI and suggest any additions on the GUI.

3.3.3.2.2 Results and Findings from Non-PD workshop participants for final iteration

These tasks were carried out by five fourth year engineering students who did not design the system. Below is a summary on what actions the users did when they were told to perform a specific task. Each task below shows different actions by the five users. Please note that if the actions described in each task is less than five this means that the user was able to accomplish the task with ease or that users had the same problem. Each point under the task is the action that a specific user did.

Task 1: You are driving the robot and you would like to change the main camera view from live camera to front camera.
- Though the user was able to accomplish the task he said that he should be able to click on the video feed because it’s convenient.
- The user clicked on “the connect to camera” button. He said he couldn’t tell that the button above the video feed can be clicked because it’s not separate from the video feed.

Task 2: The robot has entered a very hazardous area and you would like to immediately halt the movement of the robot.
- The user clicked on the image of the emergency stop because he said it looked clickable.
- Though the user was able to accomplish the task he said that there should be a short cut such as the keyboard key Esc because it is more usual to do that.
- The user said that in real life emergency buttons are normally in different shapes so that they are easier to find in critical conditions.

Task 4: You would like to save the route that the robot has taken so far.
- The user said that the tab names are quite small and it’s difficult to see.
- The order of the buttons in navigation should be standard; that is new, save and then load map.

Task 5: You would like to save events that have occurred in the robot’s surrounding
- The user went to status tab. The user said that the name of the tab does not make sense. The user suggested that the name should be changed to environmental info log.
- The user did not know where to go. He suggested that the name should be changed to surroundings because a warning implies agent not event. He also suggested that the carbon dioxide history graph should be above events because visual ways are more appealing and he sees things way quicker in graphs.
- Though the user was able to accomplish the task after a while he said that name warning refers to something that would cause errors on the system and he suggested that the name should be changed to observed indication.

Task 7: Switch off front camera
The user clicked on the cam connection settings and secondly thought switching off the camera would be on the camera view window. The user right clicked on the video feed he wanted to switch off.

- The user said that he did not draw a link between status tab and camera on/off switch.
- The user thought switching off the camera would be on the camera view window but concluded that the idea that it’s a separate button is fine..
- The user said that he had no idea what to do. He said that if there was a radio button might help.

**Task 8:** Switch on the fan.
- The user did not know where to find the button to switch on the fan. He said that the actual place where the fan switch is situated is ambiguous because a fan is not a sensor. He also pointed out that a speaker is also not sensor and should not be labelled as sensor.
- The user did not know where to find the button to switch off the fan. He suggested that the on/off buttons should be un-tabbed so that it’s quick to access.
- The user suggested that the on/off buttons label should be changed to peripherals.

**Task 9:** Set the gripping force threshold to 5.
- The user took some time to find the down and up arrows to set the wrist position. The user said he thought that the actual wrist gauge needle can be grabbed and moved.
- The user suggested that the up and down arrows should be made bigger because it’s difficult to click.

**Comments:**
- The user said that the green and red colour used on the on/off buttons gives an idea of the actual on and off of the buttons.
- The user said that the system should be used on two screens because it would be very annoying to minimize and maximize while navigating between the two windows.
- The user said it would be very useful if the buttons are 3D; that is raised from the surface so that he can tell that they are clickable
- The user suggested that the wrist and force gauge should be side by side.

### 3.3.3.3 Final design

The final system that will be implemented is based on Results and Findings for the final iteration from both Non-PD workshop participants and PD workshop participants. The final system is discussed in section 4.6

### 3.3.3.4 Final System Analysis model (Use case diagram)

After the specification of the final system was gathered using participatory design, paper prototyping and horizontal prototyping, a use case diagram was developed. This provides a graphical overview of the functionality provided by the system in terms of actors (the operators), their goals (represented as use cases), and any dependencies between those use cases. Figure 23 and 24 shows the use case diagrams of the rescue robot user interface. Figure 23 is the functionalities provided by the camera view window and Figure 24 is functionalities provided by the information control window.
Figure 23 use case diagram for CameraView Window
3.3.3.5 Final System Analysis model (Class diagram)

Classes and Objects were extracted from the use-case narratives by doing a grammatical parse. The use-case narratives were based on the use case diagrams in Figure 23 and 24 grammatical parse involves underlining nouns or noun clauses which describe candidate objects, entering the candidate objects and their associated classed into a table, removing synonyms [11]. An object was accepted to having a class if it met the following requirement [11]:

- Retained Information: the system needs to remember data about the object
- Needed Services: the object must have an identifiable set of operations that can change the value of its attributes
- Multiple Attributes: Objects should have many attributes
- Common Attributes: A set of attributes apply to all occurrences of the object
- Common Operations: A set of operations apply to all occurrences of the object
- Essential Requirements: external entities that produce or consume essential information

Based on the classes that were identified two class diagrams were created. A class diagram shows the static structure of the system model, the entities that exist, the internal structure and, relationship to other entities but does not show temporal information [11].

Figure 25 shows a class diagram of the camera view window. Since camera view window is form and has a number of attributes and operations, only the attributes and operations that are not windows form controls are
Figure 25 Class Diagram for CameraView Window

Figure 26 shows a class diagram of the **information control** window. Same with **information control** window, only the attributes and operations that are not windows form controls are shown.
Figure 26 Class diagram for Information control Window

The features that those classes provide are discussed in section 4.6.
3.4 System constraints

Even though all the hardware has been made, not every hardware component has its own microchip for control that will interface in high level language with the robot control interface. This is because the rescue robot project is a work in progress that involves MSc students in the engineering department. This would then mean that some information that is required by the system will not be available. The following devices do not have control to test with: (see Appendix A)

- Wrist
- Arm
- Gripper
- Robot base

All the other un-mentioned devices do have microchips for control and will be tested with the system.

Even though the above mentioned hardware devices are not available, this does not affect the implementation of a usable user interface. In order to continue with the project with the unavailability of hardware control a server that serves as a robot simulator was implemented and this server provided dummy valid required input. This means that the server is the “rescue robot”.

When the user sets a set action, the GUI sends the request to the server; but this will not have an effect on the hardware itself. If the system has been developed and the engineers need to test the hardware that they have made, this can easily be done since it’s a matter of deciding which number of bytes they want the robot control server to send to the robot which acts as a signal. Since some hardware devices do have controls, the robot control that is being implemented by Jaco Colyn will be used to communicate with those devices. Please note that all the required back end of the GUI is available and functional but has no effect on the robot hardware even though a command is sent.

Initially in the project proposal it was proposed that the user interface will be developed in the Scaleform GUI kit in C++. After doing research on this product, the product is commercial and thus could not be accessed. After the requirement gathering of the system was done, it was concluded that visual c# is the best feasible platform to develop the interface with the required functionalities by the users. The other reason of not using Scaleform is that the platform is intended for game industry and going this route would have placed a huge workload based on the time available to develop the interface.
4 System development and implementation

4.1 Introduction
In this chapter the features provided by the system will be discussed in detail and how the features where implemented. If the feature is basic visual c# control no further explanations will be placed on that control but the customized ones will be discussed.

4.2 Bosch SDK features used
Detailed explanation on the Bosch SDK is discussed in Appendix B.

4.2.1 Video
The device connector sends a connection request to the Bosch box by providing the camera URL. The connection to be established is asynchronous. The connection will wait for 10 seconds for the connection to be established before the connection is timed out. If a connection was established before 10 seconds an event is triggered by the Bosch box and handled by the method OnConnectResult.

If the result obtained from the Bosch box is equal to creConnected or creInitialized then this means that a connection has been established. The device proxy that is sent back from the Bosch box has a collection of video inputs. If the size of the collection is zero, then no streams are available. It is the responsibility of the user to define the index of the video input in the connection settings tab. Based on the user settings, a data stream is obtained from that video input and then displayed on the Active-X container. The stream that is set to the cameo Active-X container is continuous. This procedure is done for all the cameras connected to the Bosch box.

4.2.2 Audio input and output
The same process is done as explained in section 4.2.1. The difference is that an audio input collection is used instead of the video input. Also instead of setting the stream to be displayed on the Active-X container the stream is set to an audio receiver. An audio receiver represents the speaker of the user pc. When connecting to the audio output; the difference form section 4.2.1 is that the audio output collection is used instead of the video input. Also instead of setting the stream to the Active-X container, the stream from the audio source is set to the audio output stream (robot speakers). The audio source represents the microphone of the user pc.

4.3 Robot control server
To communicate with the robot control server TCP/IP socket communication is used. The GUI initiates the communication in all instances. The GUI can either be setting or getting values from the robot control. Each component is associated with Identification.

- Set message: “set ID value”
- Get message: “get ID”

For example to set the brightness of the spread light to 50%, the message sent to the robot control would be “set 9 50”. Since the robot control server is written in C++, the string to be sent must be encoded to ASCII bytes instead of just characters. When data is received from the robot control server an inverse function is used to convert from bytes to strings. Before data is sent to the server, the server has to be alive. The system will suppress error messages obtained from trying to connect to the robot control server and after ten seconds have passed; a message error box will appear and ask the user if he or she would like to reconnect to the server or terminate the program.

4.4 Robot vision
In order to display the detected hazmat signs from the robot vision component, the GUI receives the processed image data via a named pipe. The robot vision is in C++ and runs in a separate process. The
GUI, built in C#, opens a named pipe server and creates a separate thread that waits for a connection and the image data from the robot vision component.

During the timeline of the project, the Bosch had restrictions on the accessing of raw image data. This meant Jonathan Dowling could not access the data from the Live, Front and Rear camera that required hazmat detection. In order to continue with the project Jonathan worked with image data from a web cam. To include this on the GUI, a separate copy of the GUI was used that takes in image data from a both the web camera and Bosch box.

4.5 Adherence to Design principles on the GUI

In implementing the GUI design principles were taken into consideration. Design principles are essential for helping in improving user acceptance and usability of software. The design principles that were taken in consideration are discussed in detail in the background chapter. This section discusses how design principles were applied on the GUI.

4.5.1 GUI controls

In keeping with the principle of affordance (see Section 2.4.1), the buttons in the GUI are raised from the flat surface which gives a perception to the user to push the button. This also gives the button a different contrast from its background and thus giving at a stronger visual weight (see Section 2.4.13). All buttons rollover, that is they change colour as the mouse is over the button to indicate to the user that the object can be pushed. Figure 27 shows an example of a raised and rollover button respectively.

![Raised Buttons and Rollover Buttons](image)

**Figure 27 Raised Buttons and Rollover Buttons.**

All names of buttons refer exactly to the devices that they represent. This means that the user will not press one button thinking it does something else due to the ambiguity of the names; this is the Principle of least astonishment (see section 2.4.10). Also all Menu items and other controls have an effect that is obvious based upon their text labels and placement relative to other control.

The on/off buttons are red if the devices they represent are switched off and green light means the devices are switched on. Naturally it comes to humans that red lights require attention and refer to something that’s off or refers to “stop” and green lights refer to “go” or something that’s switched on. This is also because red colour has a stronger visual weight and green colour is has small visual weight than other colours (see Section 24.13).

Figure 28 illustrates an on/off button for the live camera. With adherence to the principle of Recognition vs. Recall(see Section 2.4.8); the button is both has labels and is icon based with an icon of a camera and the name of the button refers to what it represents.

![on/off button](image)

**Figure 28 on/off button.**
In keeping with the principle of visibility (see Section 2.4.3); instead of displaying data merely as text the GUI displays most of the retrieved data graphically and this makes it easier for the user to understand. This also makes the system less complex (see Section 2.4.6).

Since a touch screen will also be used on the system, the buttons and controls on the system are made big enough for the user to access using their fingers. Thus in Fitts’ equation (see Section 2.4.12) the width (W) of the control will be bigger and thus decrease the movement time for the user to hit the target. This also gives the big controls stronger visual weight (see Section 2.4.13).

### 4.5.2 Grouped controls

Users are used to having wording in alphabetical order. In keeping with the principle of affordance (see Section 2.4.1); all grouped control items on the GUI are placed in alphabetical order so that it is easier and quicker for the users to obtain what they want. Figure 29 shows examples of three grouped controls in alphabetical order.

![Figure 29 Carbon dioxide concentration, microphone level and speaker level bars.](image)

### 4.5.3 Input Constraints on controls

In keeping with the principle of constraints (see Section 2.4.2); the set **speed numeric up and down** (see Figure 30) only allows the user to enter numeric numbers up to a maximum of 20. All the **selection combo boxes** (see Figure 30) are not editable, so that user can only select from the predefined valid options. The text boxes that show the manipulator status and gripper status are all un-editable. This also makes sure that the system adherence to the **Role integrity** principle (see Section 2.4.9).

![Figure 30 Input constraints](image)

### 4.5.4 GUI controls Mappings to actual representation

With adherence to the principle of mapping (see Section 2.4.4); the 2D model illustrated in Figure 31 has been drawn in such a way that it mimics what the actual robot looks like. This makes it easier for users to understand the position of the arm of the robot.
The three orientations illustrated in Figure 29 are represented by the actual position of the robot, and how the user would actually look at it to determine if those three orientations have changed.

4.5.5 GUI user Feedback

With adherence to the principle of providing feedback (see Section 2.4.5); when every button is clicked the button makes a slight movement to show that it has been clicked. And when other buttons are clicked they change color to show that something has been adjusted or changed. Figure 33 shows a home button before and after it has been clicked.

In the implementation of the GUI, when the user clicks to save the connection setting (see Figure 31) on the connection settings tab the user will receive feedback of whether the settings were saved or not.

---

**Figure 31**  Mapping between the actual robot and the 2d model

**Figure 32**  Roll, Pitch and Yaw orientation representation

**Figure 33**  Before and After the Home Button is clicked.

**Figure 34**  Feedback from saving connection settings
4.5.6 Dealing with errors on the GUI
To prevent the users from encountering error; all the errors that are known in advance are dealt within the system and the user will not be aware of this.

4.6 Features

4.6.1 Camera view Window
Figure 35 shows a screenshot of the camera view window. The camera view window shows five video streams from five different cameras namely the depth, front, live, thermal and rear camera.

4.6.1.1 Features
Refer to Figure 35. If the robot is in a mission the user can time missions using the mission timer (A) with play, pause and stop capabilities. The timer times the actual time that the robot has been sent to a mission. This helps the user in measuring the how long the robot can last in the mission with respect to battery life or any other time measurable components. The battery life (E) of the robot shows the percentage of the amount of battery life that is left. The actual percentage of the battery life can be obtained by placing the mouse over the battery control. The signal bars (F) show the percentage of the wireless signal strength of the place that the robot is situated. The colors for the signal strength and battery life percentage are different with respect to their values. Figure 36 shows the color code for percentages.

<table>
<thead>
<tr>
<th>Percentage value</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>$50 \leq value$</td>
<td>Lime</td>
</tr>
<tr>
<td>$20 &lt; value \leq 50$</td>
<td>Yellow green</td>
</tr>
<tr>
<td>$10 &lt; value \leq 20$</td>
<td>Yellow</td>
</tr>
<tr>
<td>$value \leq 10$</td>
<td>Red</td>
</tr>
</tbody>
</table>

Figure 36 Color code for percentage values for signal and battery life

These colors serve as notification to the users with regard to the criticality of the battery life and the signal value percentages.
The **compass** (B) maps the input $0^\circ - 360^\circ$ to the actual compass value i.e. N, NNE, NE, ENE, E, ESE, SE, SSE, S, SSW, SW, WSW, W, WNW, NW, and NNW. Since there is a large pool of all compass values, only those above mentioned are visible. But if the input does not have a label to match, no label name will be shown. For example, say input is 33°, this value will be shown between NNE($22.5^\circ$) and NE($45^\circ$) instead of writing the actual compass.

The **robot yaw** (D) is explained in section 4.4.5.1. **Emergency stop** (L) is used to immediately halt the robot movement to prevent the robot from getting into a hazardous area.

To connect or disconnect to all cameras, the user has to use **connect / disconnect all** button (C). But if the user wants to connect to a single camera, the user simply right clicks the control of the desired camera, and a context menu will appear where the user can click connect and that single camera will be connected. From the context menu, the user can switch on/off, connect/disconnect, or select the camera.

The buttons with the camera labels can also be used to select a camera to view on the main view (R). The difference between **connect** and **switch on** action is that **connect** establishes a network connection with the camera on the robot and obtains a camera stream, whereas **switch on** will physically switch on the camera on the robot. The green light on the button shows that the camera is connected and red light is not connected.

**Pan and tilt** (Q) is for the live camera only. The left and right arrows are used for panning and the up and down are used for tilting. The **picture type** (P) gives the user an option to select the type of image they want to see from the live camera – either mirrored image, flipped image or both. The **mouse wheel control** (O) gives the user an option to use the mouse wheel for focusing or zooming the live camera.

The **audio input** (N) is used to receive an audio stream from the robot, so that the user can be able to hear any “help shouts” from victims. The **audio output** (M) enables the user to send audio stream to the robot, so that the user can talk to victims that are alive or even shout out so that victims can hear that there is help around.

### 4.6.1.2 Features implementation

- **Video and audio**
  
  Video and audio input/output is discussed in section 4.2.

- **Compass** (B)
  
  There are 16 possible visible compass labels that were mentioned in section 4.6.1.1. Those compass labels are drawn on a control of 180 pixels width. Initially, those 16 labels are given the $(x, y)$ point to be drawn at. One label’s x-coordinate is 45 pixels from the previous label’s x-axis. It can be seen that $45 \times 16 > 180$ and this makes sure that not all the labels are visible at all times because some labels will be out of range. The input compass value from the robot determines how much the labels’ x-coordinate will change by i.e. $\text{range} = \text{input}/(360/16)$, which is the offset of the input from the “N” (north) label.

- **Depth camera**
  
  The input from the depth camera is the distance of objects in millimeters from the camera. The range of those input values is $500 - 10000$ mm. Between RGB ($255, 0, 0$) and RGB ($0, 0, 255$) there are 1020 different possible colors. The depth camera information is mapped to an RGB value. The mapping is given by dividing the distance by 10 which will always give a number between $50 - 1000$. And this can be easily mapped to an RGB value where blue represents the closest distance and red is the furthest distance. Having the corresponding RGB values, this can easily be drawn on a control.
4.6.2 Navigation tab

Figure 37 shows the navigation tab on the information control window. This tab is mainly used to view the route the robot has taken drawn as a map.

![Navigation tab features](image)

A- Zoom slide bar  
B- Speedometer  
C- New map  
D- Save map  
E- Load map  
F- Current map  
G- Route obstructions (blue dots)  
H- Robot (grey square)  
I- Robot route (red dots)

4.6.2.1 Features

Refer to Figure 37. As the robot drives through a mission or a designated area an approximate route map is generated. If the points to be shown are out of the range of the screen, the interface automatically scales all points so that all points are visible. The user also has the capability of manually zooming using zoom slider (A). The user can zoom from the original point scaling factor to two times the size of the map. The size of the points do not change; only the proportion of the distances of all points from the centre. When the user is zooming some points might move out of the screen. If that happens vertical and horizontal scroll bars will appear to enable the user to move up or down and right or left respectively. The initial position of the robot at first run of the program is assumed to be at the centre.

New map button (C) allows the user to clear up the current map that had been generated and create a clean space for a new map to be generated. Save map (D) button allows the user to save the currently generated map to a .Map file. Load map (E) button allows the user to load a previously generated map from a .Map file. Current map (F) button allows the user to go back to the current map if the user was viewing previously generated maps.

4.6.2.2 Feature Implementation

4.6.2.2.1 Speedometer Gauge (B)

- Speed Gauge

Refer to the navigation tab screenshot in Figure 37 label B. Every gauge on the GUI is represented as a circle in aXY Cartesian plane. Angles are measured as shown in Figure 38 in clockwise direction. Therefore every speed value on the gauge is associated with an angle. When a speedometer value is obtained it has to be converted into angles, and this makes it possible to determine the position of the speedometer gauge needle. The equation that is used to determine the position of the gauge needle is

\[
\text{speed}_{\text{degrees}} = (\text{speed} + 8) \times 15 \quad (2)
\]

The logic behind choosing this formula is based on mapping of where the 20km/h is and where 0km/h is on the speed gauge and also the measuring of angles in Figure 38.
Angles are measures clockwise from the positive side of the x-axis as shown in Figure 38. This is because on a visual c# controls the y-axis increases going downwards. We only need 300° of the full 360° for the speedometer gauge because there is 60° space between the position of 0km/h and 20km/h. Therefore to get the divisions or the distances between values we have 300/20=15 parts. And this explains the 15 in equation (2).This means that from the 0 at 0° mapping the mapping has moved by 120° and this explains the shift by 8 in equation (2) which is 120/15.

**Speed gauge needle**

Refer to the navigation tab screenshot in Figure 37 label B. The speed gauge needle is represented by a triangle. The triangle is drawn using three points determined based on the angle or value that the gauge needle is pointing. The points’ calculation is shown in Figure 39. The gauge needle can either be in the first, second, third or fourth quadrant of the XY — Cartesian plane.

Figure 39 illustrated the speed gauge position in the first quadrant with speed values greater than 16km/h. The input angle is $\theta$ which has a value of $\theta$ .Triangle$abc$ in Figure 39 is an isosceles triangle having a perpendicular bisector $ha$. Angle $h\hat{O}g = \theta$ because they are vertically opposite, angle $g\hat{O}b = 90° - \theta$ because $ha$ is a perpendicular bisector. And angle $e\hat{O}c = 90° - \theta$ because it is vertically opposite to angle $g\hat{O}b$. 

![Figure 38 Cartesian plane of the gauge and XY — Cartesian plane of the gauge showing the position of 0km/h and 20km/h respectively.](image)

![Figure 39 Speed Gauge needle on the first quadrant of the XY — Cartesian.](image)
The three points for the triangle are determined as follows where $(center.X, center.Y)$ is the origin in Figure 39 which is the center of the gauge given $\left(\frac{gauge\_width}{2}, \frac{gauge\_height}{2}\right)$ and $radius = \frac{gauge\_width}{2}$.

- If $a\theta e < 90^\circ$
  \[
  \begin{align*}
  Point_a.X &= center.X + radius \times 0.9 \times \cos \theta \\
  Point_a.Y &= center.Y + radius \times 0.9 \times \sin \theta 
  \end{align*}
  \]

- $Point_b.X = center.X + radius \times 0.9 \times \cos(180 - (90 - \theta))$
  $Point_b.Y = center.Y + radius \times 0.9 \times \sin(180 - (90 - \theta))$

- $Point_c.X = center.X + radius \times 0.9 \times \cos(360 - (90 - \theta))$
  $Point_c.Y = center.Y + radius \times 0.9 \times \sin(360 - (90 - \theta))$

The same procedure is used to calculate the points of the speed gauge needle on the second, third and fourth quadrant with different angles.

4.6.2.2 Robot map
To draw the map on the GUI the input provided is the distance and angle of the current position of the robot and the distances and angles of objects in front of the robot with respect to the initial position (center). This is drawn using the equation of the circle given by $r^2 = x^2 + y^2$ where radius $r$ is the distance of the object from the robot.

4.6.2.3 Zooming (A)
Refer to the Navigation tab screen shot in Figure 37 Label A. The zoom slider (A) is drawn on a standard button because this prevents the drawing from flickering. When the user drags the slider the new position is calculated by first obtaining the $x$ position of the cursor and then dividing this by the width of the button to get the zooming fraction between 0-1.

4.6.3 Robot info log tab
Figure 40 illustrated a screenshot of the Robot info log tab on the information control window. This tab keeps a log of events on the robot itself.

![Figure 40 Robot info log tab on the information control window](image-url)

**4.6.3.1 Features**
When the robot is on operation, warning, errors, cautions or information that is happening on the robot itself might be of interest to the operator. If events occur, they are logged in a log grid (C). The information that is logged is the name of the event, the time that the event occurred, the GPS position of where the event and a value associated with the event if there is one. The user has the choice of either
clearing or deleting all the events on the log grid using (A) or the user can save the events in a text file with a specified location using (B).

4.6.4 Environmental info log tab
Figure 41 shows a screenshot of the Environmental info log tab on the information control window. This is the log of surrounding events.

**Figure 41 Environmental info log tab**

4.6.4.1 Features
This tab is the same as the robot info log. The difference is the events in this case are events that are happening in the surrounding of the robot. This tab is mainly used to give the user a sense of the environment the robot is currently in.

Carbon dioxide concentration is measured from the CO₂ sensor on the robot and reported every 100 milliseconds. A history of this reading is kept and is used to draw a real-time CO₂ history graph (D). Time [CO₂] is the current central time that the CO₂ was recorded. The value of the CO₂ concentration is measured in Parts per Million (PPM). The CO₂ concentration line rescales automatically based on the largest CO₂ value it has recorded. If the user wants to know the exact value of the CO₂ concentration at a point the user can put the mouse pointer at that point and the value of the CO₂ concentration there will appear.

4.6.4.2 Features implementation

4.6.4.2.1 Carbon dioxide history graph (D)
Refer to Figure 41 Label D. The graph can only display up to 50 x-coordinates at a time which are clearly labeled. The positions (xₖ, yₖ) of 50 points with the same y coordinate are stored and represent the point to draw the line which has time label. Those points are 50 pixels on the x-axis apart. Every millisecond the x-coordinate of the points decreases by 6 pixels until the first point has an x-coordinate of −11. At that point all points are shifted down i.e. \( \text{pos}[i] = \text{pos}[i+1] \). For the last point the robot control server is contacted to obtain the next \( \text{CO}_2 \) value. The x-coordinate of \( \text{pos}[49] = \text{pos}[48] + 50 \). To have a graph, a line is drawn from \( \text{pos}[i].x, \text{co2value}_i \) to \( \text{pos}[i+1].x, \text{co2value}_{i+1} \) for all the 50 points.
4.6.5 Manipulator tab

Figure 42 shows a screenshot of the Manipulator tab on the information control window.

![Manipulator Tab Features](image)

**Figure 42 Manipulator Tab features**

### 4.6.5.1 Features

Refer to Figure 42. The main purpose of the **2D image of the robot** (A) is to show the user the orientation of the different part of the arm. The diagram shows the operator the angle at which the top, middle and bottom part have tilted. The arm can move from 0-360 degrees. The **grippers** (fingers) can only move 0-90 degrees. If both fingers have rotated 0 degrees the gripper is fully closed and if they have rotated 90 degrees is fully opened. The three different positions (H) are **home**, **user** or **driving**. Driving and home positions are shown in Figure 43. When the robot is at home position; the bottom part is at 0°, the middle part at 0° and top part is at 0°. When the robot is at driving position; bottom part is at 0°, the middle part at 45° and top part is at 0°. When the robot is at user position the angles of the three parts are defined by the user and obtained from the robot.

![Driving and Home Position](image)

**Figure 43 Driving and home position respectively**

The **fingers** information shown in (A) represents the distance between the two **fingers**. If the user has selected the **adaptive gripper** there will not be any finger information because the adaptive gripper grasps anything that gets between its fingers and it adapts automatically to any shape.
The **wrist rotation gauge** (D) is used to view the actual angle at which the wrist has rotated and also the user defined threshold that the robot’s wrist rotation must not exceed. To set the threshold the user must click on the wrist gauge, and then drag the **wrist threshold needle** to the desired threshold value. To stop dragging and thus setting the value, the user must right click. The wrist threshold needle is aqua colored and the real wrist value needle is steel blue colored.

The user can also set the gripping force threshold in the same way as the wrist value threshold. The **force gauge needle** will change color as the gripping force increases, this serves as an alert to the user about the pressure being placed on an object being gripped.

**Gripper configuration** (B) enables the user to select the type of gripper to use; adaptive, angular or parallel. The user can also choose to operate the gripper in automatic or manual mode. If the user wants to grip an object, the user must specify what type of object to grip (paper, wood or glass).

In order to give the user information with regard to the orientation of the robot, the user is presented with the roll, pitch and yaw position of the robot. Yaw refers to the direction in which the robot is facing; that is its orientation within the $xy$ plane, or rotating around the $z$ -axis. Roll refers to whether the robot is upside-down or not; that is its orientation within the $yz$ plane, or rotating around the $x$ — axis. Pitch refers to whether the robot is tilted; that is its orientation within the $xz$ plane, or rotating around the $y$ axis. Figure 44 illustrate the roll, pitch and yaw orientations on a $xyz$ —plane.

![Figure 44 Roll, Pitch and Yaw positions](image)

4.6.5.2 **Features implementation**

4.6.5.2.1 **2D-robot image (A)**

Figure 45 illustrates the 3D model of the robot in 2D$xy$-plane.

![Figure 45 3D shaped Robot image in a 2D $xy$ —plane](image)
The drawing in Figure 45 is constructed from five 3D shaped rectangles connected together. Four 3D shaped rectangles form the complete robot and the fifth is the head. Figure 46 illustrates how a 3D shaped rectangle is manually drawn on a 2D plane.

Figure 46 The steps to be taken while drawing a 3D cube on a 2D surface

The first step is to draw a normal \( H \times W \) rectangle on the 2Dxy plane starting at position \((x_1, y_1)\). Secondly draw a second rectangle starting from position \((x_1 + W/2, y_1 + H/2)\) with the same width and height as the first one. Then connect the corresponding Conner’s of the first rectangle to that of the second rectangle using lines. To make sure that the lines \(a_1a_3\) and \(a_2a_4\) in Figure 46 are hidden; that is to make it a closed 3D shaped rectangle; a color filled polygon with points \(a_1, \ldots, a_4\) must be drawn to fill the space. The same can also be done in the top section of the rectangle. In total six rectangles are required for full closed 3D rectangle.

⚙️ Robot arm

Figure 47 illustrates the structure of the robot arm and the angles being measured on the arm.

Figure 47 Reference point of measuring angle of robot arm on the robot.

From Figure 47\(\theta_1, \theta_2\) and \(\theta_3\) are the angles that the bottom, middle and top part have rotated respectively. These are the actual inputs from the robot.

⚙️ Bottom part (E)

Refer to Figure 45 Label E. The first rectangle which is the reference for all others is drawn using the points defined below with \(p = (171,171)\):

- \(Point_0.X = p.X + 50 \times \cos(360^\circ - \theta_1)\)
To draw the rest of the bottom part and make it 3D shaped, the procedures discussed in Figure 46 are followed.

- **Middle part (D)**
  Refer to Figure 45 Label D. The same procedure is used to draw the middle part but now we have to define what \( p \) is. Point \( p \) is defined as:\[ p.X = Point_3.X - 10 \text{ and } p.Y = Point_3.Y \] \( Point_3 \) is the point of the top part. The reference rectangle is drawn in the same way as that of the bottom part. The difference is the value of \( p \) and that the angle being used is \( 180 + \theta_2 \).

- **Head (G)**
  Refer to Figure 45 Label G. The reference rectangle used is drawn from \((Point_3.X - 5, Point_3.Y - 40)\) of the middle part and the rectangle has \( W = 10 \) and \( H = 20 \). Since the diagram’s primary use is to show the movement of the arm this part is static. This part helps to see where the front of the robot is. To draw the 3D shaped rectangle; steps in Figure 46 are used.

- **Top part (C)**
  Refer to Figure 45 Label C. The first rectangle which is the reference for all others is drawn using the points defined below where \( rect \) refers to the reference rectangle of the head (G):

  - \[ Point_0.X = rect.X \]
  - \[ Point_0.Y = rect.Y + rect.Height \]
  - \[ Point_1.X = rect.X + rect.Width / 2 \]
  - \[ Point_1.Y = rect.Y + rect.Height \times 3 / 2, \]
  - \[ Point_2.X = rect.X + rect.Width / 2 + rect.Width \]
  - \[ Point_2.Y = rect.Y + rect.Height \times 3 / 2) \]
  - \[ Point_3.X = rect.X + rect.Width \]
  - \[ Point_3.Y = rect.Y + rect.Height \]

  The formulas used above are normal circle formulas
  i.e. let \( (a,b) \) be the center of the circle then a point on the circle with radius \( r \) is given by \( x = a + r \cos \theta \) and \( y = b + r \sin \theta \)

- **Fingers (B)**
  Refer to Figure 45 Label B. Figure 48 illustrates the separation of the fingers (gripper) from each other and the angles that are measured. In Figure 48 AB is finger1 and AC is finger2.
The input from the robot is the \( \text{Finger}_{1\theta} \) which is the angle the first finger has made with the top section of the arm and \( \text{Finger}_{2\theta} \) is the angle for the second finger. To determine the distance between the two fingers we have to determine the value of \( BC \) . It is given that the length of each finger is 10mm which is equivalent to 37.8 pixels. Therefore, triangle \( BAC \) is an isosceles triangle since \( AB = AC = 37.8 \) pixels. This then follows that angle \( \hat{A}BC = \hat{A}CB \). Then we have that

\[
\hat{A}BC = \hat{A}CB = \frac{1}{2} \left( 180 - (\text{Finger}_{1\theta} + \text{Finger}_{2\theta}) \right)
\]

Using the law of sines we obtain that the distance between the fingers can be obtained from:

\[
\frac{37.8}{\sin \hat{A}BC} = \frac{BC}{\sin(\text{Finger}_{1\theta} + \text{Finger}_{2\theta})}
\]

4.6.5.2.2 Robot orientation (E)
Refer to Figure 42 Label E, F and G. The orientation of the robot is shown via roll, pitch and yaw. This is done by having pictures rotating inside gauges. The input from the robot will be the angle at which the robot has titled.

Figure 47 shows the rest position of the robot for roll, pitch and yaw orientations respectively on the \( XY \) - Cartesian plane.

To synchronize the angles with the real Cartesian plane the input angle is changed to \( 270^\circ + \theta_i \). With relative to the rest positions, the images of the robot are rotated to exactly the specified input angle by using the following algorithm.
• Create a new empty bitmap to hold rotated image
• Make a graphics object from the empty bitmap
• Move rotation point (width/2 and height/2) to center of image
• Rotate the image to specific angle.
• Move image back since it was moved to the center
• Draw passed in image onto graphics object

4.6.5.2.3 Wrist position (B)
Refer to Figure 42 Label B. Figure 50 illustrate the reference point for measuring the angle that the wrist has rotated relative to the wrist.

![Figure 50 The reference point for measuring the angle that the wrist has rotated.](image)

To synchronize it with the real Cartesian plane the wrist input angle is changed to \( 180^\circ + \theta \).

- **Wrist gauge needle**
The wrist gauge needle is implemented the same as the speed gauge needle in section 4.6.2.2.1.

- **Setting wrist threshold**
Setting the wrist threshold is based on the position of the cursor relative to the wrist gauge. The **wrist gauge needle** will only follow the cursor if the user has clicked the gauge. To do this the following “algorithm” is used.

- If the wrist gauge is clicked
- Get the \((x, y)\) of the cursor relative to the gauge.
- Since the wrist input values are in angles, using the obtained \((x, y)\) \(\tan\left(\frac{y}{x}\right)\) is used to obtain \(\theta\)
- \(0^\circ \leq |\theta| \leq 90^\circ\), therefore it has to be made \(0^\circ \leq |\theta| \leq 360^\circ\), based on the \(xy\)–quadrant that the point \((x, y)\) is located.
- If \(x = 0\) and \(y < 0\) then wrist threshold = \(90^\circ\)
- Else if \(x = 0\) and \(y > 0\) then wrist threshold = \(270^\circ\)
- Else if \(x > 0\) and \(y = 0\) then wrist threshold = \(180^\circ\)
- Else if \(x < 0\) and \(y = 0\) then wrist threshold = \(0^\circ\)
- Else if \((x, y)\) is in the 1st quadrant then wrist threshold = \(180^\circ - \theta\)
- Else if \((x, y)\) is in the 2nd quadrant then wrist threshold = \(180^\circ + \theta\)
- Else if \((x, y)\) is in the 3rd quadrant then wrist threshold = \(360 - \theta\)
Else If \((x, y)\) is in the 4th quadrant then wrist threshold = \(\theta\)

4.6.5.2.4 Gripping force (C)

- **Gauge**
  Refer to Figure 42 Label C. The same procedure used for the other gauges is followed. The input values are from \(0 - 100\), the input has to be changed to an angle that corresponds to the \(xy\) - Cartesian plane. If the input of the force is \(\text{force}_{\text{value}}\) then the force angle is calculated by:
  
  - \(\text{radian}_{\text{increments}} = (18 \times \pi / 180)\), this is the deference in radians between the points \((0, 10, ..., 100)\) when they are converted to angles.
  
  - Convert \(\text{radian}_{\text{increments}} \times (10 - \frac{\text{force}_{\text{value}}}{10})\) from radians to degrees and let that be signed to \(gle_1\).
  
  - Then to get the force angle we have that \(\text{force}_{\text{angle}} = 360 - \angle_1\)

- **Gauge needle**
  The same procedure that was used to draw the speed and wrist gauge needle is used, with the angle \(\text{aDoE}\) being the \(\text{force}_{\text{degree}}\) in section 4.6.2.2.1

4.6.6 Robot setup tab

Figure 51 shows the robot setup tab which shows the states of the devices on the robot and also sets the values of those devices.

![Figure 51 Robot Setup Tab](image)

**Figure 51 Robot Setup Tab**

4.6.6.1 **Features**

Refer to Figure 51. The carbon dioxide concentration (F) is measured on the surrounding of the robot. This was explained in section 4.6.4.2.1. In this case the user sees the different concentration of \(CO_2\) with different colors. The colors are assigned as Green for \(\leq 23\%\), Blue for \(\leq 46\%\), Yellow for \(\leq 69\%\) and Red for \(\leq 100\%). The color serves as indicators to the users on the percentage of \(CO_2\) in the environment surrounding the robot with respect to the highest recorded.
The **MIC** (G) depicts the volume of the microphone on the robot and the **Speaker** (A) depicts the volume of the speakers on the robot. To set the mic volume the user must click on the mic control and use the **menu item** (B). This is the same for setting the speakers volume.

The lights sliders (C) are used to set the brightness of the four lights on the robot. This is set with respect to the percentage of brightness that is required. The **on/off buttons** physically switch on or off the devices that they label. The buttons are red to show that the device is off and lime when the device is on. If user places the cursor of the device button the color will change to the opposite color.

### 4.6.7 Connection settings tab

Figure 52 shows the connection settings tab used to manually set the ip addresses of the cameras.

![Figure 52 Connection settings Tab](image)

#### 4.6.7.1 Features

Since the network changes all the time, this feature allows the user to change the ip address of the cameras. The user can also change the stream number that the camera is connected to on the Bosch box. The system keeps track of all the ip addresses that the user had used so for future reuse.
5 Evaluation and Results

5.1 Introduction
This chapter will discuss the evaluation techniques that were used to test if the system was successful or not. This is to test if the design principles and user centered design did contribute in making the system usable and efficient.

5.2 System Evaluation

5.2.1 Bad design characteristics
In testing the system, the characteristics of a bad design has to be taken into consideration and checked if the system does not exhibit those characteristics. The characteristics of bad design are rigidity, fragility and immobility.

The GUI only uses one library, which is an SDK required to read video stream from the Bosch box. Since the user would only need to use this system on the rescue robot, this extra library is considered essential and cannot contribute to the immobility of the system. This means that the GUI does not depend on other software and this makes it mobile.

If the robot control server needs to be changed, what needs to change on the GUI is the communication protocol. No other parts of the GUI are connected to the robot control server. And if the robot vision needs to change no edits need to be done on the GUI because the GUI receives image data directly from memory and there are no rigid connections with the robot vision. Therefore the GUI is neither rigid nor fragile.

5.2.2 Accuracy
The purpose of this testing is to prove that the code performs in accordance with the design specifications, and also to prove that it does not fail when subjected to undefined inputs. This also involves testing whether the system provides correct outputs and functionality.

The GUI has two places it accepts keyboard input from the users, which is on the manipulator tab to set the gripping speed and the other option is when the user is setting the camera URLs and Stream numbers on the Connection settings tab. For the gripping speed; the GUI only accepts real numbers which are less that 20 and will notify the user if the input is invalid by making a warning sound.

The URL Combo selection which allows any type of string will only be saved when the user has entered a valid IP address and a stream number that is a valid positive integer. The user is notified about the invalid input they have entered. Therefore the system only takes in well-defined keyboard input from the user. The output that the GUI displays for the users is from the robot, if the values of the robot have known limits the values are first verified before being displayed on the GUI to the user.

To test that the server is receiving the correct data and sending the correct acquired data, the server prints out all the activities that it is getting. For example if the user clicks on to switch off the MIC, the server must receive “set 5”. This test was done on all components and the server did receive the correct data.

To test that the GUI was receiving correct data the GUI was ran for some time and allowed the values in the server to change. For example, with battery life, it is simulated that the battery life decreases by one every time the GUI requests it and it initially starts at 100% when the server starts. Say currently the battery level shows 60% on the GUI, then this means that if the GUI is closed and opened again it should show 60%, not 100%. This test was successfully done on all the GUI components.
5.2.3 Validation Testing
The validation testing goals is to validate and be confident about the system, that it fulfills the requirements given by the users. This test answers the question of whether the system is right and whether the system fits the required purpose. Validation of this system started in the early stages of system development. The system that has been developed was designed by operators of the rescue robot. This means that they designed the system that they require and validated the system more during the design stages on the three iterations that were done. To get extra validation that the system is right, the users who designed the system where provided with the system to test whether the required functionalities are present on the system. All the requirements that the users said they wanted the system to provide were successfully implemented. The requirements are discussed in section 3.5.

5.2.4 Stability
If the user is using the GUI and the robot control server dies, the GUI will try to connect to the server without the user noticing anything for at least 10 seconds, after which the user will get a message that the robot control server is down and is asked whether to continue or suspend execution. Most system errors that happen are suppressed and dealt with without the user knowing unless the knowledge of the user is essential.

To test stability of the GUI; the server was closed while the GUI was still running, the GUI did not crash. If any data was sent when the server died the data will be resent to the server without requesting any assistance from the user.

5.2.5 Stress testing
The GUI in its self is flooded with a lot of sensor payload that it needs to deal with and if any stress is placed on the GUI by giving it abnormal volume of commands, the GUI slows down and may start to not respond to any other command.

To test how the GUI deals with stress the GUI was given numerous commands to switch on and off devices. But since the previous command switches had not been sent yet to the server, the program waited for quite a while for the same port to send the data and this resulted in the GUI not responding to any user command.

5.2.6 Restore testing
When the GUI is opened it recovers from the last state that it was closed at if it is assumed that the state of the robot did not change. For example testing this on the wrist threshold; the wrist threshold was set to a certain value and then the GUI was closed and opened again; when the GUI restarted the same wrist threshold was being used on the wrist gauge. This test was successfully carried out on all controls of the GUI that require restoration.

5.3 User Evaluation
5.3.1 Methodology
To test that the system is usable users were required to do the same user testing in section 3.3.3. But in this case user testing was done on the users who participated in the design of the system (design users), users who already did the user testing before but not design users (old users) and completely new user who had not seen and used the system before (new users). The reason to test the design users is to determine whether the users are familiar with the system they designed and can use the system design. This is to test how well the system implementation maps with the paper designs that the users did in the 1st iteration of the system.

Testing the old users will be to determine whether the users do remember the task they did in 1st user testing and how well their usability skills of the system has improved by comparing the mistakes they did then with the mistakes they do on this second user test. Both the old and design users have the chance to see how well their feedback is incorporated in the final design.
The third user group is completely new users of the system. They are meant to test whether the system that has been influenced by the old and design users is usable by other users and if the mistakes that the old users did are still being done by the new users.

After the users completed tasks, they were required to fill in a questionnaire that asks them about the usability of the system and the design of the system. The results of the questionnaire will determine whether the design principles where successfully implemented and whether the system is usable for all users.

5.3.2 Testing
The old users and the design users have not seen the system in two months. There were four design users, four old users and six new users. The reason for not having all the seven old users is that all the users did not have time with the engineering projects that they are currently doing, but this does not have a huge effect on the results. The population of the users is good enough to determine whether the system is successful or not.

The testing was not done on the real robot because it had not been assembled yet. The system was only connected to the devices that were available; so some values that the users saw on the system where simulated by the robot control server.

5.3.2.1 New users
Refer to section 3.3.3.2.1 for the task definition. Figure 53 shows the result from the user testing with the new users. The figure shows the percentage of users who accomplished each task correctly.

<table>
<thead>
<tr>
<th>Task number</th>
<th>Percentage of users correct (out of 6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 0</td>
<td>100</td>
</tr>
<tr>
<td>Task 1</td>
<td>100</td>
</tr>
<tr>
<td>Task 2</td>
<td>100</td>
</tr>
<tr>
<td>Task 3</td>
<td>100</td>
</tr>
<tr>
<td>Task 4</td>
<td>100</td>
</tr>
<tr>
<td>Task 5</td>
<td>100</td>
</tr>
<tr>
<td>Task 6</td>
<td>100</td>
</tr>
<tr>
<td>Task 7</td>
<td>50</td>
</tr>
<tr>
<td>Task 8</td>
<td>100</td>
</tr>
<tr>
<td>Task 9</td>
<td>83</td>
</tr>
<tr>
<td>Task 10</td>
<td>100</td>
</tr>
<tr>
<td>Task 11</td>
<td>100</td>
</tr>
<tr>
<td>Task 12</td>
<td>100</td>
</tr>
<tr>
<td>Task 13</td>
<td>100</td>
</tr>
</tbody>
</table>

Figure 53 Percentage of new users who were able to accomplish a certain task

- **Task 7: Switch off the front camera.** The user was expected to either right click on the front camera view on the camera view window or switch off the camera on the on/off button in the robot setup tab. One user said thought he had to click on the camera and then click emergency stop. The 2nd user disconnected all cameras by clicking the disconnect all button on the camera view window. And the 3rd user said she has no idea on how to do it. But when all the users were told that they had to right click the front camera, and then select to switch it off there, the users pointed that if they had known about the right click possibilities then they would probably have known how to switch it off.
Task 9: Set the gripping force threshold to 5. Only one user was unable to do this task. To do this task the user was expected to go to the manipulator tab and use the wrist gauge to set threshold by dragging the wrist gauge needle to the desired position. The user said she expected to see the options on the object mode combo box selection.

Comments:
- One user said that the on and off buttons which have the camera options are not visible enough.
- To choose the automatic gripping one user was expecting to see this on the gripper type combo selection box.
- The users said if they got used to the system they would have done better.

5.3.2.2 Old users and design users
All the users where able to accomplish all the 13 specified tasks.

Comments:
- One user said that the roll input maybe should be $-180^\circ$ to $180^\circ$, because it gives a better idea than $0^\circ - 360^\circ$.
- Most users said that the play button used to start the timer on the camera view window was a bit vague since it might be to start playing previously recorded media.

5.3.2.3 Comparison with previous user testing
This section will compare the previous user testing with the old users and the user testing with the new users. This will help determine if the users are still making the same mistakes and to determine whether usability has increased. This comparison is done on all tasks that the old users had problems with before.

Task 1: You are driving the robot and you would like to change the main camera view form live camera to front camera. No new user had a problem with this, but in the previous testing 43% (3/7) users were not able to do this task. In the previous user testing users where clicking inside the video stream, this functionality was added on the system.

Task 2: The robot has entered a very hazardous area and you would like to immediately halt the movement of the robot. No user had a problem with this, but in the previous testing 29% (2/7) users were not able to do this task. The user had suggested that key Esc must also be added as an emergency stop and this functionality is also implemented. The shape of the emergency stop also changed as required by the user before.

Task 4: You would like to save the route that the robot has taken so far. The users had problems with the small name tabs before, but now all users felt they are clear enough to see.

Task 5: You would like to save events that have occurred in the robot’s surrounding. In the previous testing 57% (4/7) users had problems with the naming of the tab and were unable to accomplish the task. In this testing no user had problems with the naming of the tab which changed to environmental info log from warnings.

Task 7: Switch off front camera. In the previous testing 100% (7/7) of users did know how to accomplish this task. The users suggested right click capabilities, which were successfully implemented. In this session 21% (3/14) users did not know to do this but when told what to do they said they would remember.
**Task 8: Switch on the fan.** In the previous testing 100% (7/7) of user of users did know how to accomplish this task. Users had problems with `sensor` as the name of the buttons. The names of the buttons have been changed to `on/off buttons` and the buttons are not tabbed as requested by users in previous testing. In this test all the users where able to accomplish this task.

**Task 9: Set the gripping force threshold to 5.** In previous testing 29% (2/7) of user of users did know how to accomplish this task. All Users had problems with the up and down setting arrow being small. One user suggested the gauge needle should be draggable and this was successfully implemented. In this test all the users where able to accomplish this task.

### 5.3.3 Questionnaire Results

After the users did the user testing they were required to rate from 1 to 5 the usability and the quality of design of the system. The following question were asked:

**Usability**
- **Learnability:** How easy was it for you to accomplish basic task at first glance of the system?
- **Efficiency:** Once you got used to the system, how easy was it to use the system?
- **Memorability:** If you go for period of time how easy will it be for you to remember to use the system?
- **Satisfaction:** How pleasant is it for you to use the system?

**Design**
- **Visibility and visual weight:** How do you rate data visibility of the design?
- **Environmental awareness:** How aware are you of the environment around the robot?
- **Robot awareness:** How aware are you of the robot status and information?
- **Affordance:** How well do components function the way you expect them to?
- **Mapping:** How well are components mapped to the real world/ real device?
- **Feedback:** How well does the system give you feedback?
- **Fitts’ law:** How easy is it to access the desired components?

Figure 54 shows the results obtained from the questionnaire. The bar graph shows a comparison of usability and system design quality ratings between the three user groups.
Figure 54 Comparison of usability and system design quality ratings between the 3 types of users.

New users
The new users rated memorability highest with 100%. This means that if the new users where to use the system after 6 months or so they would not forget the functions of the system at all. The average of the users rating is 85.8% which is close to system feedback and Fitts’ law. The users rated their satisfaction of the system the least with 80%.

Old users
The old users rated environmental awareness the highest with 95%. This means the users quite understand the environment the robot is situated in when they are using the system. The average of the users rating is 80% which close to system robot awareness and Fitts’ law. The users rated their satisfaction of the system the least with 65%. The reason for such a low rating might be that the users were expecting more than they had seen on their previous testing.

Design users
The design users rated robot awareness, learnability, efficiency and memorability the highest with 90%. This means that the users’ usability rating is very high since 3 out of 4 usability requirements are in the highest rating. The average of the users rating is 85.9% which is close to everything else except the highest ratings and satisfaction. The users rated their satisfaction of the system the least with 80%.

Figure 55 illustrates the results obtained from the questionnaire. The bar graph shows the total usability and system design quality rating for all users.
Figure 55 Total usability and system design rating for all users.

The total users rated memorability highest with 94.2%. The users rated the mapping of the system design the least with 68.5%. This was also affected by the fact the only 78.5% users answered the question. Therefore a valid value would be 87.7% that takes for consideration the unanswered questions. Therefore the lowest would be the satisfaction of the users with 74.2%. This was mainly contributed by the low satisfaction of the old users. The average of the users rating is 84.7%.

5.3.4 Conclusion

From the findings in the user testing, it can be seen that there was a very big improvement from the previous user testing and users where able to do the tasks that where provided. And this was not only biased to the users who have seen the system before, but even the users who were new to the system showed that the system was indeed user centered and user-friendly and the users who had used the system before were able to accomplish all tasks.

The questionnaire findings showed 85.7% of system usability and 84.1% effective design principles. In order to have a definite proof of user friendliness, usability and effective design principles, the system can be tested with more users. During the test the same user testing procedures can be done. The testing can also be done in an environment that depicts a real disastrous area with the information coming from the real robot.
6 Conclusion and Future work

6.1 Conclusion
The aim of the project was to develop a user-friendly graphical user interface using effective Human robot interaction through iterative design. The interface had to be intuitive and did not subject the user to sensory overload. Human robot interaction had to take a minimalistic approach and only display the selected video streams and what is core to operating the robot. The user interface had to only bring in other data if requested or if required e.g. motor overload. Intelligence in the system was needed that is to think for the user to simplify the operation of the robot.

In order to fulfill the aim of the project, a user centered design approach was adopted that involved users from the first stages of the design until the final design was obtained. The users did different tasks on the system and based on the feedback from the tasks, improvement on the system was made. The system development that was adopted for this system was the iterative and incremental approach.

A usable and user centered system was successfully implemented. This was obtained from the last user evaluations that were done. The questionnaire findings in section 5.3.3 showed 85.7 % of system usability and 84.1% effective design principles. These results obtained show that involving users in the design stages increases the probability of obtaining a usable system since the users would have contributed in the design and the system output would be what they expected.

The final system generally received very positive feedback from the users as they were comparing the system to the available robot user interfaces. The other reason for very good system acceptance was that all the functionalities and more that the users had requested were successfully implemented.

6.2 Future work

The system had improved upon usability between different iterations; however it did not reach the highpoint of usability. This was underlined during the system evaluation discussed in section 5.2.6 when the system was presented with stressful actions or data and also the results from the questionnaire did not give 100% of usability from all the users. A good place to start with future research could therefore be in creating an even more usable system that has high performance and can deal with different stressful situations and has a 100% of usability from users.

Due to time constraints of the project and unavailability of an assembled complete rescue robot, user testing and system testing could not be done on the real robot and in a real simulated disastrous area. Therefore basing the usability of the GUI only on this user testing is not sufficient to declare complete usability of the system. Therefore the user testing could be done on a real robot with real information and in a simulated disastrous area. Performing user testing in simulated disastrous area helps in determining how quickly users react to real events, how long it takes the user to perform tasks and how quickly the commands are reaching the robot and taking effect.

The information that is obtained from the depth camera is very large in size and difficult to process in real-time without any help with libraries. Another interesting future work would be to optimize the retrieving and processing of the depth camera information in real-time. Currently the one implemented uses data that is read from a text file and shown in a 2D plane. A Graphical Processing Unit would be a very good platform to start the optimization. The depth data would be very useful if it was represented in full 3-D space, which is possible since the depth camera can give as input the \((x, y, z)\) points. Having the depth information helps the operator in path planning for reaching a victim or desired object because depth information gives the user accurate reading how far away from the robot objects are.
The image of the robot that was drawn to show the position of the arm can also be drawn on 3D space and can also include the movement of the head and the flippers. This would give the user more information on the robot and the users can have different viewpoints of the robot.

Giving the user as much information as possible but also preventing information overload is very important on a rescue robot. Having a usable and efficient user interface for a rescue robot helps in increasing situation awareness of the user and therefore assisting the user in finding victims trapped in those disastrous areas and thus saving lives. An unusable interface increases human errors which would lead to operators harming victims or even killing the victims instead of saving them. Therefore for these reason future research is justified.
7 References


Appendix

A.1 APPENDIX A HARDWARE

A.1.1 UCT rescue robot overview

Figure A.1.1 shows the rescue robot that has been developed by the MSc students with all the components used on the GUI shown and explained when it is in driving position.

Figure A.1.1 Rescue robot in driving position
[Source: Duncan McMillan Robotics and Agents Lab, UCT]

Figure A.1.2 shows the rescue robot that has been developed by the MSc students with all the components used on the GUI shown and explained when it is in user position.

Figure A.1.2 Rescue Robot in User Position
[Source: Duncan McMillan Robotics and Agents Lab, UCT]

Figure A.1.3 shows the sensor payload of rescue with all the components used on the GUI shown.
A.1.2 Hardware connection to robot control server

Figure A.1.2 shows how the different cameras and devices are connected to the user PC and how they communicate.

![Sensor payload of rescue robot](source: Duncan McMillan Robotics and Agents Lab, UCT)

Figure A.1.4 A connection overview of video input, audio input and audio output and Overview of how devices that are to be accessed on the robot by the robot control system are connected to the robot control system respectively.

All cameras namely the live, front, rear and thermal camera except the depth camera are connected to the Bosch box. The Bosch box enables the extraction of DataStream from the cameras and display the stream on an Active-x container. A depth camera approximates the distances of objects from the camera. The depth camera can only detect objects within a distance of 500mm and 10 000mm. Controls for speaker and the microphone are done through a separate microchip control. Every component below the wireless network are all located on the robot. The Tibbo box is used to convert between Ethernet and serial.
A.2 APPENDIX B BOSCH VIDEO SOFTWARE DEVELOPMENT KIT

The Video SDK is a reusable software library that exports a high level, object-oriented API based on COM objects and ActiveX controls. It provides the following features.

- Network device detection.
- Concurrent network connections to multiple devices.
- Live video rendering from multiple devices including in-window pan / tilt / zoom (PTZ) control.
- Playback video rendering from multiple devices including direction, speed, and stepping control.
- Live and playback audio rendering.
- Audio streaming to capable devices.
- Direct audio and video streaming to client applications.
- Recording of live video and rendering of recorded video.
- Still image capture.
- Control of device video and audio.
- Control of relay outputs.
- Event notification from device relays and alarms.
- Device event searching including input alarms and motion alarms
- Integrated diagnostic logging.

The Video SDK consists of ten main components. Each component is an ATL COM component that is either an ActiveX Control with a user interface (UI) or a non-UI COM object. The following table depicts the Video SDK’s major components and their relationship to the client application.

<table>
<thead>
<tr>
<th>Component</th>
<th>Type</th>
<th>High-Level Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AudioReceiver</td>
<td>COM object</td>
<td>Renders audio from a network device.</td>
</tr>
<tr>
<td>AudioSource</td>
<td>COM object</td>
<td>Captures audio for streaming to a network device</td>
</tr>
<tr>
<td>Cameo</td>
<td>ActiveX Control</td>
<td>Renders a live or playback video stream from a network device</td>
</tr>
<tr>
<td>DeviceConnector</td>
<td>COM object</td>
<td>Establishes a connection to a network device</td>
</tr>
<tr>
<td>DeviceFinder</td>
<td>COM object</td>
<td>Detects network devices</td>
</tr>
<tr>
<td>DeviceProxy</td>
<td>COM object</td>
<td>Manipulates a network device</td>
</tr>
<tr>
<td>DiagnosticLog</td>
<td>COM object</td>
<td>Logs messages from client applications and the SDK itself</td>
</tr>
<tr>
<td>MediaFileReader</td>
<td>COM object</td>
<td>Renders a media file recorded with the MediaFileWriter</td>
</tr>
<tr>
<td>MediaFileWriter</td>
<td>COM object</td>
<td>Records a media file to the local file system</td>
</tr>
<tr>
<td>PlaybackController</td>
<td>COM object</td>
<td>Controls playback operations and trick play operations for playback streams</td>
</tr>
</tbody>
</table>

Table 1 Major Video SDK Components description