1-1-2011

Control of a single ground vehicle using aerial and onboard camera views

Vishal Lowalekar
Wayne State University,

CONTROL OF A SINGLE GROUND VEHICLE USING AERIAL AND ONBOARD CAMERA VIEWS

by

VISHAL LOWALEKAR

THESIS

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

2011

MAJOR: COMPUTER ENGINEERING

Approved by

____

Advisor Date

DEDICATION

This is dedicated to my mother and father for their support.

ACKNOWLEDGEMENT

I would like to thank Dr Abhilash K. Pandya for all the guidance and assistance he has provided. His guidance was helpful for me not only for my thesis but also for my career growth and development. I consider him to be a very friendly mentor who is always ready to help. I would like to thank Shawn Hunt, Sam Lee, Luke Reisner, Vimlesh Shukla, Brady King, Ankur Saraf and the rest of my in the CARES group for their help and support at different stages of my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

MOTIVATION AND BACKGROUND

A robot can be simply defined as a machine that reduces the dependency of humans to accomplish a task which requires precision and caution at times. Over the past few decades, robotic architecture has replaced numerous manual operations.

Robotic systems have become an integral part of the military, automotive and medical fields. An assortment of robotic systems has evolved on par with the development of science and technology. Packbot, iRobot 710, iRobot210 SUGV, Micro robot and Nano robot, and DaVinci System are a few of the highly developed robotic systems with distinct functionalities.

Interestingly, different techniques have been coupled with the robots to enhance their performance. Many robots are equipped with image guiding tools along with different sensors, IMU (inertial measurement units) and GPS which allow for better precision, visuals and error correcting mechanisms together with providing their own locations.

Image guiding tools such as onboard cameras could be used for the teleoperation and visual servoing of the robot. This, in turn, could be used for surveillance and combat task visual servoing, which can be defined as sending the robot to the desired location by clicking on its video feed. Teleoperation controls the robot from a distance by using its camera. A study conducted by Chen [1] on teleoperation shows the examinations of various factors that affect human performance like Field of View (FOV) of the camera and the orientation of the robot, while teleoperating the robot. The paper also discusses some of the solutions for improvising teleoperation by using the audio, haptic display and gesture input controls. Teleoperation is a useful feature and the video feed could be used for visual servoing suggesting that robots can take on multiple roles.

Many robots are available that can perform various tasks, such as combat and surveillance in different environments. Despite the varied design of the robots, many of them have similarities like using cameras and sensors. For example, the Foster-Miller TALON is a new military robot that is claimed to be fastest as compared with other robots available in military. It has numerous capabilities, ranging from on board automatic machine guns to camera and sensors. It is designed for various environments and can work in snow, sand or water.

For complex tasks like performing surveillance this robot has limitation which can make teleoperation difficult. For instance, it does not have visual servoing and surveillance. It relies on the EYE DRIVE [2] a recent surveillance robot with panoramic vision. Using four cameras (illuminated by a white or NIR LED) the operator has an enhanced view of what is happening around the robot. An additional fifth camera with tilt and zoom allows the operator to further investigate a situation. Thus different robots are used for different task, but multiple applications can be performed using the same robot.

Many robots have cameras, and sensors, that can be used for surveillance tasks as well as attack purposes also. Thus a single robot provides the capability for multiple operations with ease. It is not necessary to change the robot for every application.

Sufficient research could provide a typical camera enabled robot with, surveillance features and or a visual servoing-controlling algorithm for analysis or for combat purposes. Furthermore, it could enable an aerial robot with a camera to track a ground based robot.

This thesis project encompasses the development of software for controlling a single ground vehicle using an aerial camera view or the onboard camera view of the robot. Various capabilities like teleoperation and “point and go” control algorithm are implemented on the ground robot in this research. Teleoperation is controlling the robot from a distance by using its camera view and “point and go” sends the robot to a desired location using the onboard or aerial camera views.

According to Fong and Thorpe [3] teleoperation is vital for human robot communication. Teleoperation is required, so that a robot can perform well in various surrounding environments. Their study describes various teleoperation of a UAV (Unmanned Aerial Vehicle), UGV (Unmanned Ground Vehicle) and Underwater Vehicles. It also describes the various interfaces used for the teleoperation of these various vehicles. Teleoperation has achieved a level where stereoscopic displays and simulations are used for driving the robot. A study conducted by Chen [4] at US Army Research Laboratory tested the use of 3D stereoscopic display to drive the robot, providing an indirect but realistic view. In their experiment they were using 3D glasses to drive the robot rather than directly viewing on the site.

The field of telerobotics is also playing a huge role in surgeries. According to Ballantyne [5] laparoscopic surgery could be enhanced by using and modifying the existing imaging techniques to use 3D views. Ballantyne used a AESOP arm as it simulates the human hand and optimizes the ergonomics of the operating positions. A robot having the teleoperation capabilities could be used for various tasks like de-mining operations [6], cleaning a nuclear plant [7] or locating and handling hazardous material.

Teleoperation could be enhanced if some other capabilities could be added to it, such as “point and go” capabilities using an aerial camera view as well as an onboard camera view. The “point and go” control mechanism could be implemented using several different methods. For this research augmented reality and correlation tracking were used to implement various schemes of “point and go” algorithms to control the ground vehicle.

Azuma [8] presented a survey on the field of Augmented Reality. In his paper he briefly describes the various applications of Augmented Reality ranging from Military usage to the Operation Theater. He also discussed how Augmented Reality works and highlighted some of its weaknesses while proposing some solutions on the drawback.

1.1 BACKGROUND

1.1.1 Augmented Reality

Augmented Reality (AR) is a technique that helps to connect the physical world with the virtual world. There is a significant difference between Augmented Reality and Virtual Reality. Virtual Reality is a computer simulated physical world that transforms the real world to the virtual world, whereas AR places the virtual world with the physical world. According to Milgram and Zhai [9], in the human-robot team, humans are not accurate, but the robots are accurate [10]. In their paper [9] they propose an ARGOS

(Augmented Reality through Graphics Overlay on Stereovideo system that was shown to overcome operator visual limitations.

Augmented Reality provides a platform for teleoperation and achieves various benefits by overcoming various limitations. It can also be incorporated with advanced algorithms like SIFT (Scale Invariant Feature Transform) [11] and SURF (Speeded Up Robust Feature). It has also been used with object detection and 3D reconstruction, and Monte Carlo Localization [12] (a localization technique in which the position of the robot can be determined by the use of the 3D maps).

Augmented Reality is a system which detects a marker's position and orientation in the camera coordinate system. Thus it can act like a tracking sensor for various applications. Also it is reliable and operable unless and until the marker is not in the field of view or the lighting condition are drastic. The figure below provides an example of the common marker of Augmented Reality.

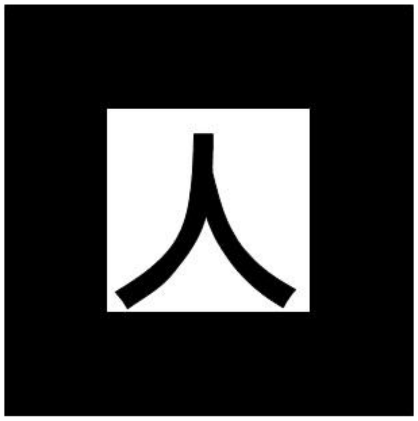


Figure 1: An example of Augmented Reality Marker.

Various 3D objects can be visualized on these markers using an Augmented Reality view. The 3D objects become part of the 3D environment and can be viewed on the screen when the marker is detected by the Augmented Reality system. The following figure shows a virtual cube on an AR marker.

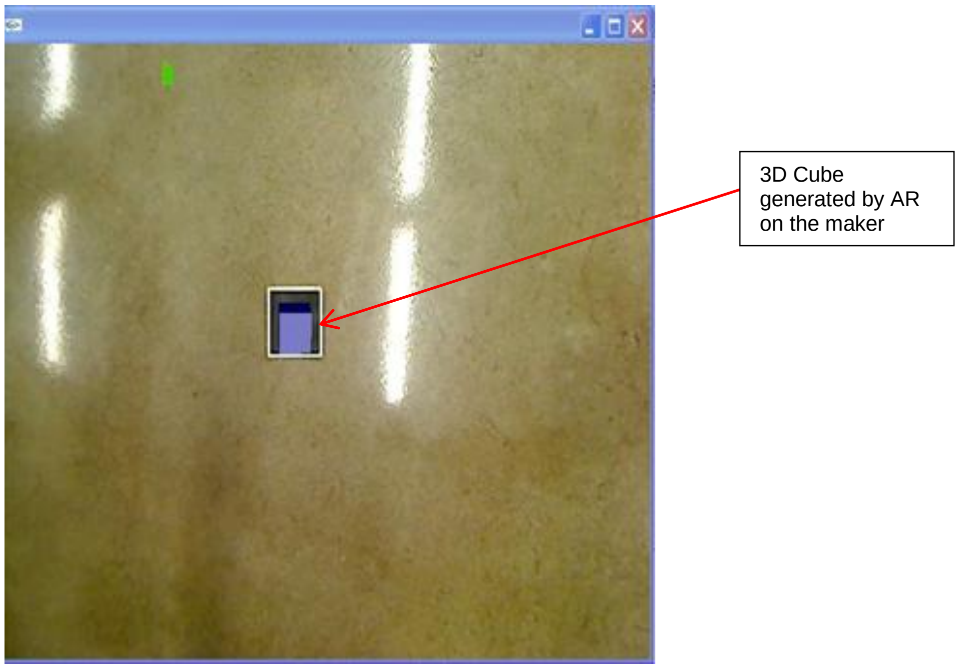


Figure 2: Augmented Reality View form the aerial robot seen by the operator.

The Augmented Reality view is shown in above figure. As soon as the object is visible it denotes that the Augmented Reality system is active and dynamic tracking data such as orientation, position and coordinates of the marker are available.

1.1.2 Correlation Tracking

Visual Servoing [13] also known as Vision Based Robot control, is used for controlling the motion of a robot using its camera. The operator is not required to be present in the environment in which the robot is working. It could be remote controlled at a distance using its camera view. The applications of visual servoing could be simply surveillance or even in combat situations.

According to Wood [24] Correlation is a process which assesses the similarity between two images also it uses the highest frequency content. Correlation tracking is an algorithm that can track an assigned goal point in a camera's view. It can be used in video image processing to track a point from frame to frame. Correlation tracking uses a digital image which provides a direct measure of the similarity between two images. Correlation tracking works in the following manner. First a point is assigned on a video frame. The value of the pixel is stored and tracking starts. The pixel value is checked on a frame-by-frame basis and the value is tracked from each frame. If there is low correlation the frame is discarded and a new frame is fetched thus keeping track of the point assigned.

For this research the correlation tracker tracks the goal point using a foveal kernel [14]. The implementation of the correlation tracker was developed by Turing Associates Inc. [14] as part of a research grant from the US Army.

Kernel and Convolution:

Kernel is a matrix of pixels, and it acts like a filter. The matrix could be of any size, it is used as an operator during the image convolution. The kernel is applied to each and every pixels of the image. The structure of the matrix will erode certain feature on the image for example if the kernel is a diagonal matrix, then the diagonal features of the image will be eroded. The weights are distributed accordingly for the filter to extract that feature.

Convolution in mathematical terms could be defined as, the mathematical operation on two functions and resulting into the new function that is the third function that could be visualized as the modified version of one of the functions. There are various types of convolutions viz. circular convolution and circular discrete convolution.

The kernels are set prior to the convolution so that it can highlight specific features of the image. The kernel works like a spatial filter each of them are made for specifically set to the set spatial frequency for convolution that is intended to be highlighted. There are various types of kernel like Gaussian blur, Laplace etc. Each has a different structure of the matrix and it will work on the image accordingly.

An example of Laplace kernel is given below:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \text{[25]} \qquad (1.1)$$

After applying this filter to the image the following results could be seen on the image. The Laplace kernel works as an edge extractor or detector which will extract the edges from the image.

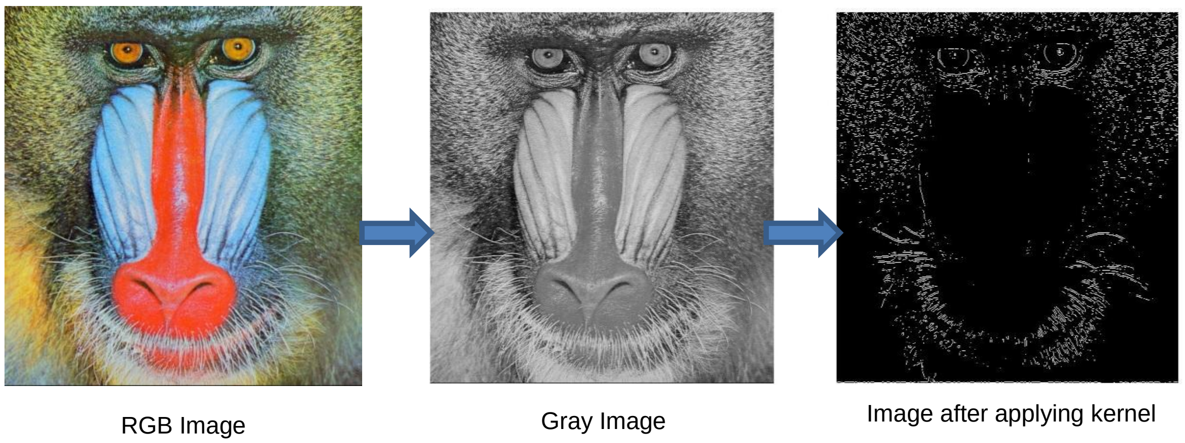


Figure 3: Implementation of Laplacian Kernel on the image.

For edge detection using Laplacian it is necessary to convert the RGB image to gray scale image or binary image and then the filter could be applied on it. The above figure shows the RGB image converted into gray scale image and the edges detected from the image after applying the Laplace kernel to it.

In the same fashion the foveal kernel works. The structure of the kernel is designed in such a manner that it is sampled densely at the center and less densely to the periphery. The following image shows the visualization of foveal kernel.

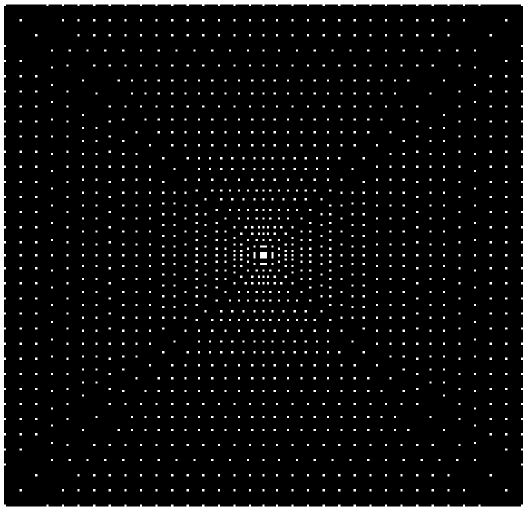


Figure 4: An example of Foveal Kernel which is used by the Correlation Tracking for tracking a point. This method is used in Aim 3 for the "point and go" operations from the ground robot's camera view. (Image Used With Permission of Shawn Hunt [14])

The correlation tracker uses the Multiple Resolution Progressive Alignment (MRPA). The expansion of the kernel is a progressive alignment. A gain is applied to the expansion factor for the incoming image or test image at time t+1 and not the reference image at time t. The center point in the kernel is tracked by the correlation tracker.

As the operator clicks on the image this kernel is applied and the point is tracked on the frame by frame basis. The following figure shows the flow chart of the Correlation Tracking algorithm starting from the frame capture up to the tracking of the points

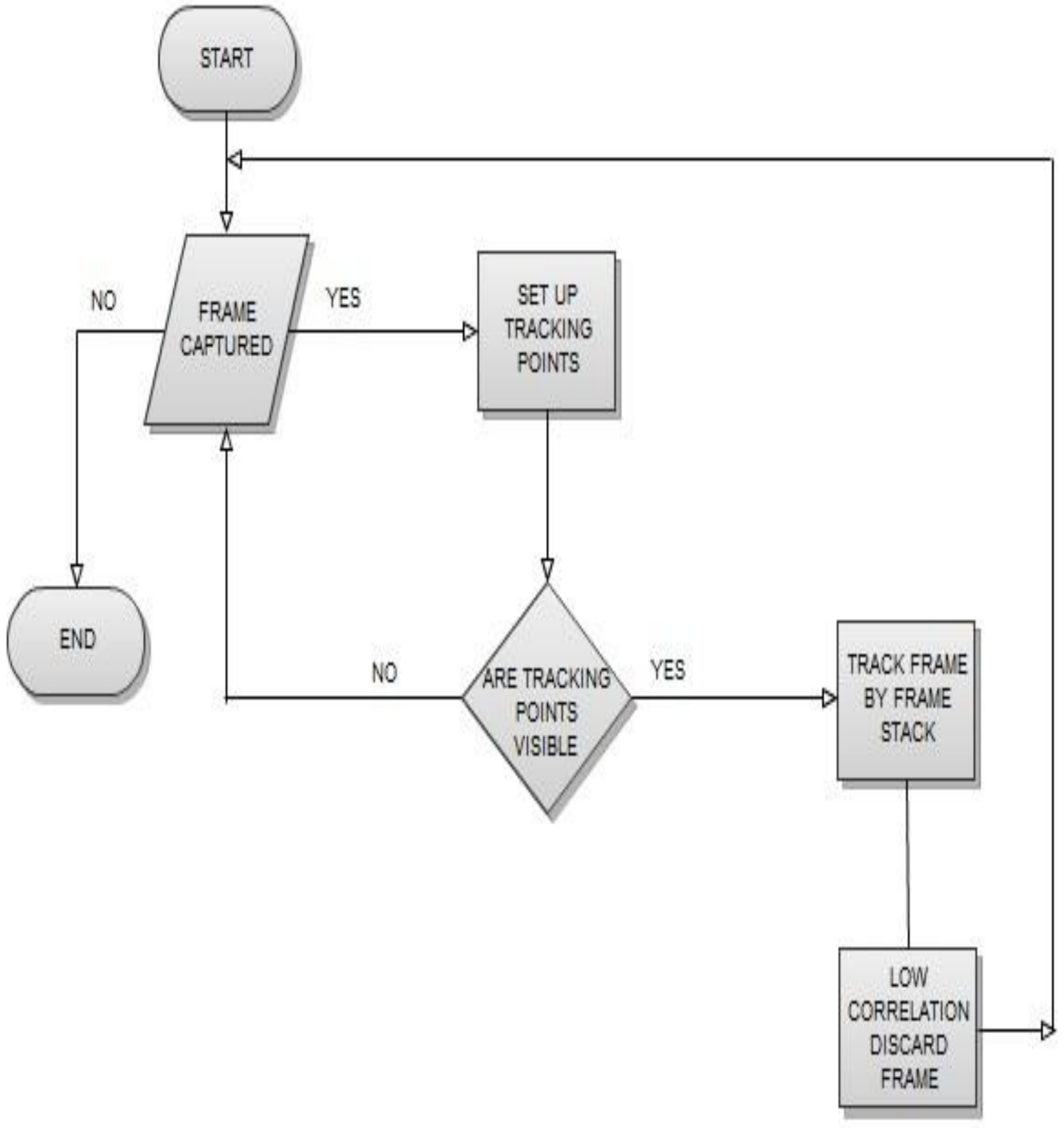

START
FRAME CAPTURED
NO
YES
SET UP TRACKING POINTS
END
NO
ARE TRACKING POINTS VISIBLE
YES
TRACK FRAME BY FRAME STACK
LOW CORRELATION DISCARD FRAME

Figure 5: Flowchart of Correlation Tracking.

The above flowchart provides a detailed description of Correlation Tracking. The algorithm starts by capturing a frame. If the frame capture is successful it proceeds to the next step and where the user sets up the tracking points. The tracking points are the pixel values (goal point) that are assigned by the user by clicking on the screen.

The tracking points are again checked to ensure that the capture is successful. If so the tracking is ready to start at that particular point/ pixel value. If it is unsuccessful tracking is aborted and the algorithm ends.

If the tracking point setup is successful it will proceed to actually tracking the point/ pixel on a frame-by-frame basis. If the tracker successfully tracks the point in all frames the tracking remains active. Occasionally, due to a change in the environment of the scene, the correlation between the frames might be too low. In this case the frames are discarded. New frames will be acquired to continue the process of the tracking.

The tracking might be aborted due to low light or scaling and rotation of the objects being tracked. However it can resume when the algorithm acquires better conditions for operation. Thus correlation tracking can be used to track objects in an environment and to match the image frame by frame for feature tracking.

1.1.3 Significance

The goal of this project is to provide the framework for tracking and movement capability of a ground robot using different camera views (e.g. on-board camera or aerial camera). Various algorithms are implemented including a point-and-go capability and movement directed from aerial camera views.

These capabilities will help the operator monitor the ground vehicle and send information about the desired location by using the aerial view of the camera or by using the onboard camera view from the robot. Additionally, the aerial robot (simulated with a robot arm holding a camera) will autonomously track the ground vehicle. This research provides a framework of tracking the robot together with sending it to the desired locations. This research can be extended for potential use in military or surveillance applications, where guiding and sending robots to the chosen locations is crucial.

1.1.4 Related Work

The research has three aims. The first aim is to track a ground robot using an aerial view. According to Morris [15] a group of UAVs (Unmanned Aerial Vehicles) can conduct cooperative tracking of ground vehicles. They proposed various approaches in which a group of UAVs perform a range of tasks such as escorting a convoy, signaling alerts on the path, tracking single ground vehicles and deciding between friendly and unknown/enemy vehicles.

Ariyur and Fregene [16] proposed an autonomous tracking of the ground vehicle using UAVs. They proposed a model and also use waypoints, which were applied on the wings of the UAV for tracking of the ground vehicle. A waypoint is a physical reference which is used for different purpose of the navigation.

The above studies use some type of sensor information for their accuracy and consistency. An alternative approach is proposed in this research using an Augmented Reality system which is capable of providing position and orientation information. An AR system offers additional capabilities when compared to sensors alone. Because AR systems are based on a marker that allows minor changes in the orientation or location to be detected.

In the first aim of the project, a ground robot will be tracked and followed by a simulated aerial robot (robot arm). So in the Tracking system, an AR approach is used in which a camera is kept on the aerial robot and a marker is kept on the ground robot. The aerial robot will autonomously adjust itself with the help of AR and keep tracking the ground robot. AR Toolkit is a software package that helps to build the Augmented Reality applications. The idea of using the AR system for tracking is significant because it can be applied to any aerial vehicle which has a camera on it. Additionally it can be applied to any aerial vehicle for autonomous tracking of the ground vehicle. The applications generated through AR software are a blend of the virtual objects in the video stream of the real world. AR assists by adding 3D objects into the real scene.

The second aim of this research is to implement “point and go” using the aerial camera view. Previously, Lee [17], proposed a “point and go” system for multiple robots using the aerial camera view. He developed “point and go” algorithm based on the camera coordinates and video coordinate. Additionally, the system uses the information of the markers placed on the robot. His system was a simple model where the aerial camera provides an aerial view and an operator has to click on the incoming video feed from the aerial camera. Therefore the robot can be moved to the desired location from a teleoperated position. A limitation of this system is that it is camera dependent, and the camera has to be static all the time. Thus, if the camera moves when the operator clicks, then the robot loses a frame of reference and will move to an incorrect location. Since the scene is changed, the camera coordinates have to be re-registered. This thesis will extend this work by adding a frame of approach was required to make the system well optimized and accurate without depending on the camera. The main feature which makes this system better and different from the previous system is: 1) It uses the world coordinate system 2) A two marker approach is implemented and 3) It is a camera independent system. Thus, even if the camera moves or the camera lost its focus, the ground vehicle will move to the desired location without depending on the camera.

The two marker approach is implemented in the system one marker is used as a reference marker, and it acts like a landmark of the system and other marker is kept on the robot which is related to that reference marker. The world coordinates and other information for the robot are set on the reference marker. The other marker will work according to the reference marker. Thus by using the two marker approach, the system is camera independent and unaffected by camera movement as long as the reference marker is visible in the AR view.

The third aim of the thesis is to control the robot from an on-board camera view. An approach that has been implemented to this “point and go” system is correlation tracking [14]. Recently, Hunt [14], proposed a “point and go” on the Packbot using the Correlation Tracking developed by Turing Associates LLC. In his research he has provided the capability of “point and go” to the Packbot, using its onboard camera, odometer and other sensors available on it. The system was based on Visual Servoing. It was using the data from the sensors and odometer of the Packbot for its location in the field. As the correlation tracking works on the camera coordinate system, it can be applied to map the world coordinates system with the camera coordinate.

If the point is defined in the world coordinate, it can be tracked through camera coordinate. When the camera moves, the scene changes but the correlation tracking will help to keep the selected point in the same coordinates where it was selected on the screen. By this a direct relation can be established between the world and camera coordinates which can increase the accuracy of the entire system.

Additionally, he was using a single point Correlation tracking for the “point and go”. The idea behind this is that when the operator clicks on the video the pixel value of the click is stored and a tracking is turned on, so even if the robot moves the clicked point will always be tracked through video feed and the robot could be sent in that direction where it is directed through the onboard camera view.

Inspired from this research an aim three was designed that is Teleoperation and “point and go” using the onboard camera view. The “point and go” capability was implemented on SRV-1 which could complete this architecture. This system is different with the previous one in thee respects 1) A “point and go” was implemented to the robot (SRV-1) which is accessible to everyone 2) Multiple point Correlation Tracking was implemented 3) No sensors or odometers are used and 4) A pure Visual Servoing approach is used.

Multiple Point correlation tracker approach was used and suggested for the estimation of time to go to the destination by using the output of the multiple point correlation tracker, by Markham [18]. According to this study, an accurate and cheaper alternative could be available using the multiple correlation Tracker for approximating the time required to move the airborne vehicle from the initial position to the destination position without using radar to estimate time.

The use of the Multiple Point Tracking in this research is for the calculation of the distance between the robot and the destination position which is clicked. These points are not only used for the calculation of the distance, but also the distance calculated from them could be used as stopping matrices for the robot. As the system is not using the sensors or odometer data, a multiple point correlation tracking is used as the alternative for the system thus making the entire system purely Visual Servoing based.

A capability of (1) using the aerial and camera views to send the robot to a desired location (2) providing autonomous tracking through the aerial robot (3) “point and go” control via an on-board camera to control the ground vehicle can make remote teleoperation very efficient and accurate. Thus an entire structure is defined through this research for controlling the ground vehicle using the various camera views.

All the aims are interrelated. The “point and go” can use the view from the aerial camera, the robot can also be guided using its on-board view and the robots can be followed by an aerial robot.

1.3 System Architecture

The entire architecture and the relationship between the three aims are shown in the following figure:

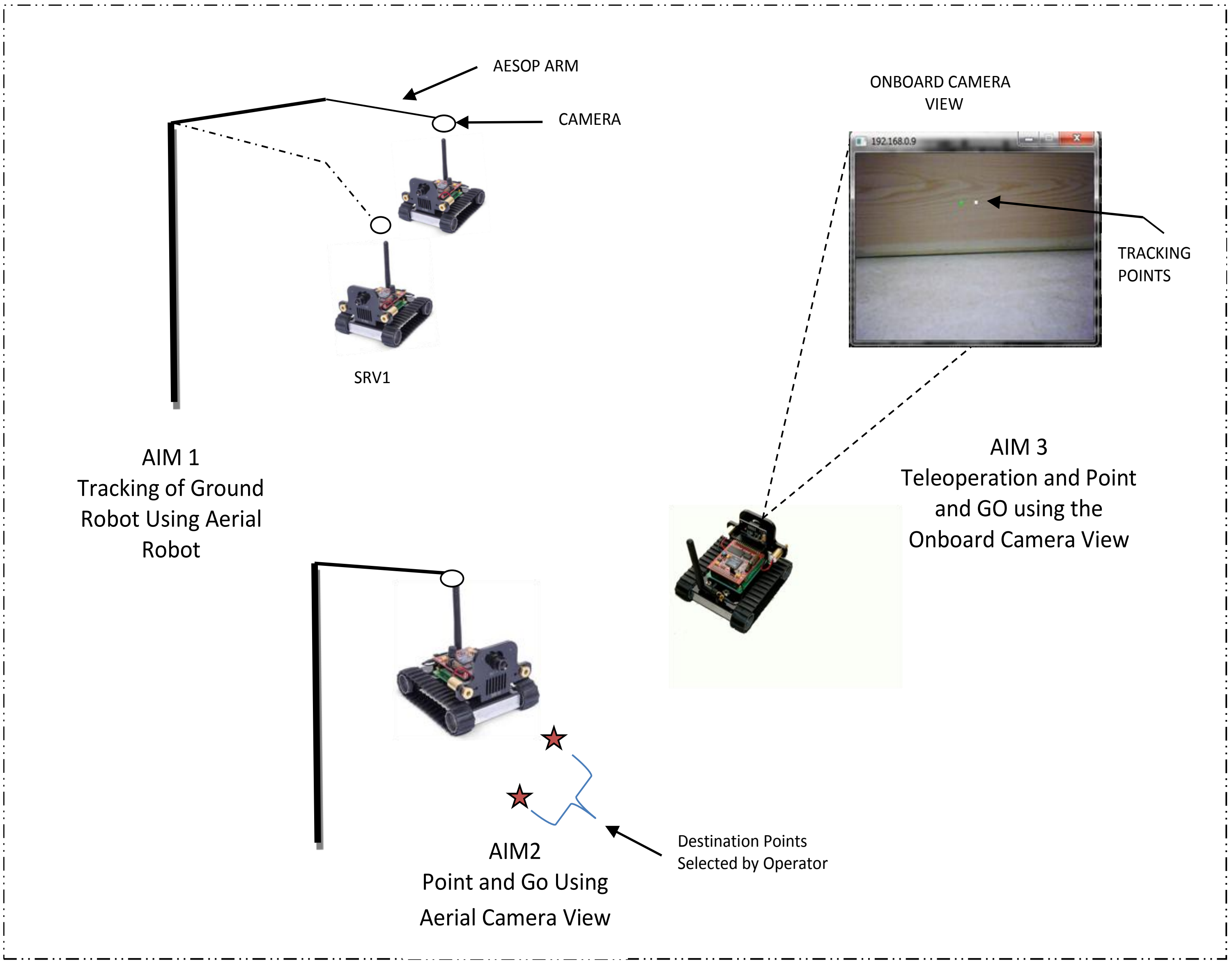

AESOP ARM
CAMERA
SRV1
ONBOARD CAMERA VIEW
192.168.0.9
TRACKING POINTS
AIM 1 Tracking of Ground Robot Using Aerial Robot
AIM 3 Teleoperation and Point and GO using the Onboard Camera View
AIM2 Point and Go Using Aerial Camera View
Destination Points Selected by Operator

Figure 6: System Architecture encompassing the three aim of Thesis, Aim1: Tracking of ground robot using the aerial robot, Aim 2: The aerial camera view being used for "point and go", Aim 3: Visual servoing by using the ground robot's onboard camera view.

.

1.2 Outline of Dissertation

The second chapter details specific Aim 1 of the research and various component of the system. The third chapter provides information about the specific Aim 2 of the research and explains how the aim is achieved and evaluated using efficiency testing. The fourth chapter presents details on specific Aim 3 and discusses about the algorithm and the testing of its component. The fifth chapter provides conclusion drawn from the research and it also discusses some future work that could enhance the system.

CHAPTER 2

TRACKING THE GROUND ROBOT USING AN AERIAL ROBOT

2.1 Introduction

This chapter discusses my first aim, which is focused on the use of aerial camera view, to track a ground robot. The idea behind this is to build a tracking system in which the ground robot can be tracked from any aerial robot that has a camera on it. The ground robot is tracked with a marker on it. In this way, wherever the robot moves on the ground, it will be autonomously tracked by the aerial robot (as long as the marker is visible).

Tracking of the ground robot is very important in combat and surveillance. This is useful for the operator to obtain updated information from the environment and the location of the ground robot using an aerial view. The operator can send the robot to different places or can receive detailed information of the specific place from the ground robot.

A novel approach has been implemented in this research for tracking the ground robot. An Augmented Reality system is used to track the robot using the information of the marker kept on the ground robot. Augmented Reality is powerful system which is implemented in various fields from medical to military fields [8]. As various systems rely on the sensor information, it can be crucial to control or navigate the system on failure of these sensors. AR system surpasses potential problem with sensor failure because as long as the marker is visible the system is active and can provide the data from the marker like orientation and position.

To demonstrate this, a setup was built in which a camera is mounted on the arm of an AESOP 3000 robot and SRV-1 is used as a ground robot. An algorithm was written, in which AESOP takes commands through a serial (RS-232) port and an Augmented Reality system is used to track the marker kept on the SRV-1.This chapter explains the entire architecture with its all components, used to achieve the Tracking system. The system level flow chart is shown in following figure.

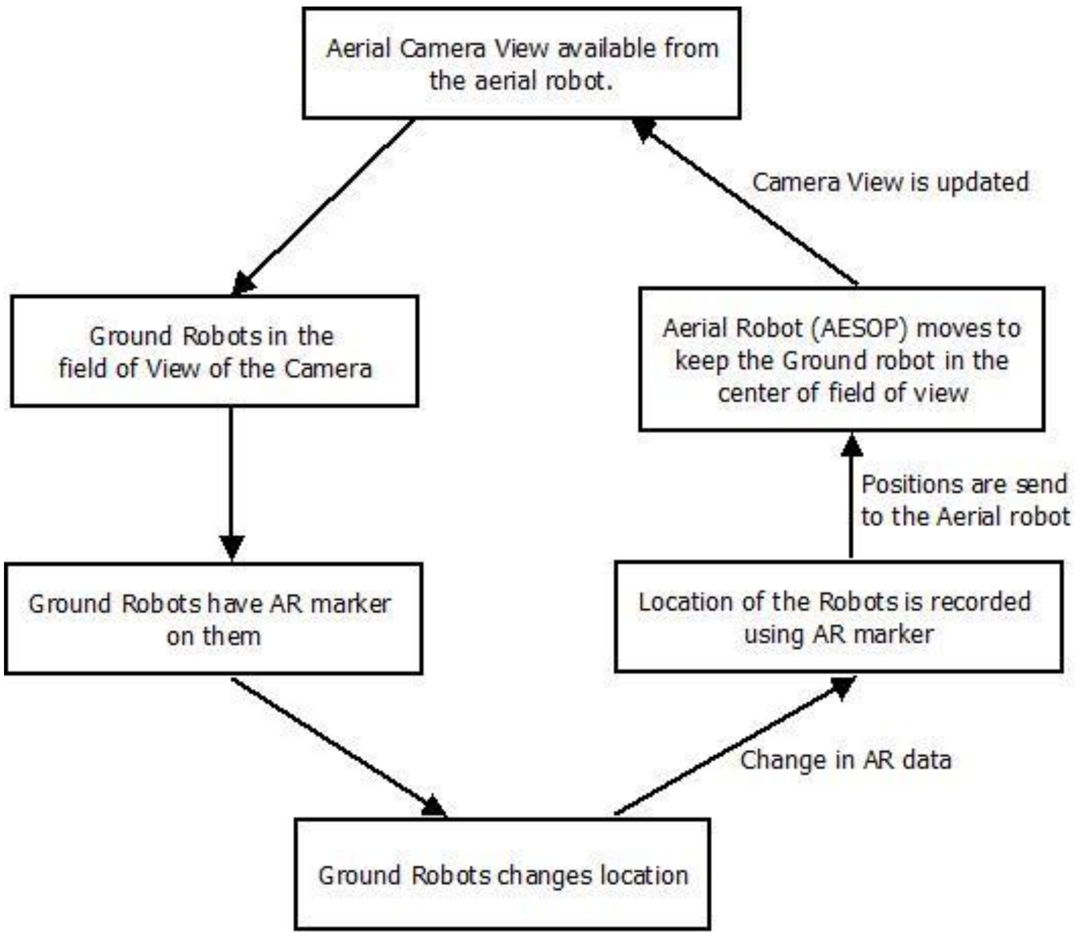

Aerial Camera View available from the aerial robot.
Camera View is updated
Ground Robots in the field of View of the Camera
Aerial Robot (AESOP) moves to keep the Ground robot in the center of field of view
Positions are send to the Aerial robot
Ground Robots have AR marker on them
Location of the Robots is recorded using AR marker
Change in AR data
Ground Robots changes location

Figure 7: System-level flow chart of Aim 1. This system allows the ground robot to always be kept in the field of view of the aerial robot.

The overview of the system is given in the above figure. It shows the system-level flowchart of the algorithm. Each block in the flowchart provides a step by step working of the system, starting from capturing the aerial camera view from the aerial robot up to moving the aerial vehicle to keep the ground robot in center of field of view.

2.2 Methods

This section discusses various hardware components used in the system. It also discusses the system setup and the sub aims, which envelopes the entire aim. Efficiency testing and results are also discussed in this section together with the solution for the limitations are also discussed.

2.2.1 Hardware Components

2.2.1.1 AESOP 3000

AESOP 3000 is a Medical robot developed by Computer Motion. AESOP stands for (Automated Endoscopy System for Optimal Processing) It can be used in minimally invasive surgery (MIS), endoscopy etc. as it imitates the functions of surgical staff that holds and moves the camera operator.

AESOP is the FDA approved robot for clinical use. It is a voice-controlled robot, previously it was used as a camera holder for the laparoscopic camera but since then has been used for operating, which required empowering it with the operating tools so that it emulates the human hand. The AESOP was then modified and also used as replacement of the Zeus robot and was also used in Da Vinci systems.

AESOP arm has a six degree of freedom (DOF). The arm could be rotated approximately around 300 degrees in X and Y direction. The end effector of the AESOP arm has the pan and tilt functionality. In this research, only 2 degrees of freedom (DOF) are used viz. shoulder and elbow joints of the robot are used to emulate the aerial robot.

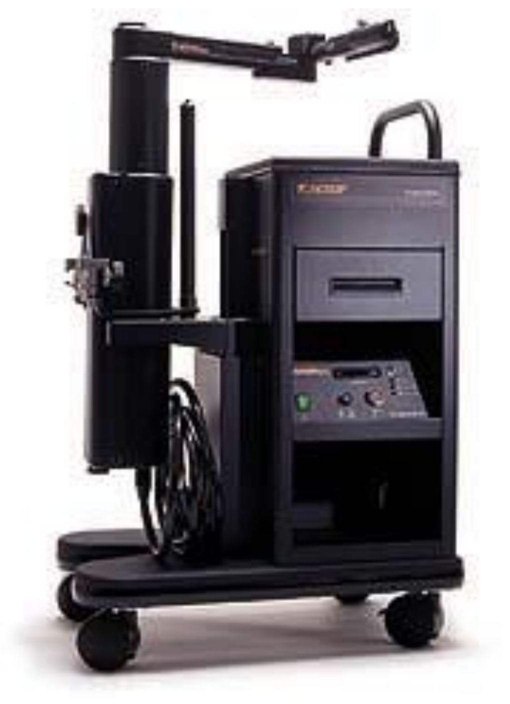


Figure 8: AESOP 3000 was holding a camera on its arm and was used as a simulation of an aerial robot.

2.2.1.2 SRV-1

SRV-1 [19] is a Wireless Mobile Surveillance Robot with video capability, which is connected through a Wi-Fi network and has a C programming interface. It has the feature of Tele-operation and can be used for SWARM robotics and surveillance.

It supports third party software like RoboRealm, Microsoft Robotic Studio and Webots. It has an onboard digital camera with various resolutions, from 160 × 128 to 1280 × 1024 pixels. It is also equipped with laser pointers that can be used as a distance sensor and WLAN 802.11b/g.

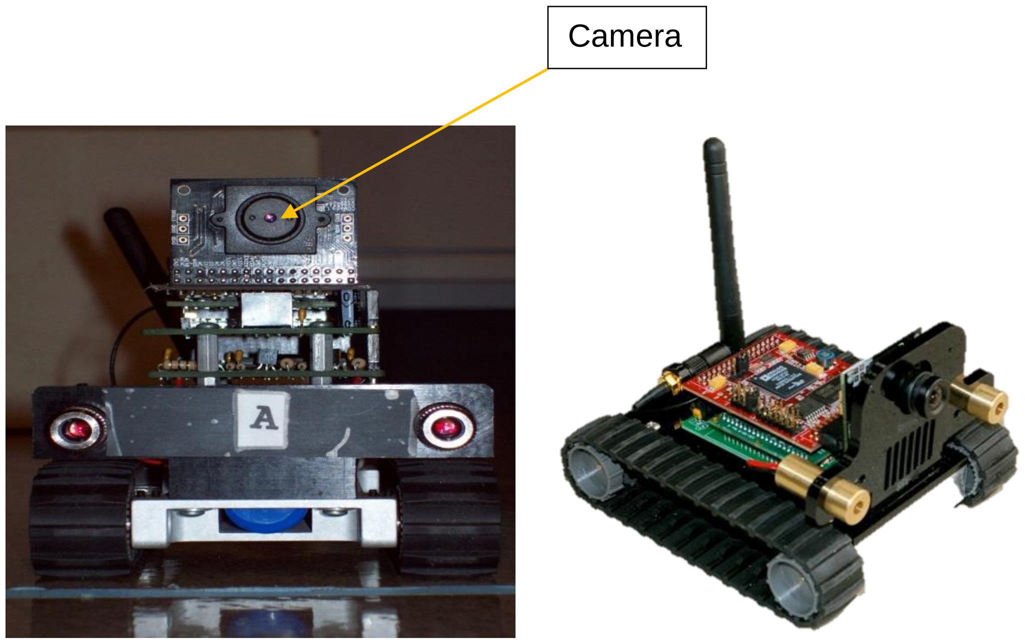


Figure 9: SRV-1 used in the Thesis as the Ground robot.

The SRV-1 has many useful features like full access to source code and schematics (General Public License). It can be used for full autonomous operations, Teleoperational mode, and has compatibility with a C interpreter, Linux 2.6.

2.2.1.3 LOGITECH QUICKCAM ORBIT AF

A Logitech Quickcam is used for the Tracking System, which is the most important component in the project. This camera is chosen for the research for many reasons: it has a Carl Zeiss lens, 2.0 megapixel sensor, and a motor for movement of the camera, which has autofocus capabilities [20].

Figure 10: LOGITECH QUICKCAM ORBIT AF camera used for aerial view and was kept on the AESOP arm [20].

The feature like good lens of the camera is required because it is resilient to the change of light conditions and can capture more data in a pixel. The higher resolution is required because the quality of the image is good compared to previous versions like VGA (Video Graphic Array) or 1.5 megapixel sensor and also it is generally available in all the cameras.

2.2.2 SYSTEM SETUP

A Logitech camera was fixed on the arm of the AESOP robot and an AR marker was put on the SRV- 1. A motion control board was used for controlling the AESOP arm.

The SRV-1 was controlled through Wi-Fi and an augmented reality system was used to track the robot using the marker on it. Also a serial interface was used to control the AESOP arm. A custom built motion control board shown in figure [11], created by Lee [17], was used in this research.

The AESOP arm allowed for two degrees of freedom (DOF) provided from the shoulder and elbow joints. The system uses two joints via the elbow joint and the shoulder joint of the AESOP arm. The shoulder joint was used to track the robot at positions the elbow joint is unable to reach.

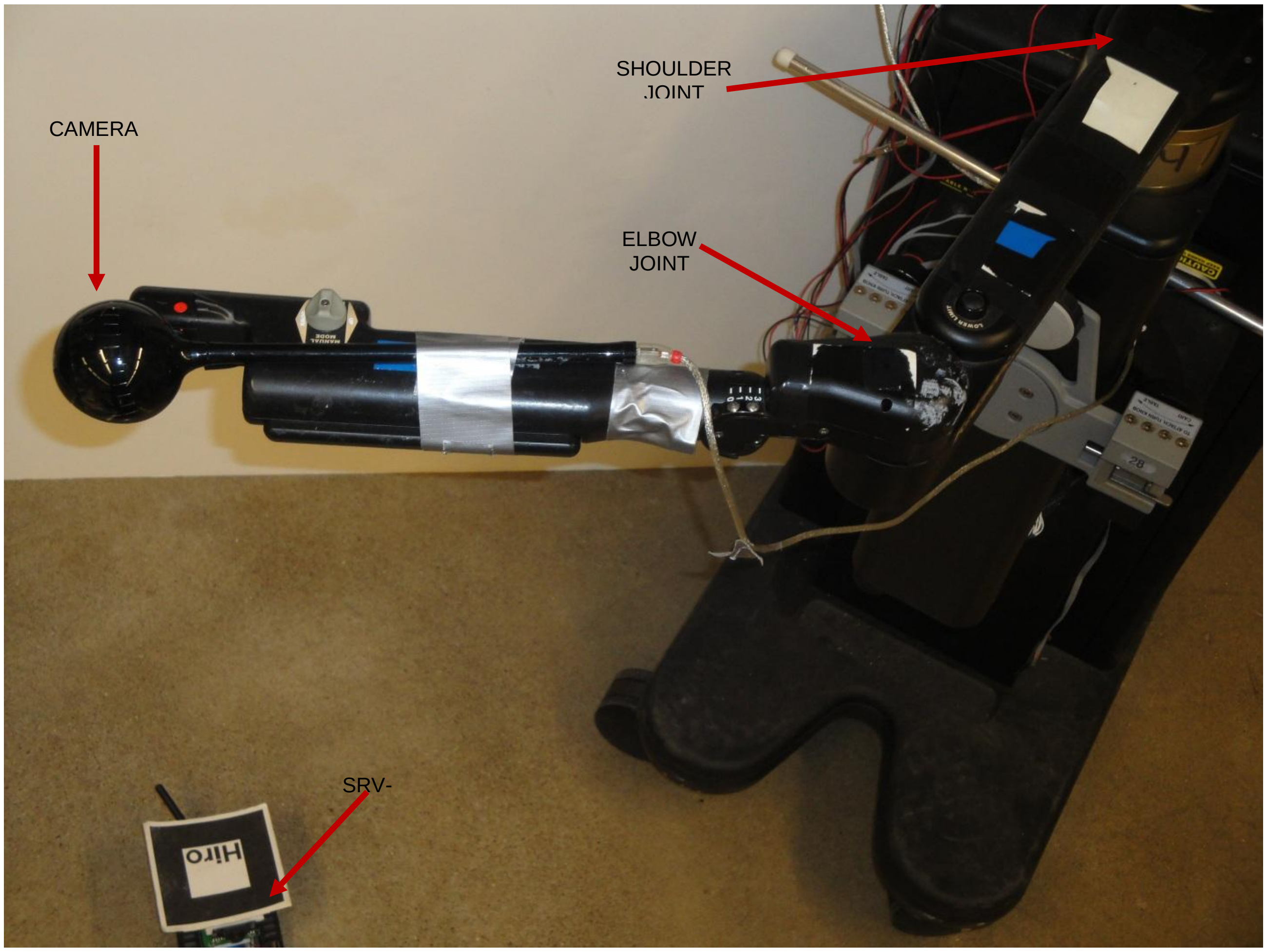


Figure 11: Here, the camera is held by the Aesop arm, and it tracks a ground robot pattern.

2.2.3 Sending Commands to AESOP via RS232

The serial program communicates with AESOP 3000 through an RS232 port which is located on its Motion Control Board. The AESOP arm can be moved by assigning encoder counts to different motors to move along the X and Y axes. The actual process is to first convert movements into the encoder count and then command the motion controller board to move the joint to that required count.

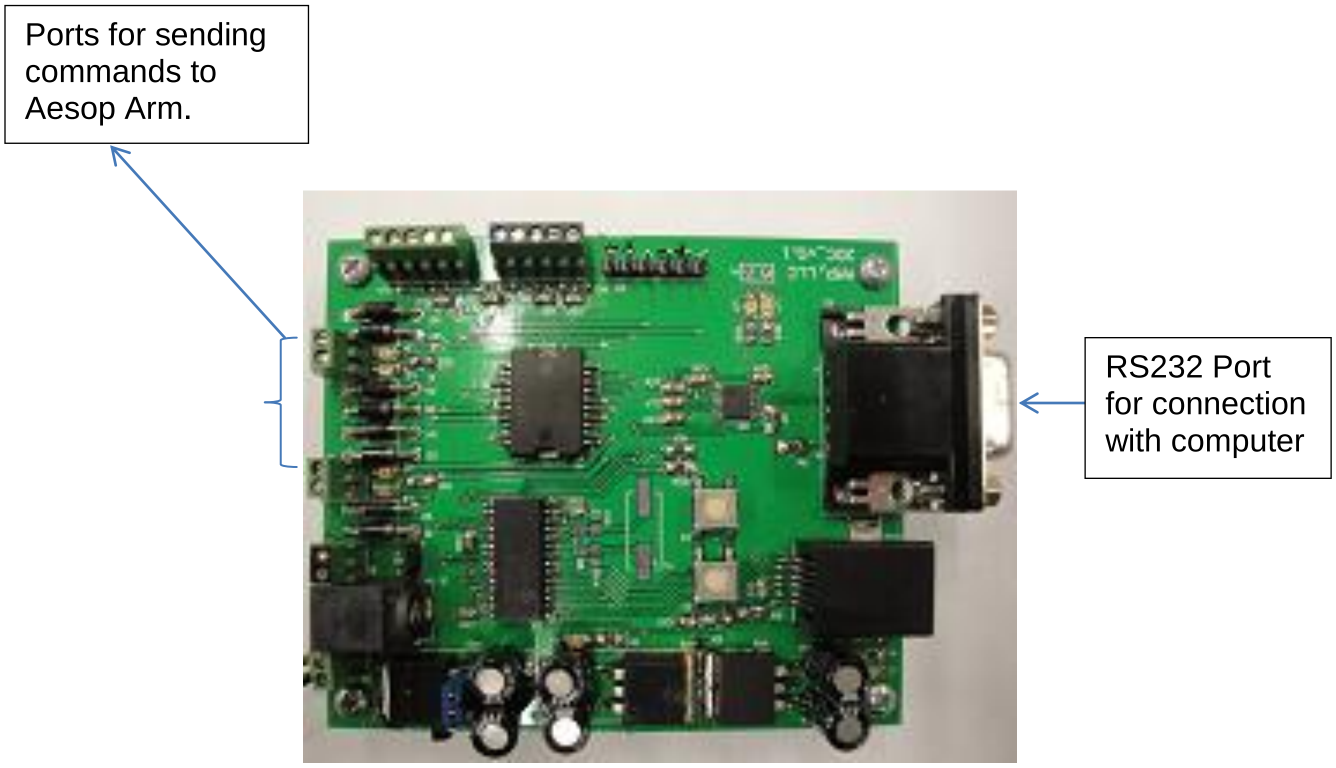

Ports for sending commands to Aesop Arm.
RS232 Port for connection with computer

Figure 12: Custom built Motion Control Board for sending command to AESOP arm [17].

The ASCII commands to control the robot movement are defined in following manner as shown in the following table

COMMANDS	EXPLAINATION
PRX	Position Relative for X axis
PRY	Position Relative for Y axis

Table 1: Commands used to move the AESOP arm

PRX and PRY: The Position Relative for X axis is for the elbow joint, and the Position Relative for the Y axis is for the shoulder joint of AESOP. So if the value of PRX is a positive count it will move in a counterclockwise direction; conversely, if PRX is negative it will move in a clockwise direction. The PRY behaves the same way. This movement is used for compensating the robot movement in a forward or backward direction in such a manner that the ground robot is in the field of view. The following figure shows the AESOP arm and its movement.

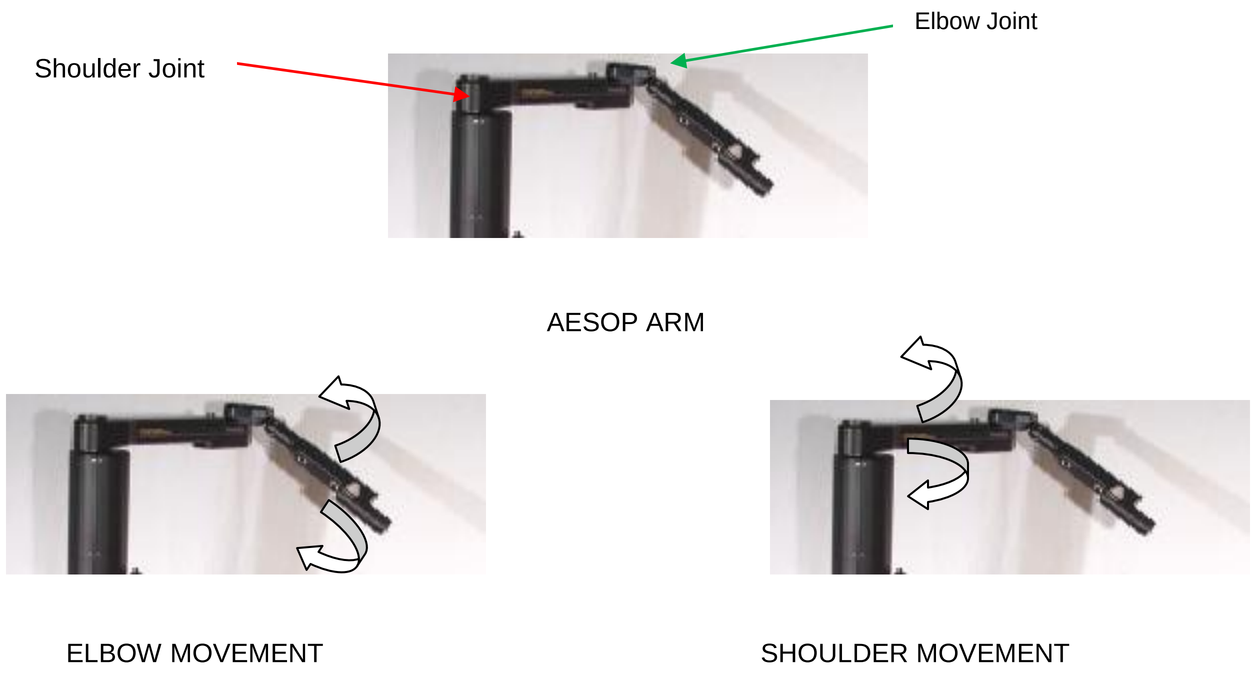
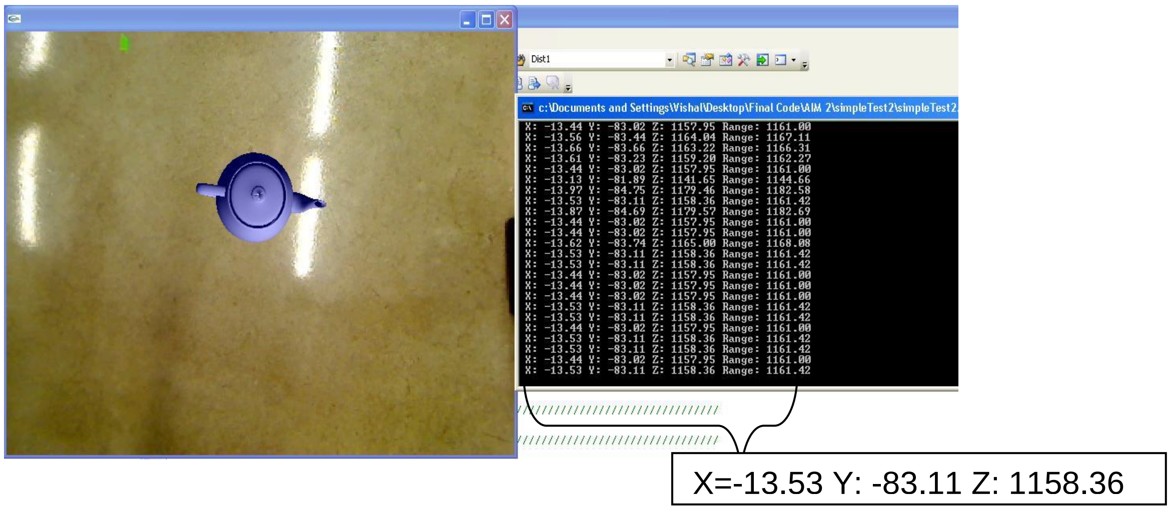


Figure 13: AESOP Arm Movement.

Once this mechanism of commands was tested, it was implemented in the algorithm of the tracking system. A function was created in the algorithm which calculates the encoder counts and sends them to the serial port. The motion controller receives the commands and moves the AESOP accordingly. The major issue faced here was how to compute the correct encoder counts in order to track the ground robot. Also, the position of the SRV-1 has to be known (via the tracking markers) as it has no sensors or odometer in it as discussed in previous section. In order to get the position of the SRV-1, AR is used. As discussed previously Augmented Reality helps to connect the virtual world to the real world. AR has a provision to find the position and orientation of the marker. AR computes the orientation and position of the marker and one can get the X, Y and Z position and range of the marker in world coordinate system.

The following figure shows the X,Y,Z coordinates and range of the marker. The range of the marker is distance between the marker and the camera in camera coordinate system.



```
c:\Documents and Settings\Vishal\Desktop\Final Code\AIM 2\simpleTest2\simpleTest2
X: -13.44 Y: -83.02 Z: 1157.95 Range: 1161.00
X: -13.56 Y: -83.44 Z: 1164.04 Range: 1167.11
X: -13.66 Y: -83.66 Z: 1163.22 Range: 1166.31
X: -13.61 Y: -83.23 Z: 1159.20 Range: 1162.27
X: -13.44 Y: -83.02 Z: 1157.95 Range: 1161.00
X: -13.13 Y: -81.89 Z: 1141.65 Range: 1144.66
X: -13.97 Y: -84.75 Z: 1179.46 Range: 1182.58
X: -13.53 Y: -83.11 Z: 1158.36 Range: 1161.42
X: -13.87 Y: -84.69 Z: 1179.57 Range: 1182.69
X: -13.44 Y: -83.02 Z: 1157.95 Range: 1161.00
X: -13.44 Y: -83.02 Z: 1157.95 Range: 1161.00
X: -13.62 Y: -83.74 Z: 1165.00 Range: 1168.08
X: -13.53 Y: -83.11 Z: 1158.36 Range: 1161.42
X: -13.53 Y: -83.11 Z: 1158.36 Range: 1161.42
X: -13.44 Y: -83.02 Z: 1157.95 Range: 1161.00
X: -13.44 Y: -83.02 Z: 1157.95 Range: 1161.00
X: -13.44 Y: -83.02 Z: 1157.95 Range: 1161.00
X: -13.53 Y: -83.11 Z: 1158.36 Range: 1161.42
X: -13.53 Y: -83.11 Z: 1158.36 Range: 1161.42
X: -13.44 Y: -83.02 Z: 1157.95 Range: 1161.00
X: -13.53 Y: -83.11 Z: 1158.36 Range: 1161.42
X: -13.53 Y: -83.11 Z: 1158.36 Range: 1161.42
X: -13.44 Y: -83.02 Z: 1157.95 Range: 1161.00
X: -13.53 Y: -83.11 Z: 1158.36 Range: 1161.42
```

X=-13.53 Y: -83.11 Z: 1158.36

Figure 14: X, Y, Z Positions obtained from the marker in the Augmented Reality System. The command prompt window shows the reading of the marker at its position.

Range of the marker is calculated using the following formula.

$$\text{Range}=\sqrt{XPos^2+YPos^2+ZPos^2} \qquad (3.1)$$

This data of the marker is used for the calculation of the encoder counts for the AESOP arm. The X position of the marker is input into the function which is built to calculate the number of encoder counts for the elbow joint of the AESOP. The calculation is performed by multiplying the X position of the marker with the value which in turn forms an encoder count. The data of the encoder counts are sent to the AESOP so it can move accordingly.

2.2.4 Inverse Kinematics of the AESOP arm

To keep the ground robot in the center of the view it is necessary to implement an inverse kinematics to the AESOP elbow/shoulder joints. The idea behind this implementation is to compensate the movement of elbow joint. This is because the SRV-1 can move greater than 180 degrees with respect to the AESOP's position and could go out of focus. To overcome this problem, inverse kinematics was implemented so that when the SRV-1 goes out of view, the shoulder joint will move accordingly. This keeps the ground robot in the visual field of the camera.

Inverse kinematics is a calculation of the angle of the different joints of the robot to keep the desired arm in a specific position and orientation. One way to implement the inverse kinematics in the algorithm is by using the Jacobian Transpose Method given by equation below,

$$\Delta\theta = \alpha J^T \vec{e} \text{ [21]} \qquad (3.2)$$

A problem with using this method was that an extensive, mathematical computation was required, before producing the data for the encoder count. This could make the algorithm slower and unnecessary complicated. To overcome this problem, a simple solution was required. This was achieved through an analysis of how the arm of AESOP moves, how much it moves, and in which directions.

The simple solution was developed by using the Y position of the marker. The Y position of the marker is input into the function which is built to calculate the number of encoder counts for the shoulder joint of the AESOP. And after the calculation, that data of counts are send to the AESOP so thus it can move accordingly. The results were promising, as the ground robot moves in Y direction, the AESOP shoulder shown the effects. It moved backward and forward respectively to keep the ground robot in the field of view.

This solution is then included into the algorithm, by creating a function which directly takes the values of the Y position of the marker. After calculation, it converts them in to the encoder counts and then sends the count to the shoulder motor of the AESOP.

Correction was required to move the arm perfectly, such that it keeps tracking the robot while keeping the robot in the center of the field of view. The value which is multiplied to Y position of the marker in the function was tested and changed up to it reacts properly with Y direction movement of ground robot. So now all the parts of the

Tracking system were tested individually. The next step was to create an algorithm that coordinates the system together.

2.2.5 Adding Serial Communication to the Augmented Reality system

As mentioned in the background section Augmented Reality (AR) program can help to combine the physical world with the virtual world. So an algorithm was built in which AR could be used to identify a marker on the SRV-1.

With the help of AR the range of the SRV-1 can be detected and it’s X, Y, Z positions are obtained in camera coordinate system. As the SRV-1 moves around the camera mounted on the arm, the SRV-1 is detected by the marker, and the robot arm moves to follow the SRV-1

When the serial program was ready for communication, it was time to combine them into a single algorithm. An idea of creating a client-server program was also taken into consideration. But the client-server program was not yielding the real-time tracking of the SRV-1, as the data coming from the server takes some time to go through the client program and also to pass through the serial communication.

To overcome this required bringing the serial communication into the AR program. So a new challenge was faced, that is, where to implement the serial port program in AR as the AR program have two main functions. The first function is a main loop, for the initialization of various software modules, such as GLUT and video capture. GLUT is an OpenGL Utility Toolbox that is helpful for various input/output operations. For example it is used to perform various keyboard monitoring and mouse input function. It is also employed to draw 3D objects on the marker like cubes and teapots.

The other function is a loop for capturing the next frame of video, detecting the marker's position and computing the transformation between the camera and the marker.

The serial communication must be active all the time and also must be initialized first, so that the communication is ready to go. Thus the serial code was kept with the rest of initialization code so it was ready at the start of the program.

Once the serial communication was added to the AR system the next task was to use SDL (Simple Direct Media Layer) for communication of SRV-1. SDL is a free cross-platform multimedia development API, which can be used for games and multimedia applications. A Wi-Fi communication could be established using the SDL, to communicate with SRV-1.Before moving to it, first the entire combined algorithm was tested with the SRV-1 with the marker on it. The SRV-1 was driven by its console [19], shown in the figure.

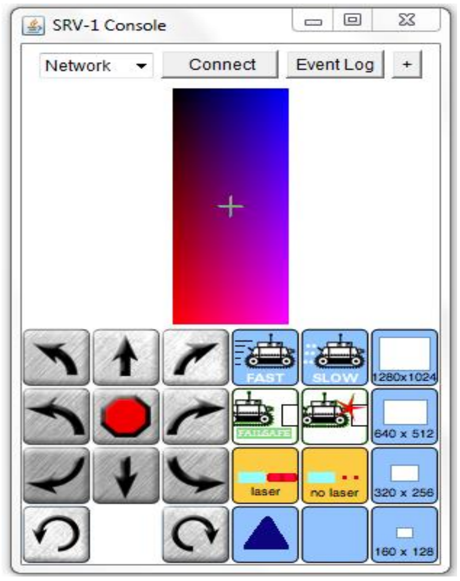


Figure 15: SRV-1 Console by Surveyor [19].

The test result was encouraging and AESOP was reacting according to the movement of the SRV-1, it was tracking the SRV-1. So to complete the system it was also required to have the SDL, which could command and control the SRV-1.So the SDL program was introduced to the combined serial and AR algorithm.

The SDL program must be initialized and closed every time so as to keep the connections active and to communicate with the new packets from the SRV-1. So it was kept in the main loop and in addition a MovementCommand.cpp file was created for commanding the SRV-1 for the movement.

After writing `MovementCommand.cpp` it was included into main algorithm and again checked for the proper operation. Now the SRV-1 is controlled and AESOP was commanded through the single program using the AR software.

The movement of AESOP was checked on the X and Y axis by moving the SRV-1 in X and Y directions, so that to ensure whether SRV-1 was in the middle of the video frame. The AESOP was working as per expectation and working autonomously with accordance to SRV-1.

It was necessary to apply a dead zone in the system, because a minute change in the marker position, affects the position of the arm and it becomes unstable and shakes. To overcome this, a dead zone spanning a certain number of camera pixels was created. The idea was to make a zone or a box of equal pixel values like 40×40 or 50×50 pixels, such that if the marker is inside that particular region the arm does not move at all. Also it could help to keep the marker in the center of the view.

The values of the marker position are analyzed and an area of 100×100 pixels was created accordingly for the dead zone, in such a manner that it work like a linear proportional controller to the dead zone. The AESOP arm keeps on readjusting itself as long as it comes inside the dead zone.

2.2.6 Testing and Results

An efficiency and accuracy test has been performed to evaluate the system's performance. Since the AESOP arm moves in the radius of 180 degrees, a test platform was created. Different points were set up with different lengths to test the tracking system. The following figure shows the working envelope of the AESOP and the different points which are selected for the testing. Table 2 shows various points selected for the test, the distance between two points in inches and the distance travelled by AESOP arm to track the SRV-1.

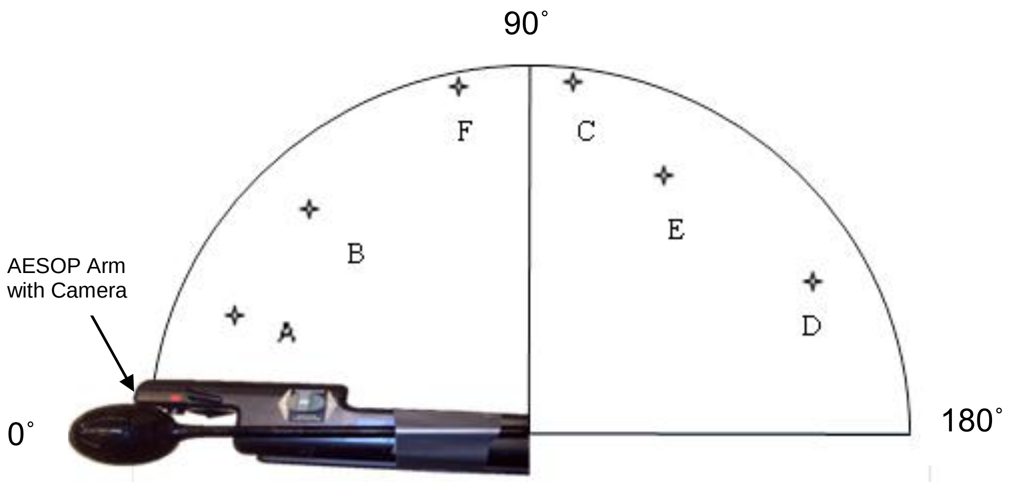


Figure 16: Test Platform setup for testing of the tracking system. The A,B,C are the points setup arbitrarily for measuring the difference between the distance covered by the ground robot vs. AESOP arm moved for tracking it.

Points	Distance (inches)	Aesop moved (inches)
A-B	8	7
A-C	20	19
D-E	10	9
D-C	15	14
D-F	25	24

Table 2: Efficiency test table to shows the distance covered by the AESOP vs SRV-1 moved between two points.

From the above result table it is concluded that the AESOP arm is an inch behind the center of the ground robot. The total length from camera to the endpoint of the AESOP arm is 1 inch; also the length of the dead zone of 100×100 pixels created for the SRV-1 is also approximately 1 inch. The results seems to be promising ensuring that the tracking system is working perfectly as expected, also the ground robot remained at the center of the camera view which shows that the dead zone created was also working as expected.

The length from the camera to the endpoint of the AESOP arm and the length of the dead zone in inches compensate the required reading between the ground robot moved and the aerial robot moved. It is clear from the test that the ground robot is tracked by the aerial robot and it is always at the center of the field of view of the camera.

The results of the test of the tracking system show that it was successful. The AESOP arm was able to autonomously track the SRV-1 when it was in the field of view of the camera. Also the arm kept on moving until the ground robot was in the center of the video screen. As desired the arm adjusted itself in the X and Y axes to ensure the ground robot always remained in the center. Thus the aim of tracking the ground robot using the AR system was fulfilled.

2.2.7 Discussions

2.2.7.1 Limitations Encountered and Solution

A major limitation encountered with this tracking system was that the AESOP arm's movement was not optimized according to the SRV-1.The speed of SRV-1 is much faster than the movement of the arm and hence, the SRV-1 could move out of the field of view of the camera without detection. This problem could be fixed by making changes in the firmware of motion control board and also by reducing the speed of the motors of the SRV-1.

Just to mention the problem is isolated with this AESOP, SRV-1 hardware combination, and this is not a generally in surmountable problem with the approach.

CHAPTER 3

POINT AND GO USING AERIAL CAMERA VIEW

3.1 Introduction

This aim is focused on sending the ground robot to the desired location using the view from a simulated aerial camera residing on the arm of AESOP 3000. The idea behind this project is that the operator or user will command the ground robot by viewing and clicking on the video feed available from the camera which is kept on the aerial robot. The ground robot will go to the clicked screen position and stop and wait for the next command from the operator. To achieve this aim, a test bed was built in which a camera is mounted on arm of AESOP 3000 which is used as an aerial robot and SRV-1 is used as a ground robot.

Figure 16 refers to the overall strategy for achieving this aim. In this system, an Augmented Reality system of patterns with two markers is used for tracking purposes. One marker is fixed on the SRV-1 with respect to the reference marker. Also a Simple Direct Media Layer (SDL) is used to communicate with and move the SRV-1.

Two markers are used in the algorithm, one for setting the World Coordinates and other one to reside on the SRV-1 for local coordinates. This configuration allows for a reference marker and frees the camera system to be movable.

The system level flow chart is given below. It shows the system-level flowchart of the algorithm. Each block in the flowchart provides a step by step working of the system, starting from capturing the aerial camera view from the aerial robot up to moving the ground robot to the clicked destination point.

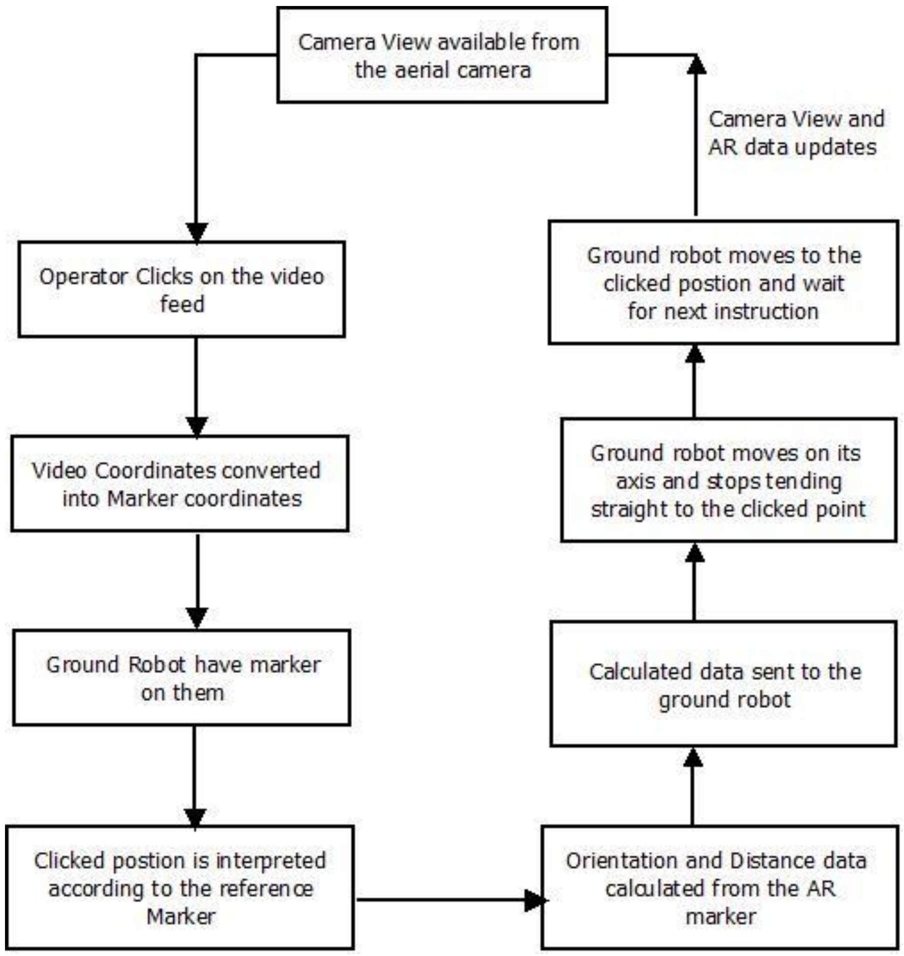

Camera View available from the aerial camera
Camera View and AR data updates
Operator Clicks on the video feed
Ground robot moves to the clicked postion and wait for next instruction
Video Coordinates converted into Marker coordinates
Ground robot moves on its axis and stops tending straight to the clicked point
Ground Robot have marker on them
Calculated data sent to the ground robot
Clicked postion is interpreted according to the reference Marker
Orientation and Distance data calculated from the AR marker

Figure 17: System-level flow chart for Aim2. Here the user has the aerial view from the camera attached to the Aesop arm, he clicks on the video feed, and this system calculates where to move the selected robot and the robot is commanded to move.

3.2 Methods

3.2.1 The AR System and the relationship between the two markers.

As a related work [17] discussed in the previous section it was necessary to design a system that is robust, independent and precise enough to work in different conditions. Before starting to work on the aim, the system was required to work independent of the orientation or position of the camera. For example, if the camera moves the ground robot must go to the chosen location regardless to the position or orientation of the camera.

To overcome this problem a solution was developed with the help of a reference marker created with the AR system. The reference marker provides us an efficient platform to perform this task as the transformation between two markers and the camera system is known. The AR system has the capability of tracking multiple markers at the same time and it computes the position and orientation of all the markers.

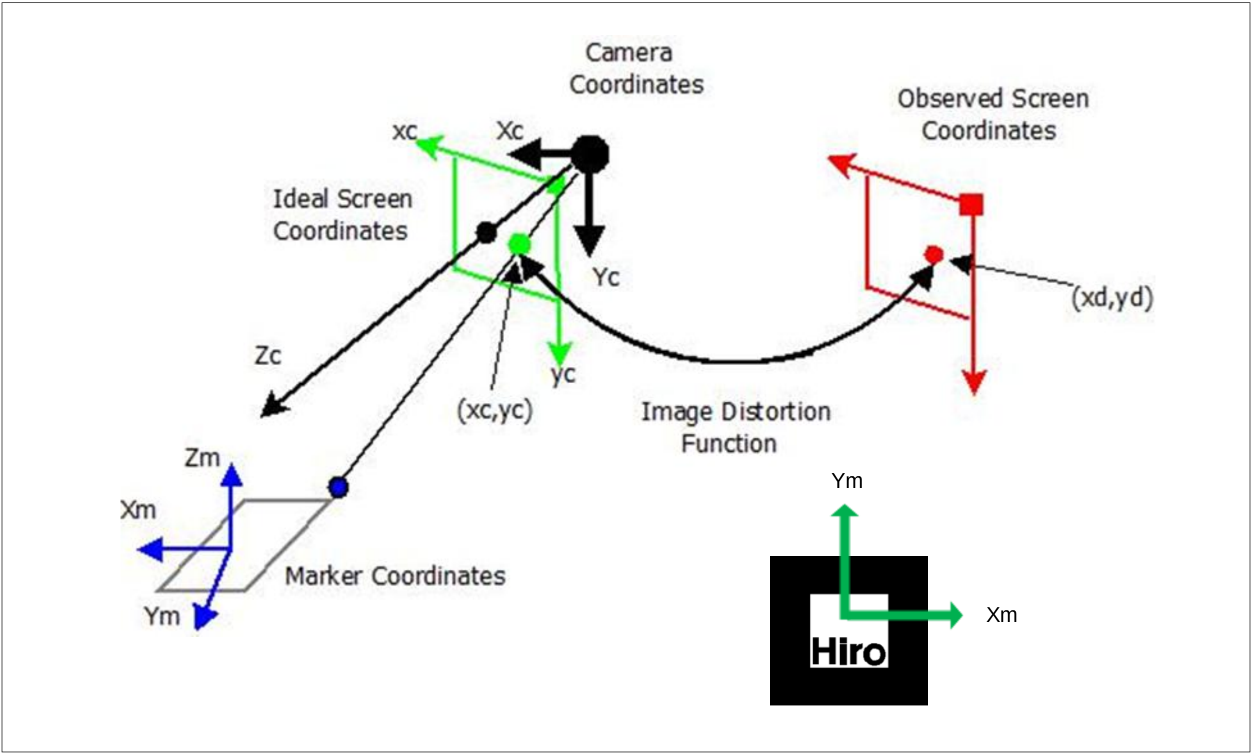
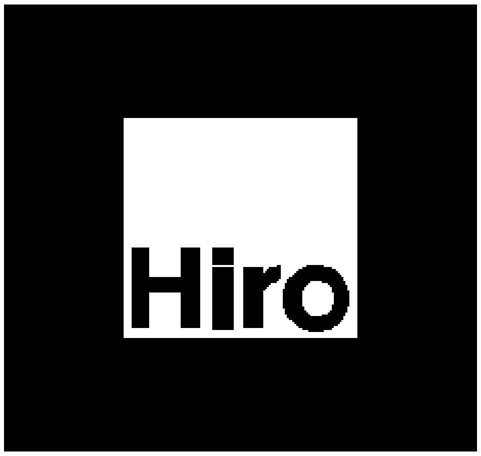
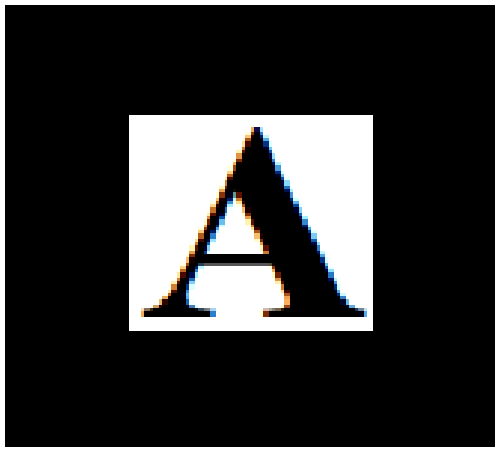


Figure 18: AR Coordinate System showing Marker Coordinate and the screen coordinates and how it is observed in the system. Note here that the marker's coordinate in the real world is knows both in the camera's coordinate system and also in the observed screen coordinates. Tangential and radial distortions parameters are also incorporated for accuracy. Figure adapted from [23].

Therefore a reference marker is setup in such a manner that even if the camera moves or changes its position, as long as the reference marker is visible the ground robot can be directed to the desired position.





Reference Marker SRV-1 Marker

Figure 19: Markers Used.

These are the two markers which are used in the system. Marker A's position is tracked relative to the "Hiro" marker, therefore as long as the Hiro marker is visible the SRV-1 can move to the desired location, because the movement computed is relative the Hiro marker. Hence, the reference marker acts as a global coordinate system for the whole system and the other marker is affixed to the ground robot. Thus as long as the reference marker along with the marker on the SRV-1 is visible, the entire system will work regardless of the cameras position and orientation.

3.2.1.1 World Coordinate System:

The world coordinates are the coordinate that represent the real physical coordinate that are associated with the particular object.

In this case the world coordinates of the marker A is related to reference marker (Hiro) in the physical world. The camera coordinates have just X, Y coordinates in it there is no Z coordinate in camera coordinate system.

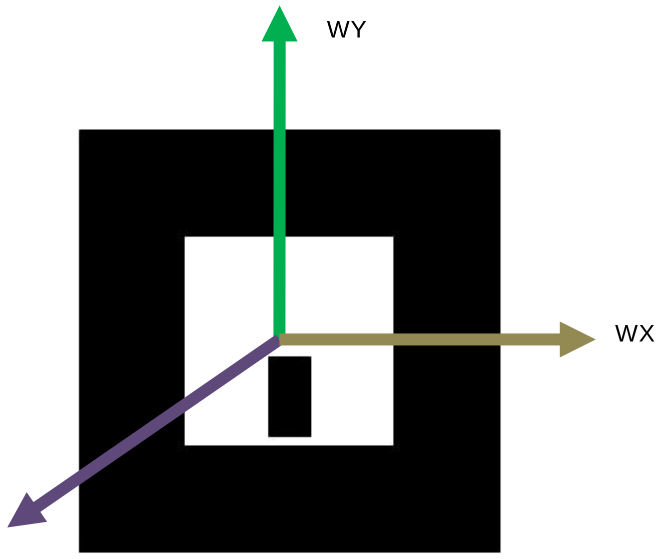
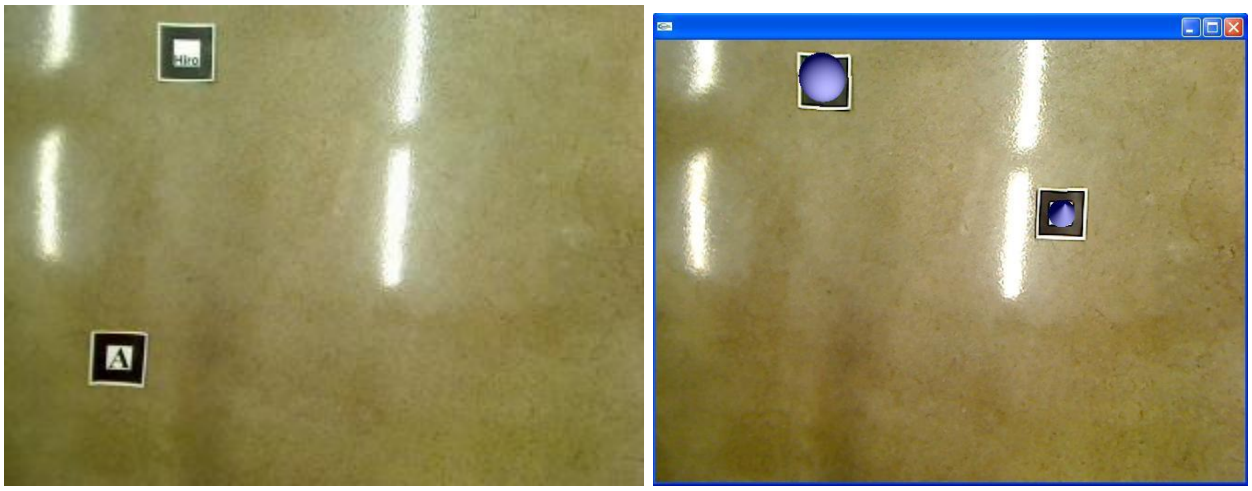


Figure 20: World/ Marker Coordinates of the Pattern in Augmented Reality System.

The above figure describes the axes of the world coordinate system. The WY is Y coordinate of the marker. The WX is the X coordinate of the marker. The WZ always points towards camera. It is used for measuring the distance of the marker from the camera when the "glUnProject" function is used. The "glUnProject" function is used to convert the screen coordinates to the world coordinates. It is required because when the operator clicks on the video feed on the screen, the camera coordinates must be converted to the world coordinate, so the system remains in the same coordinate system.

So per the figure, if the direction of the marker is changed in either the clockwise or counter clockwise direction, the coordinate system moves respectively. So the reference marker is always kept still, as it acts as the base of the world coordinate system for the system. Thus the world coordinate system remains constant.



Normal View AR View

Figure 21: Classification of Two Views the normal view shows, the normal view of the Markers from the camera and the Augmented Reality view shows the 3D objects on the Marker projected by the Augmented Reality system.

Thus, the various data like orientation and position of the referenced marker “A” can be evaluated using the “Hiro” marker and all the data can be calculated with respect to the “Hiro” marker.

The `relation.c` is available in the AR Toolkit, which is used as a template for this algorithm. The program uses two markers and also has a transformation and other code for the two markers. This code is selected as it is a perfect code to start with and it can be further enhanced for various purposes as it will be discussed in next section.

3.2.2 Acquiring World Coordinates based on Reference Marker

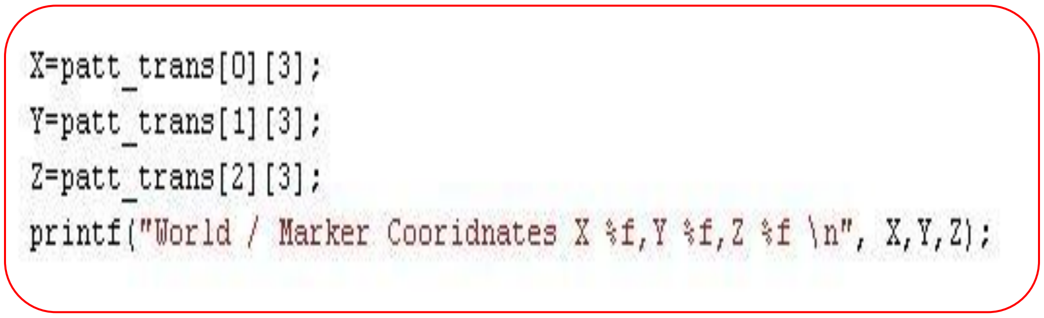
Thus, the various data like orientation and position of the referenced marker “A” can be evaluated using the “Hiro” marker and all the data can be calculated with respect to the “Hiro” marker.

The `relation.c`, which is available in the AR Toolkit, is used as a template for the algorithm. The program uses two markers and also has a transformation and other code for the two markers. This code is selected as it is a perfect code to start with and it can be further enhanced for various purposes as it will be discussed in next section.

3.2.2.1 Projection and its types

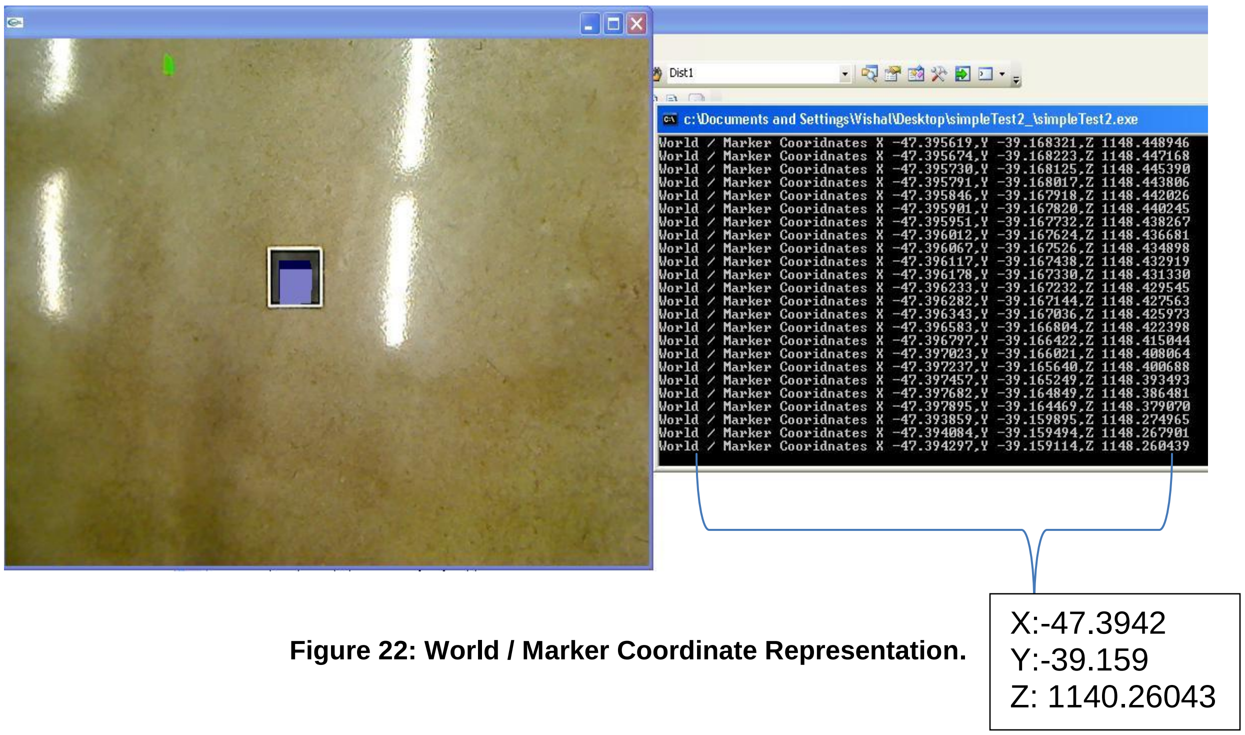
A projection transform is used to convert the 3 dimension object into the 2 dimension that can then be displayed on the camera coordinate system. As the camera coordinate system doesn’t have the z axis. There are two types of Projections viz Perspective and Orthographic. The Perspective Projection is in general the projection that has the X, Y and Z coordinates. Thus it can project the 3 dimension objects to the 2 dimension, therefore it is useful fort the real-time applications and gaming. Whereas the Orthographic Projection does not have the z axis. Thus in turn it cannot render the 3 dimensional objects to 2-dimensional screen.

The following code is used to acquire the world coordinate from the marker.



```
X=patt_trans[0][3];
Y=patt_trans[1][3];
Z=patt_trans[2][3];
printf("World / Marker Cooridnates X %f,Y %f,Z %f \n", X,Y,Z);
```

The output is shown in the below figure after applying this snippet to the algorithm. The command window displays the world coordinate of the “HIRO” marker. The coordinate readings changes slightly (< ~0.2 cm) due to lighting conditions of the environment and error in the computation of the transform.


Dist1
c:\Documents and Settings\Vishal\Desktop\simpleTest2_\simpleTest2.exe

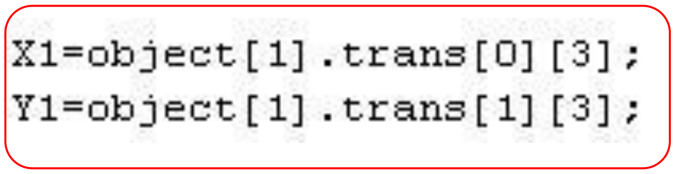
```
World / Marker Cooridnates X -47.395619,Y -39.168321,Z 1148.448946
World / Marker Cooridnates X -47.395674,Y -39.168223,Z 1148.447168
World / Marker Cooridnates X -47.395730,Y -39.168125,Z 1148.445390
World / Marker Cooridnates X -47.395791,Y -39.168017,Z 1148.443806
World / Marker Cooridnates X -47.395846,Y -39.167918,Z 1148.442026
World / Marker Cooridnates X -47.395901,Y -39.167820,Z 1148.440245
World / Marker Cooridnates X -47.395951,Y -39.167732,Z 1148.438267
World / Marker Cooridnates X -47.396012,Y -39.167624,Z 1148.436681
World / Marker Cooridnates X -47.396067,Y -39.167526,Z 1148.434898
World / Marker Cooridnates X -47.396117,Y -39.167438,Z 1148.432919
World / Marker Cooridnates X -47.396178,Y -39.167330,Z 1148.431330
World / Marker Cooridnates X -47.396233,Y -39.167232,Z 1148.429545
World / Marker Cooridnates X -47.396282,Y -39.167144,Z 1148.427563
World / Marker Cooridnates X -47.396343,Y -39.167036,Z 1148.425973
World / Marker Cooridnates X -47.396583,Y -39.166804,Z 1148.422398
World / Marker Cooridnates X -47.396797,Y -39.166422,Z 1148.415044
World / Marker Cooridnates X -47.397023,Y -39.166021,Z 1148.408064
World / Marker Cooridnates X -47.397237,Y -39.165640,Z 1148.400688
World / Marker Cooridnates X -47.397457,Y -39.165249,Z 1148.393493
World / Marker Cooridnates X -47.397682,Y -39.164849,Z 1148.386481
World / Marker Cooridnates X -47.397895,Y -39.164469,Z 1148.379070
World / Marker Cooridnates X -47.393859,Y -39.159895,Z 1148.274965
World / Marker Cooridnates X -47.394084,Y -39.159494,Z 1148.267901
World / Marker Cooridnates X -47.394297,Y -39.159114,Z 1148.260439
```

X:-47.3942
Y:-39.159
Z: 1140.26043

Figure 22: World / Marker Coordinate Representation.

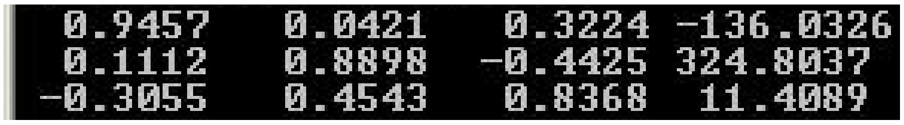
Patt _Trans is the Marker or Marker transformation with respect to the camera. “patt_trans[0][3]”is for X coordinate, “patt_trans[1][3] ”is for Y coordinate and “patt_trans[2][3]” is for Z coordinate. It is computed by the AR toolkit. A tricky part here is that there are two markers in the system and if the following code is applied for getting the world coordinates then the information of both the markers are displayed. To get the marker information individually, it is necessary to find a solution as it is the key for the project.

Both the markers are defined into the structure called “OBJECT_T” and the various related data like transformation of it and marker name are defined into this structure as data members. The variable (object) of the structure is defined as “object”. The idea behind doing this is to retrieve the information according to marker. So the information of the marker could be retrieved using the object element 1 or 2. So to get the information of the specific marker, the corresponding element number is used. The following code is an example of how to get the X and Y coordinate of the marker which is kept as the second element of the structure array:



```
X1=object[1].trans[0][3];
Y1=object[1].trans[1][3];
```

The “object[1].trans” is the transformation of the robot marker, and “object[0].trans” is the transformation of the reference marker. An inverse of transformation of the reference marker to the other marker will give the transformation of marker 2 in marker 1’s coordinate system.



```
  0.9457    0.0421    0.3224 -136.0326
  0.1112    0.8898   -0.4425  324.8037
 -0.3055    0.4543    0.8368   11.4089
```

Figure 23: Values of World/Marker Coordinates.

The above figure shows the 12 elements of the transformation matrix available from the inverse transformation of the two markers. Since the system must be in reference coordinates, it is require to acquire the world coordinates of the robot marker to get the exact position of the robot in real world. So a function “GLunProject” was developed and it can convert the camera coordinates into the world coordinate system.

To get the world coordinate of the robot marker it is required to click on the marker, preferably near its center. The user also has to click on the destination point. At this point, the algorithm has the information of the robot marker in world coordinates with respect to reference marker.

Thus the system is a two-click system which can send the robot to the desired location. it is required to click on the robot marker and then on the desired destination. When the robot stops at the previously clicked position the operator just has to click on new location and the robot will move there. The requirement for the two click system was that the clicked destination position is retrieved from the camera coordinates in terms of world coordinate using “GLunProject” function. The position of the robot could be retrieved from the robot marker, but it does not set with the destination position obtained from the function.

3.2.2.2 Calculating the Distance and Angles between Two Points

It is necessary to obtain a distance between the two points as it will be used for initiating the movement and also how much to move the robot and stop. As there is a two click system, the distance between the robot and the clicked point can be computed using the following Euclidean distance formula

$$D = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2} \qquad (4.1)$$

The above equation provides a distance between the two points. Once this distance is available it can be used for measuring the distance between the robot and the clicked position.

The angle between two points can be acquired by simple coordinate geometry. The angle can be calculated using the origin. Let the origin be at (0, 0). The geometry can be represented as:

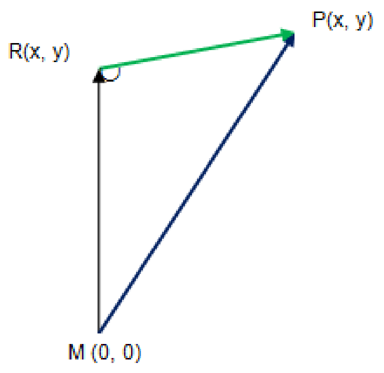


Figure 24: Distance between two linear points.

R(x, y) is the location of the robot and the P(x, y) is the location of the destination point. Here the origin is defined as the center of the robot marker. The angle is calculated using the following equation:

$$A = \tan^{-1}\frac{(y_2 - y_1)}{(x_2 - x_1)} \times {}^{180}/_{\pi} \qquad (4.2)$$

The arc tangent of the x and y coordinates of two points, in this case the position of the robot and the desired location, is multiplied by ${}^{180}/_{\pi}$ to calculate the angle between the robot and the desired location.

The screen coordinates of the robot are converted into the world coordinates before the angle is calculated to ensure that the system is in world coordinate. After the angles are calculated the angle is offset using the quadrant division with respect to the marker.

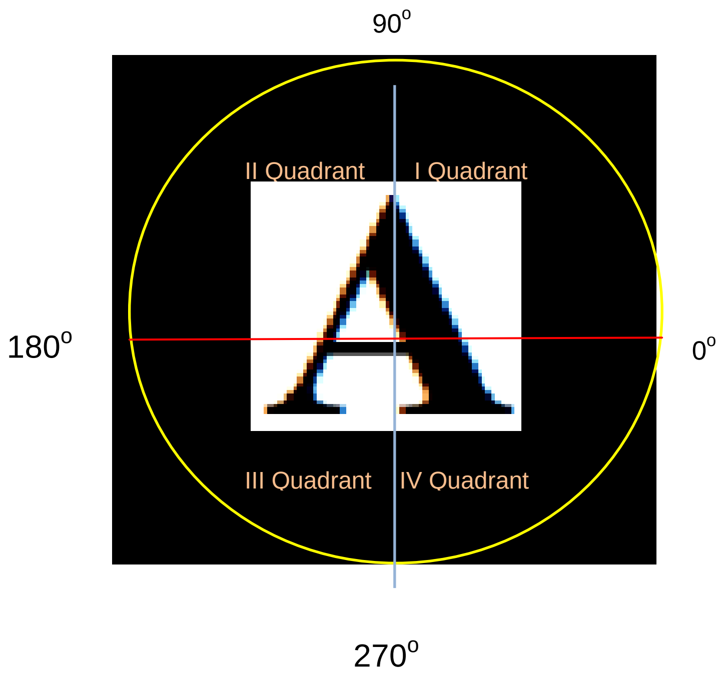


Figure 25: Division of the Robot Marker into Quadrants to get Orientation of the robot.

The above figure shows how the marker is divided into four quadrants. After the angle is calculated it is necessary to adjust the values of the angle according to the quadrants, so that the computed angle is accurate. To obtain an accurate angle a conditional statement is used to add the appropriate value to the computed angle based on the quadrant containing the point. If the destination click is in first quadrant then the angle is not modified, if the click is in second quadrant 90˚ is added to the angle.

After the angle is summed with the appropriate quadrant dependent value, the angle is ready to be used for comparing it with the orientation of the robot marker. It is required to compare it with the orientation because the position of the destination clicked point is to be determined, in such a way that the robot will always move straight to the clicked position

3.2.3 Acquiring the Orientation of Robot Marker based on the Reference Marker

As the angle is calculated based on the destination position to go, it was necessary to know the orientation of the robot for example if the click is at 45 degrees and the orientation of the robot might be at 330 degrees, then it is necessary to move the robot around 40 to 45 degrees such that it could then travel straight to the clicked position.

To achieve this it is necessary to analyze, how the transformation matrix works. The transformation matrix is a 4×4 matrix. It has 16 elements in it. The 4th row of the matrix has 0's and also has 1 in it. So always the 4th row and is discarded and the matrix is viewed as 3×4 matrix. So the total 12 elements are to be considered for the answer of orientation.

$$\begin{bmatrix} \mathbf{00} & \mathbf{10} & \mathbf{20} & \boldsymbol{x} \\ \mathbf{01} & \mathbf{11} & \mathbf{21} & \boldsymbol{y} \\ \mathbf{02} & \mathbf{12} & \mathbf{22} & \boldsymbol{z} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \qquad (4.3)$$

4×4 Transformation Matrix

A 4×4 transformation matrix is shown above. So a 3×4 matrix is generated discarding the 4th row and columns which is shown below

$$\begin{bmatrix} \mathbf{00} & \mathbf{10} & \mathbf{20} & \boldsymbol{x} \\ \mathbf{01} & \mathbf{11} & \mathbf{21} & \boldsymbol{y} \\ \mathbf{02} & \mathbf{12} & \mathbf{22} & \boldsymbol{z} \end{bmatrix} \qquad (4.4)$$

3×4 Transformation Matrix Extracted from 4×4 Transformation Matrix

It is required to concentrate on this matrix, to get the orientation from it. A solution was searched to calculate the orientation from the transformation matrix. The Euler angles and rotational matrix have the answer for it.

3.2.3.1 Euler Angles:

Euler angles are the angle which defines the orientation of the rigid body. They can describe the orientation in the Euclidean's Space. They signify that any orientation could be accomplished by rotating any of the X, Y or Z axes. To achieve the rotations, it is required to convert the matrix to the rotational matrix and then the elements of the matrix will be moved around and the rotation is achieved.

So it was required to have a study of rotational matrix, which elements are required to be move and also how to use them in AR.As the transformation matrix is known, it is required to analyze which axis is required to be moved.

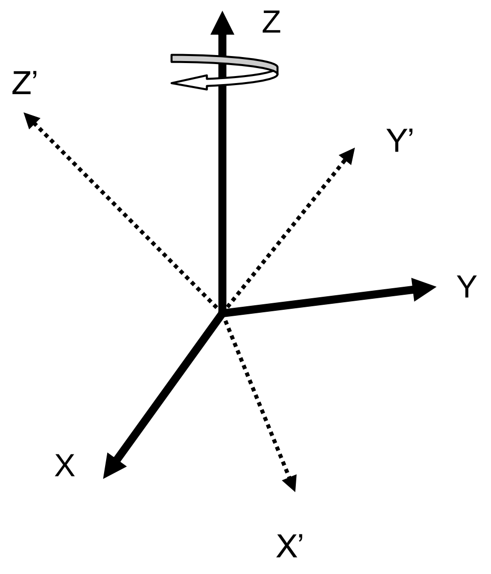


Figure 26: Rotation of the Axis.

To determine the axis, the concept of frustum was considered and the Y axis was chosen to move so a swapping of the elements are undertook. The 3×4 matrix is decomposed to 3×3 matrix, as this 9 elements represent the axis, the elements 10, 11 & 12 are replaced with 20, 21 & 22 as shown in the below matrices.

$$\begin{bmatrix} \mathbf{00} & \mathbf{10} & \mathbf{20} \\ \mathbf{01} & \mathbf{11} & \mathbf{21} \\ \mathbf{02} & \mathbf{12} & \mathbf{22} \end{bmatrix} \qquad (4.5)$$

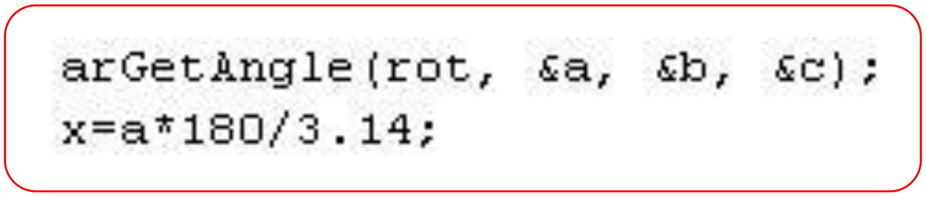
3×3 Transformation Matrix Extracted from 3×4 Transformation Matrix

$$\begin{bmatrix} \mathbf{00} & \mathbf{20} & \mathbf{10} \\ \mathbf{01} & \mathbf{21} & \mathbf{11} \\ \mathbf{02} & \mathbf{22} & \mathbf{12} \end{bmatrix} \qquad (4.6)$$

Rearranged 3×3 Transformation Matrix Extracted

In the rotational matrix the elements of the matrix are move. In our case the elements of the Transformation Matrix, are required to be moved to satisfy Euler Angles. After the concept was clear which elements to move it was time to implement it.

An empty 3×3 rotational matrix is generated and the values are swapped accordingly. Then the orientation is obtained using the following snippet.



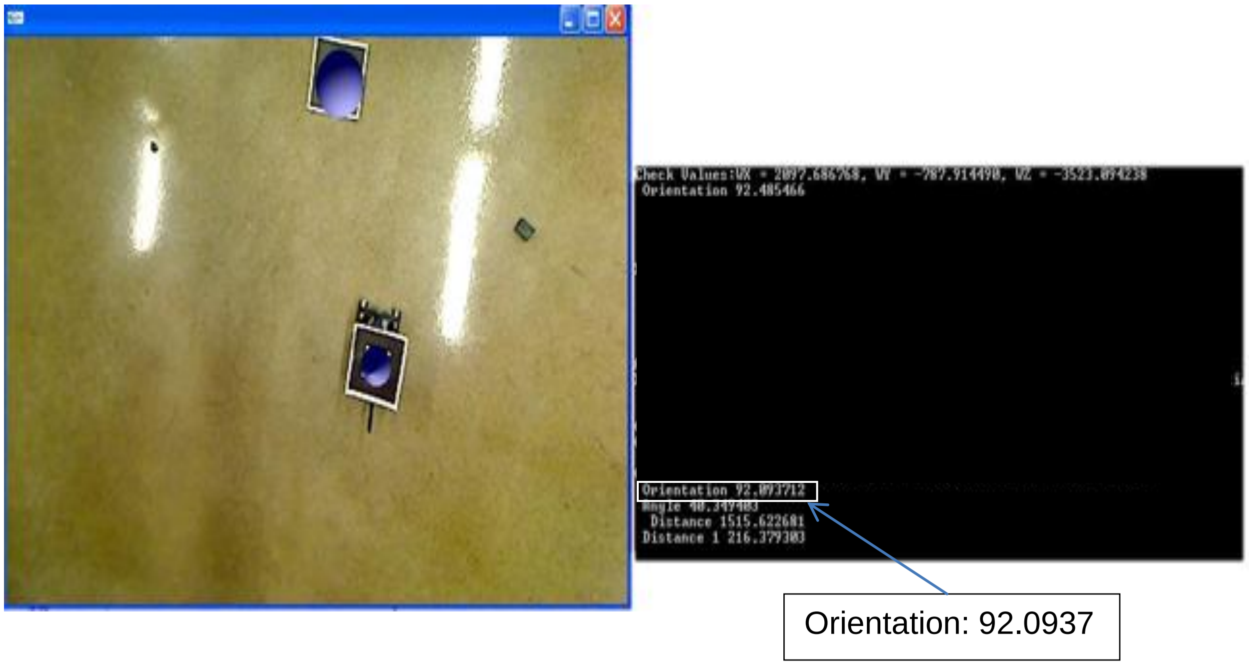
```
arGetAngle(rot, &a, &b, &c);
x=a*180/3.14;
```

The “arGetAngle” function takes the rotational matrix and three variables in it. The values returned from this function are than examined for the orientation. It is required to multiply these values by $^{180}/_{\pi}$ to get the angle of orientation. After an in depth analysis of all three values returned value of variable “a” contains the accurate orientation of the robot marker with respect to reference marker.

As the whole system is dependent on the reference variable thus the values are available according to the orientation of the reference marker. We can get the robot marker reference with respect to the reference marker thus making the entire system camera independent.

3.2.3.2 Test for the Robustness of Orientation Calculation

The following figure shows the orientation of the robot marker which is represented as a cone in AR view and also it represents the robot.

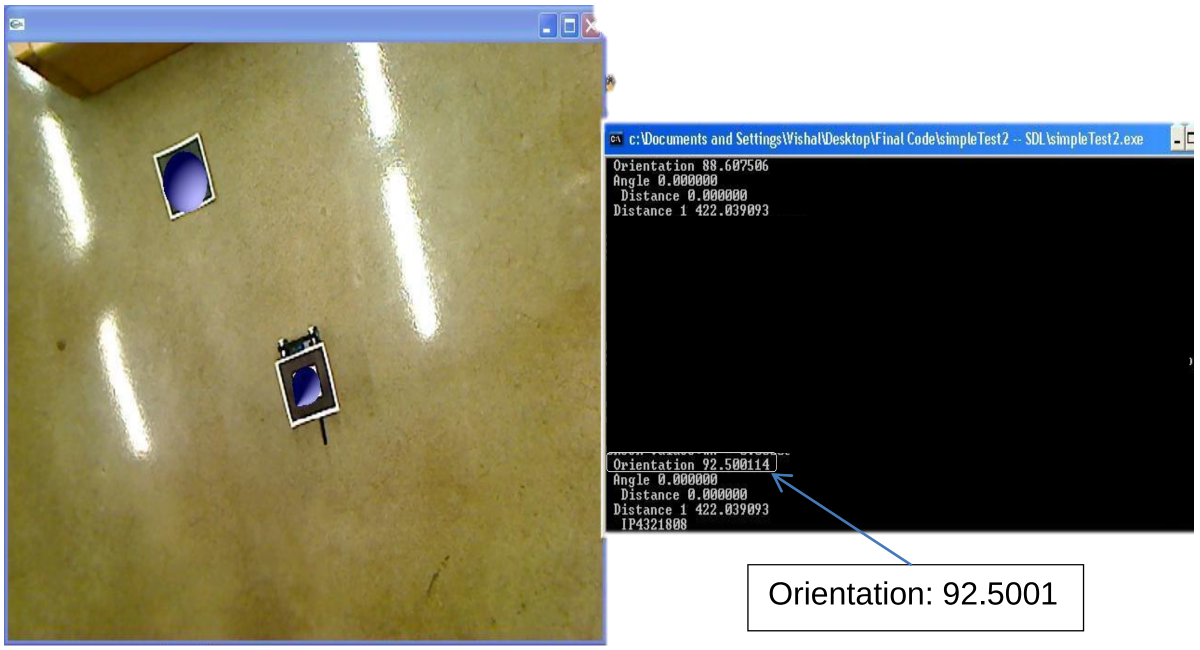


```
Check Values:VX = 2097.686768, VY = -787.914490, VZ = -3523.094238
 Orientation 92.485466

Orientation 92.093712
Angle 40.349403
  Distance 1515.622681
Distance 1 216.379303
```

Figure 27: Representation of Orientation of Robot with respect to Reference Marker.

The above picture shows the orientation of the robot according to the reference marker. The orientation reading is highlighted it is 92.09˚ in this view. The camera is moved to the different location to check whether any changes in the orientation readings.

The following figure shows the change in the orientation of the camera and also the camera is moved at different angle. The reading of the orientation is not changed it is 92.50˚.


c:\Documents and Settings\Vishal\Desktop\Final Code\simpleTest2 -- SDL\simpleTest2.exe
Orientation: 92.5001

```
Orientation 88.607506
Angle 0.000000
 Distance 0.000000
Distance 1 422.039093
```

```
Orientation 92.500114
Angle 0.000000
 Distance 0.000000
Distance 1 422.039093
 IP4321808
```

Figure 28: Robustness of Orientation Calculation.

From previous two pictures it is confirmed that the system is totally camera independent and the readings of orientation does not change according to the movement of the camera.

3.2.4 Combining AR and SDL

After obtaining the orientation it was time to combine the SDL program with the AR program, in such way that any change from AR reading could be captured in SDL as discussed in previous section. The commands used for controlling the SRV-1 are provided by Surveyor Company for the SRV-1. The commands are encapsulate in the following table.

Command “8”	Move forward
Command “5”	Stop
Command “4”	Move Left
Command “6”	Move Right

Table 3: Commands for SRV-1

As per the table Command “8” is used for moving the SRV-1 forward, so after the orientation is available, a conditional statement is generated and the value of orientation is compared with the values of angle available. The angle was added with ± 10 degrees to move the SRV-1 and to adjust itself with the angle. The Command “4” is used to move the robot to move round on its position, such a manner that it adjusts itself with the angle and Command “5” is then issued when the angle is ± 8.5 degrees.

The requirement for making this adjustment is as the traditional AR system is not robust according to the lighting condition of the environment, thus the reading might change ± 5 from the original value, and also sometime the camera gets out of focus due to change in AR system. To surmount these challenges it is required to keep some threshold values, to allow the robot to adjust itself with the AR. As the comparisons get over the, robot is ready to move straight to the destined position. So now Command “8” is issued to move the robot in straight forward, to the clicked position.

A new challenge was how to stop the robot at the clicked position. The distance calculated previously was used for this purpose. A threshold was kept to ensure that the robot did not move ahead of the goal point. So different thresholds was tested to stop the SRV-1 at desired location, after testing the threshold, a value was selected and applied in the code.

As per the table Command “8” is used for moving the SRV-1 forward, so after the orientation is available, a conditional statement is generated and the value of orientation is compared with the values of angle available.

The angle was added with ± 10 degrees to move to adjust itself with the angle. The Command “4” is used to move the robot round on its position, such a manner that it adjusts itself with the angle and Command “5” is issued when the angle is ± 8.5 degrees.

The requirement for making this adjustment was, as the traditional AR system is not robust according to the lighting condition of the environment, thus the reading from the marker might change ± 5 from the original value, and also sometime the camera gets out of focus due to change in light conditions and change in the position of ground robot. To surmount these challenges it was required to keep some threshold values to allow the robot to adjust itself with the AR.

As the comparisons of angle versus orientation get over the, robot is ready to move straight to the destined position. At that point Command “8” is issued to move the robot in straight forward, to the clicked position.

A new challenge was how to stop the robot at click position. So the distance calculate previously was used for this purpose. A threshold was kept to ensure that the robot does not move ahead of the goal point.

So different thresholds was tested to stop the SRV-1 at desired location, after testing the threshold, a value was selected and applied in the code.

3.2.5 Applying Correlation Tracking for point and go.

As discussed in previous section correlation tracking can provide a direct measure of the similarity between two images. So it could provide an additional assurance that even if the camera moves the point where the robot has to go does not changes.

As the correlation tracking works on the camera coordinate system, it could help to map the world coordinates system with the camera coordinate. That is, if the point is defined in the world coordinate can be kept track through camera coordinate. So when the camera moves, the scene changes but the correlation tracking will help to keep the clicked point at same coordinates where it was clicked on screen. Thus a direct relation could be accomplished between the world and camera coordinates.

3.2.5.1 Applying Correlation Tracking to AR system.

Before applying the Correlation Tracking to AR it is important, first to perform an in-depth analysis for how the AR system captures video frame in the program, and after that using the correlation tracking in it. The AR system grabs the video frame through Direct X and stores in the data pointer. There is not a direct access to the pointer to perform image processing. The data pointer is of type “ARUnit 8 *” is a type that defined in unsigned char format. AR stores the images in different format that is in RGB, BGR, and RGBA etc.

To fetch these frames from the pointer an OpenCV IPL image is used. A simple memory move was used to solve this problem. The idea behind using the OpenCV is it is a very useful and powerful tool for the image processing and also once the IPL image is available it could be easily accessible and can be used for various purposes like for further conversion of the image.

First a IPL image was created and the data of the data pointer was moved to the IPL image by keeping the width and height same as it was in AR frame. Once this memory move was successful, the frame is converted to the OpenCV IPL image.

The IPL image is then converted into the 2D Array, to apply the Correlation Tracking to it. A function was created which takes the IPL image in it and also takes the plane for the conversion. The planes are viz Red, Green, and Blue. The function goes through each and every x and y elements of the image and converts them to 2D Array.

The IPL image which was created is passed through this function to create a 2D Array. Once this 2D Array is available it is ready for the Correlation Tracking.

The idea is to provide a single point correlation tracking to the system. To implement the correlation tracking first the Boolean variable is set to true and the Constellation Track Wrapper function is called which takes the 2D image and x and y coordinates and starts tracking to that point. Once the Correlation Tracking is applied, the image data of OpenCV is send back to the data pointer of the AR and thus displayed back over the AR scene.

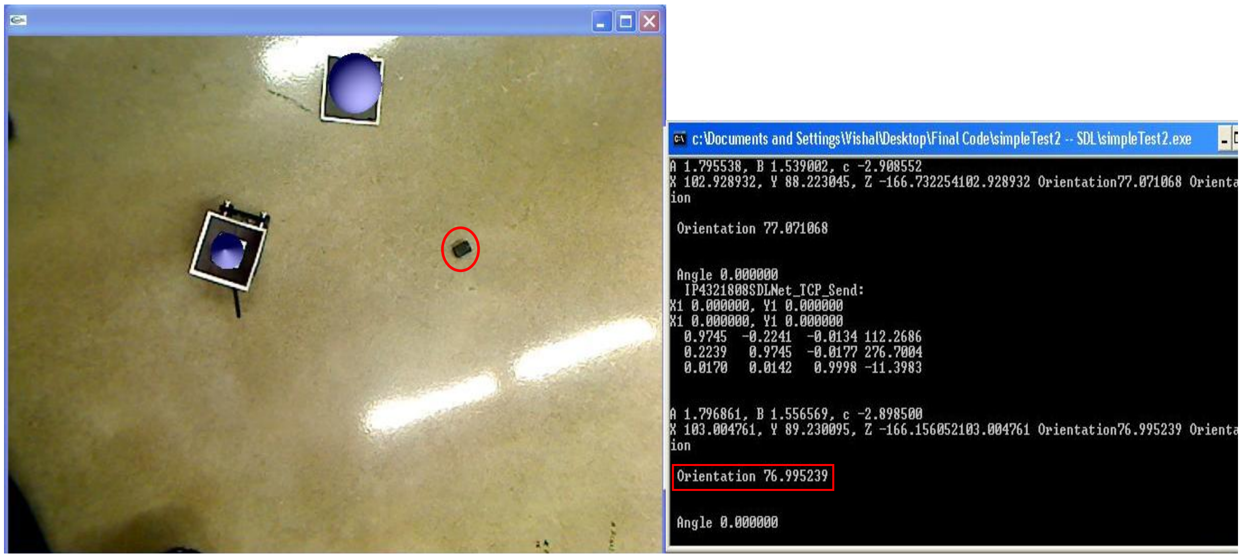
3.3 Testing and Results

The system was tested for its accuracy and efficiency, the robot was send to 3 different locations and its Orientation versus Angle was measure. This provides the detailed information to determine if the robot is moving accurately at its axis and whether it stops at the straight line of its target position. The movement of the robot on its axis is focused and prioritized because it is the backbone of the system and it is required that it must be accurate.

A threshold value was kept of ± 8.6 degrees of the angle. The value was decided after different testing conduct on the spherical movement of SRV-1 and stopping positions, such that the SRV-1 can then move straight to its clicked destination. It allows the robot to stop accordingly on its axis in such a manner that it always point straight towards its clicked destination position. The three different locations are shown in the red circle also the test results are shown in the red rectangles in the following figures.

3.3.1 Test1: Location 1st Quadrant

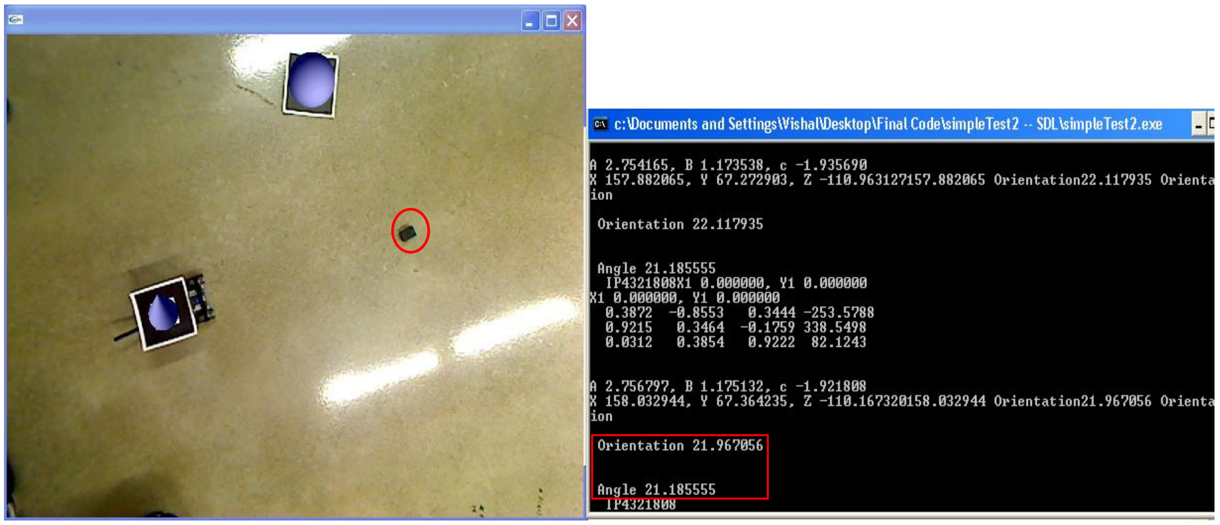
Before test of orientation vs. angle.



```
c:\Documents and Settings\Vishal\Desktop\Final Code\simpleTest2 -- SDL\simpleTest2.exe
A 1.795538, B 1.539002, c -2.908552
X 102.928932, Y 88.223045, Z -166.732254102.928932 Orientation77.071068 Orienta
ion

 Orientation 77.071068


 Angle 0.000000
  IP4321808SDLNet_TCP_Send:
X1 0.000000, Y1 0.000000
X1 0.000000, Y1 0.000000
  0.9745  -0.2241  -0.0134 112.2686
  0.2239   0.9745  -0.0177 276.7004
  0.0170   0.0142   0.9998 -11.3983


A 1.796861, B 1.556569, c -2.898500
X 103.004761, Y 89.230095, Z -166.156052103.004761 Orientation76.995239 Orienta
ion

 Orientation 76.995239


 Angle 0.000000
```

Figure 29: Initial Condition.

After test of orientation versus angle.



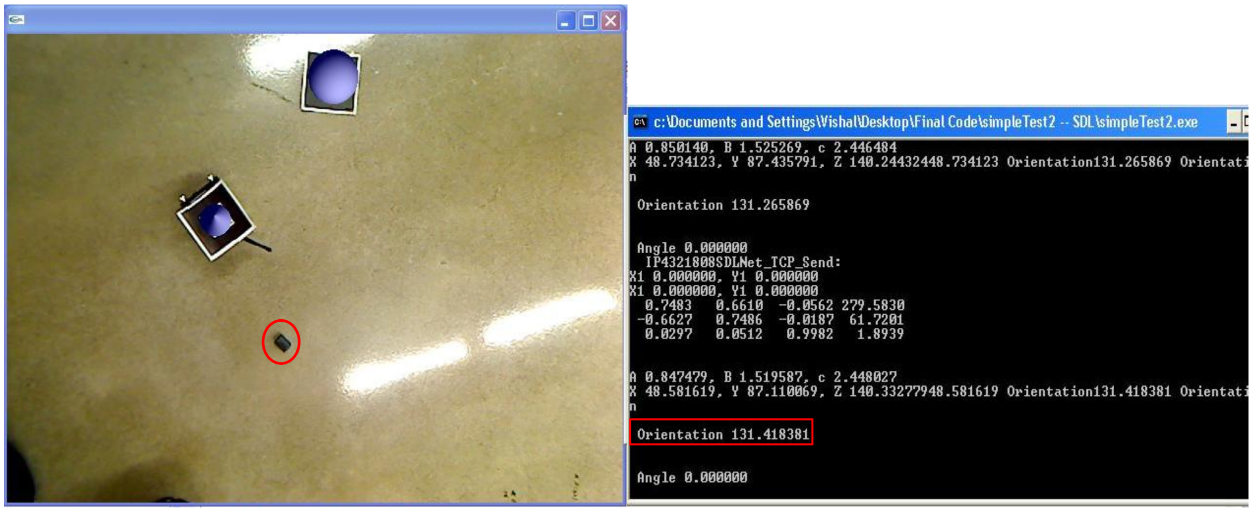
```
c:\Documents and Settings\Vishal\Desktop\Final Code\simpleTest2 -- SDL\simpleTest2.exe
A 2.754165, B 1.173538, c -1.935690
X 157.882065, Y 67.272903, Z -110.963127157.882065 Orientation22.117935 Orienta
ion

 Orientation 22.117935


 Angle 21.185555
  IP4321808X1 0.000000, Y1 0.000000
X1 0.000000, Y1 0.000000
  0.3872  -0.8553   0.3444 -253.5788
  0.9215   0.3464  -0.1759 338.5498
  0.0312   0.3854   0.9222  82.1243


A 2.756797, B 1.175132, c -1.921808
X 158.032944, Y 67.364235, Z -110.167320158.032944 Orientation21.967056 Orienta
ion

 Orientation 21.967056


 Angle 21.185555
  IP4321808
```

Figure 30: Change in Orientation after Movement of Robot.

From the Figure 28 the orientation of the robot is 76.9952, and after clicking and moving the robot to the position from the Figure 29 the angle was calculated as 21.1855 and the orientation of the robot after stopping was 21.9670**.** This proves that the SRV-1 has stopped perfectly at the threshold range of ± 8.6 degrees of the angle calculated.

3.3.2 Test2: Location 3rd Quadrant

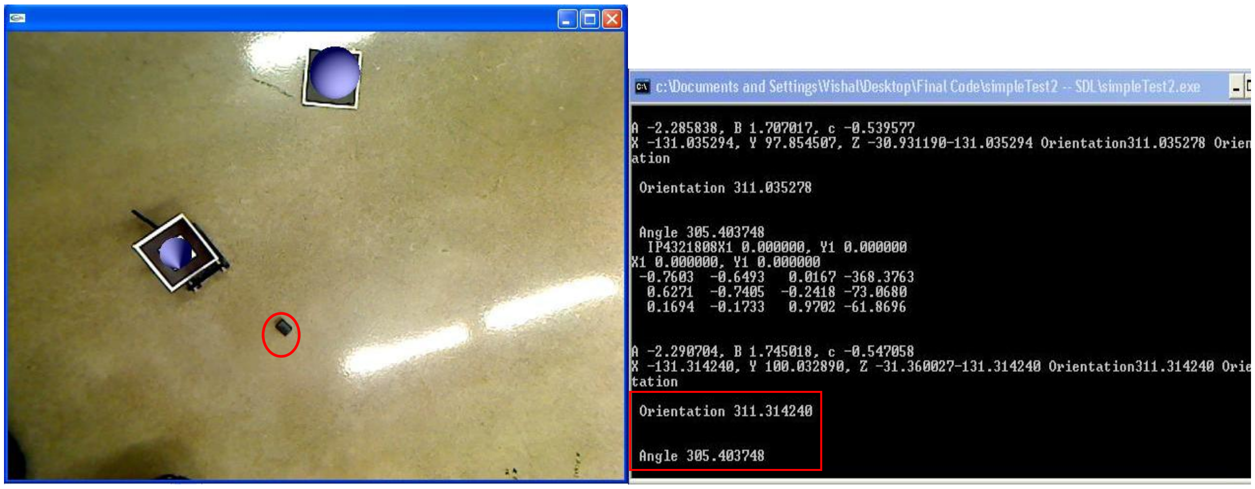
Before test of orientation vs. angle.


c:\Documents and Settings\Vishal\Desktop\Final Code\simpleTest2 -- SDL\simpleTest2.exe

```
A 0.850140, B 1.525269, c 2.446484
X 48.734123, Y 87.435791, Z 140.24432448.734123 Orientation131.265869 Orientati
n

 Orientation 131.265869


 Angle 0.000000
  IP4321808SDLNet_TCP_Send:
X1 0.000000, Y1 0.000000
X1 0.000000, Y1 0.000000
  0.7483   0.6610  -0.0562 279.5830
 -0.6627   0.7486  -0.0187  61.7201
  0.0297   0.0512   0.9982   1.8939


A 0.847479, B 1.519587, c 2.448027
X 48.581619, Y 87.110069, Z 140.33277948.581619 Orientation131.418381 Orientati
n

 Orientation 131.418381


 Angle 0.000000
```

Figure 31: Initial Condition.

After test of orientation vs. angle.


c:\Documents and Settings\Vishal\Desktop\Final Code\simpleTest2 -- SDL\simpleTest2.exe

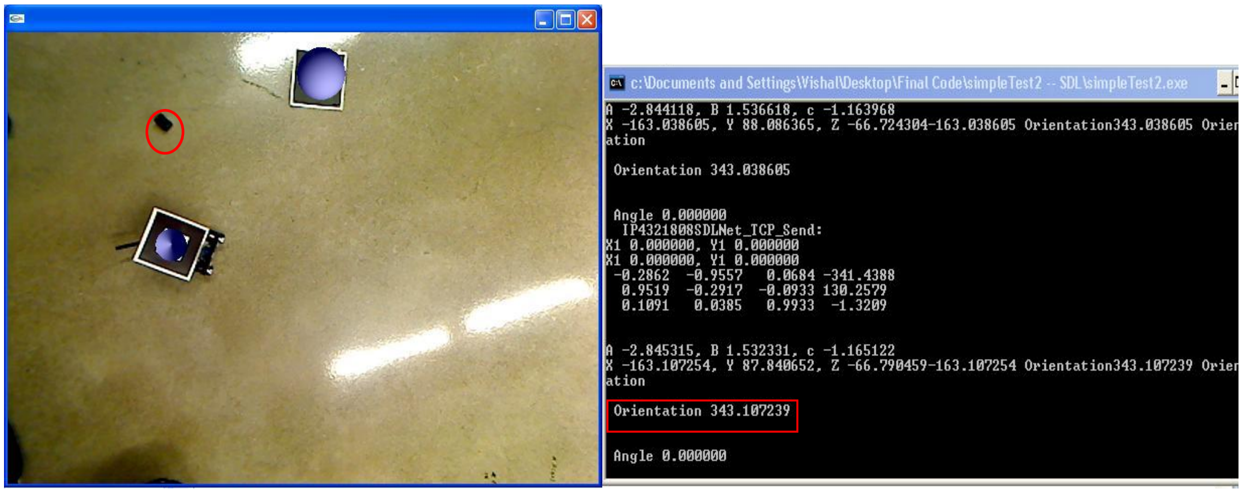
```
A -2.285838, B 1.707017, c -0.539577
X -131.035294, Y 97.854507, Z -30.931190-131.035294 Orientation311.035278 Orien
ation

 Orientation 311.035278


 Angle 305.403748
  IP4321808X1 0.000000, Y1 0.000000
X1 0.000000, Y1 0.000000
 -0.7603  -0.6493   0.0167 -368.3763
  0.6271  -0.7405  -0.2418 -73.0680
  0.1694  -0.1733   0.9702 -61.8696


A -2.290704, B 1.745018, c -0.547058
X -131.314240, Y 100.032890, Z -31.360027-131.314240 Orientation311.314240 Orie
tation

 Orientation 311.314240


 Angle 305.403748
```

Figure 32: Change in Orientation after Movement of Robot.

From the Figure 30 the orientation of the robot is 131.4813 ˚, and after clicking and moving the robot to the position from the Figure 31 the angle was calculated as 305.4037˚, and the orientation of the robot after stopping was 311.3142 ˚.

3.3.3 Test3: Location 2nd Quadrant

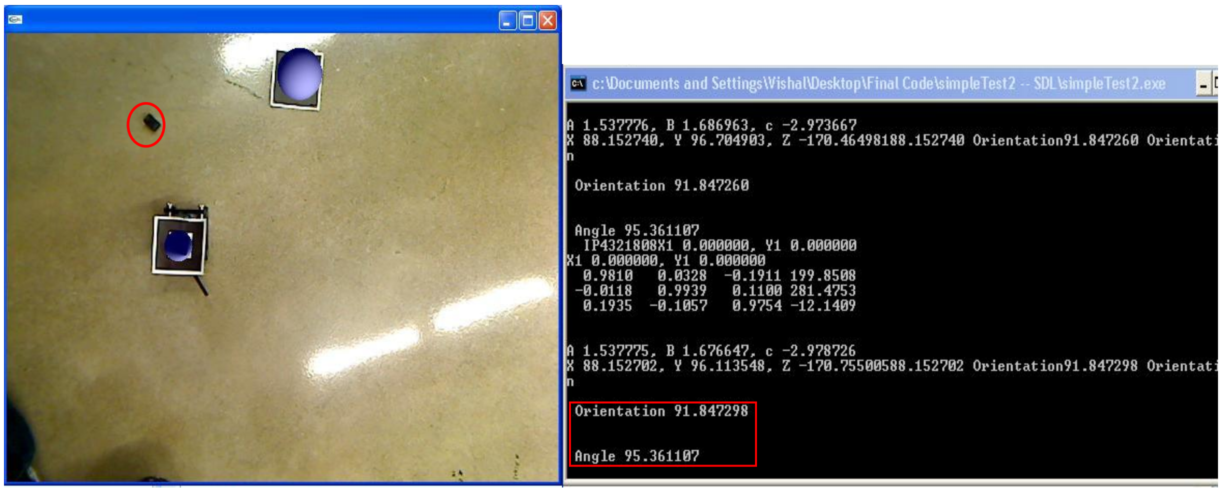
Before test of orientation vs. angle.



```
c:\Documents and Settings\Vishal\Desktop\Final Code\simpleTest2 -- SDL\simpleTest2.exe
A -2.844118, B 1.536618, c -1.163968
X -163.038605, Y 88.086365, Z -66.724304-163.038605 Orientation343.038605 Orien
ation

 Orientation 343.038605


 Angle 0.000000
  IP4321808SDLNet_TCP_Send:
X1 0.000000, Y1 0.000000
X1 0.000000, Y1 0.000000
 -0.2862  -0.9557   0.0684 -341.4388
  0.9519  -0.2917  -0.0933 130.2579
  0.1091   0.0385   0.9933  -1.3209


A -2.845315, B 1.532331, c -1.165122
X -163.107254, Y 87.840652, Z -66.790459-163.107254 Orientation343.107239 Orien
ation

 Orientation 343.107239


 Angle 0.000000
```

Figure 33: Initial Condition.

After test of orientation vs. angle.



```
c:\Documents and Settings\Vishal\Desktop\Final Code\simpleTest2 -- SDL\simpleTest2.exe

A 1.537776, B 1.686963, c -2.973667
X 88.152740, Y 96.704903, Z -170.46498188.152740 Orientation91.847260 Orientati
n

 Orientation 91.847260


 Angle 95.361107
  IP4321808X1 0.000000, Y1 0.000000
X1 0.000000, Y1 0.000000
  0.9810   0.0328  -0.1911 199.8508
 -0.0118   0.9939   0.1100 281.4753
  0.1935  -0.1057   0.9754 -12.1409


A 1.537775, B 1.676647, c -2.978726
X 88.152702, Y 96.113548, Z -170.75500588.152702 Orientation91.847298 Orientati
n

 Orientation 91.847298


 Angle 95.361107
```

Figure 34: Change in Orientation after Movement of Robot.

From the Figure 32 the orientation of the robot is 343.1072, and after clicking and moving the robot to the position from the Figure 33 the angle was calculated as 95.3611, and the orientation of the robot after stopping was 91.8472.

The results of the three tests which are shown above is encapsulated in the following table

Orientation (Before test)	Angle	Orientation (After test)
76.9952˚	21.1855 ˚	21.9670 ˚
131.4813 ˚	305.4037 ˚	311.3142 ˚
343.1072 ˚	95.3611 ˚	91.8472 ˚

Table 4: Analysis of the Orientation and Angle

From the above table it may be concluded that the orientation of the robot is in the range of the angle calculated which is ± 8.6 degrees of the angle, thus proving that the system is working as expected.

As per the results of the test it is clear that the robot is moving on its axis and stops at the threshold value range. The robot can stop at + 8.6 degrees of the angle value or -8.6 degrees of the angle value.

3.4 Discussion

3.4.1 Limitations Encountered and Solution

One potential problem was evoked through this test that the robot could stop a little before or next values of the angle, which may results in an error and causes the robot to move little away from the required destination. Also the SRV-1 gets inclined when it travels straight at long distances. Slight errors in the angles, can result in large distance errors at longer distances. This problem could be overcome by using an infrared source, sensors, or an IMU board for the SRV-1 which could give the position of the SRV-1 correctly which could be cross checked with augmented reality values and thus SRV-1 could be sent to the correct location regardless of the problems.

Another problem is encountered is that the camera has an autofocus facility, that is not compatible with the augmented reality system. Since the AR system is based on visual information, the auto focus capability poses a problem for the AR working properly. Also if the robot moves on the ground or the light is diffracted the autofocus feature comes into action and tries to adjust itself which affects the accuracy of the system. This problem could be overcome using the camera without autofocus or the ability to control the autofocus from the algorithm it could also resolved using perfect light conditions, and also though leveling the sensitivity of the camera.

CHAPTER 4

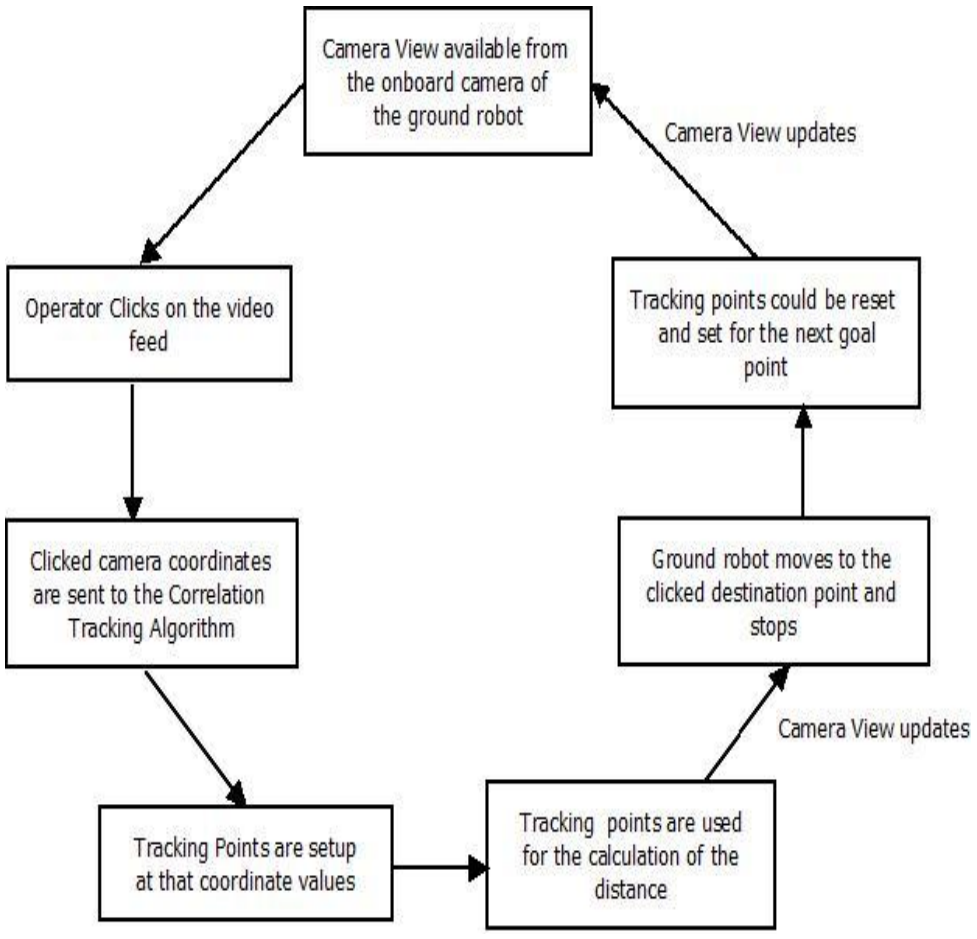
TELEOPERATION AND POINT AND GO USING ONBOARD CAMERA VIEW

4.1 Introduction

Typical teleportation systems utilize some form of joystick control to move robots from remote locations. This form of control can be time-consuming and cumbersome for the user. For instance, when controlling a robot from an oblique view, the left and right movements for the robot can be reversed. This can cause confusion. Automated systems have also been developed, but are not yet sophisticated enough for fully manual approaches. The system developed here is considered semi-autonomous. The user still inputs the goal conditions, but, the system automatically maneuvers to the desired location.

Figure [34] shows the system-level flowchart of the algorithm. Each block in the flowchart provides a step by step working of the system, starting from capturing the onboard camera view from the ground robot up to stopping and resetting the tracking points to send the robot to new location.

The prototype built here is a simple model which is aimed to provide a “point and go” capability to a simple ground robot (the SRV-1). The system uses the robots real-time video feedback from its onboard camera and is provided a tracking point by the user clicking on the screen. Once a goal point is designated, the correlation tracker starts tracking the goal point. Another point is a placed a set number of pixels apart. As the robot is moved towards the tracked point, the distance between the two points increase. We use a threshold distance between the two points as our stopping metric. As soon as SRV-1 reaches the destination it is kept in a standby mode ready for other tasks. It can also be sent to other positions using a reset feature which is implemented in the algorithm. Manual control was also provided in the algorithm that increases the capability of overall system, as the SRV-1 can be used manually for the surveillance of the surrounding environment. Visual Servoing could be then activated to move the SRV-1 autonomously.



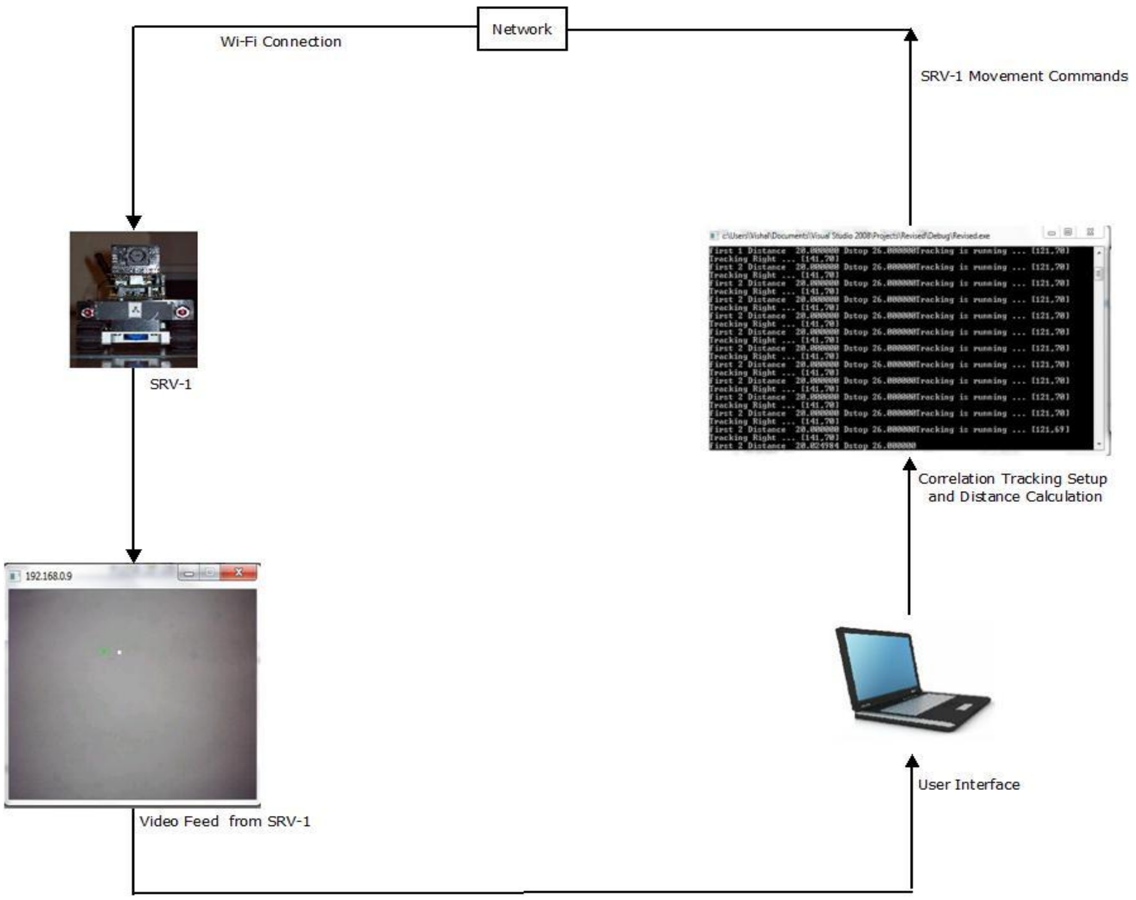
Camera View available from the onboard camera of the ground robot
Camera View updates
Operator Clicks on the video feed
Tracking points could be reset and set for the next goal point
Clicked camera coordinates are sent to the Correlation Tracking Algorithm
Ground robot moves to the clicked destination point and stops
Camera View updates
Tracking Points are setup at that coordinate values
Tracking points are used for the calculation of the distance

Figure 35: System-level flowchart of Aim 3. Here, the user has access to the ground robot's onboard camera view. As the user clicks on the view, a correlation tracking system tracks these points. A control algorithm then moves the robot to the user chosen location.

4.2 Methods

4.2.1 Hardware Setup

The figure below shows the hardware setup of the system. The SRV-1 is connected to the network though a Wi-Fi connection. The SRV-1 provides the video feed through its onboard camera to the user interface. The correlation tracking is setup and distance is calculated. The movements commands are send back to the SRV-1 via the network. Each block is described in details as a module of the methods section below.


Network
Wi-Fi Connection
SRV-1 Movement Commands
SRV-1
c:\Users\Vishal\Documents\Visual Studio 2008\Projects\Revised\Debug\Revised.exe
Correlation Tracking Setup and Distance Calculation
192.168.0.9
User Interface
Video Feed from SRV-1

```
First 1 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
First 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
First 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
First 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
First 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
First 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
First 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
First 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
First 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
First 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
First 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
First 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,69]
Tracking Right ... [141,70]
First 2 Distance  20.024984 Dstop 26.000000
```

Figure 36: Hardware Setup shows the flow from the various components of the system.

4.2.2 Video Feedback from SRV-1

Video feedback from the SRV-1 is the most important part of the system. It is necessary because it can be used for Teleoperation and also used for “point and go” and navigation of the robot.

It is necessary to setup the robot first to start communicating with the processing server. Therefore the SRV-1 is connected through the Wi-Fi Router, by using the IP address and the port number assigned. The SRV-1 is configured and setup to the infrastructural mode and an IP address of 192.168.0.9 and port number: 10001 is assigned to it. By this the SRV-1 can be connected and used. The SRV-1 is connected, by using SDL and SDL NET. First the IP address and Port No is assigned in the global variable of the program and the SDL NET is initialized in the main program.

Two functions were created and used to store and set the video frame received from the SRV-1. The SDL is used as follows: (1) it is used to display images; (2) it is also the user interface for getting user-selected points.

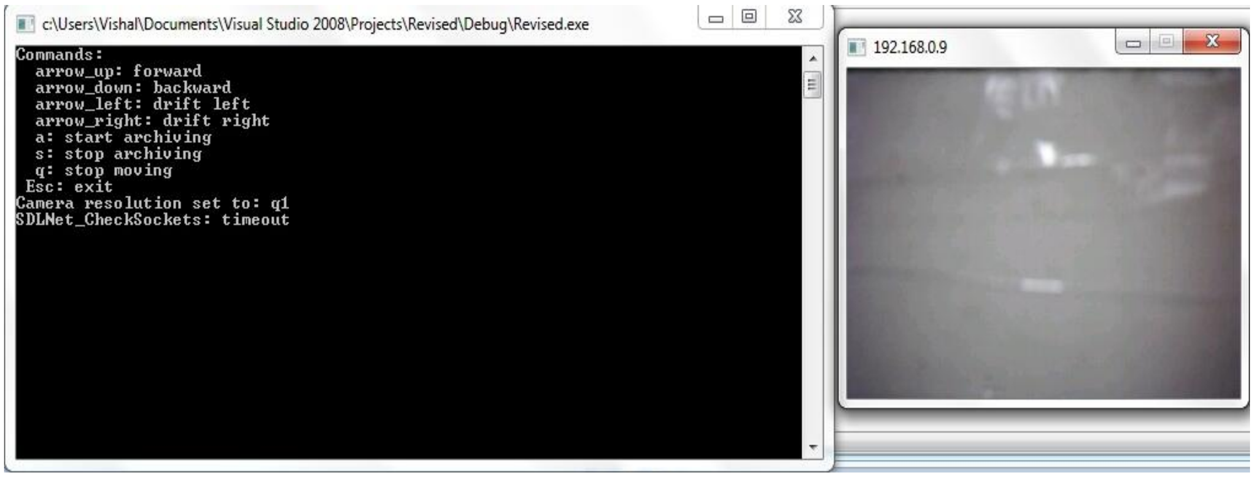
Also a command (”I”) was used, which is a predefined function provided with the SRV-1 to grab the frame. Once the frame is grabbed it is set to appropriate frame size or resolution and displayed to the user.

In this program a function is created which is used to connect to the TCP IP packets and the image frame is fetched from the socket connection. This function takes these arguments 1) TCP Socket, 2) SDL_ Surface **, 3) SDLNet_SocketSet.

The TCP socket is for connecting to the socket available. The SDL surface used for displaying the video frames available from the robot and the SDLNet socketset are used for connecting the socket set for various purposes as it helps to keep the server unblocked. A function was created to show the frames on the screen. The following snippet shows that function.

```
ShowFrame(imageBuf, index, (*screen));
```

This function takes the image buffer and displays it on the SDL surface that is the output screen. These steps enable the real time video from the robot to start. The following figure shows the command window and the video feedback from the SRV-1.


c:\Users\Vishal\Documents\Visual Studio 2008\Projects\Revised\Debug\Revised.exe
192.168.0.9

```
Commands:
  arrow_up: forward
  arrow_down: backward
  arrow_left: drift left
  arrow_right: drift right
  a: start archiving
  s: stop archiving
  q: stop moving
 Esc: exit
Camera resolution set to: q1
SDLNet_CheckSockets: timeout
```

Figure 37: Video Feedback from SRV-1.

Once the video feed is available the SRV-1 can be used for Teleoperational mode. It can be used for survillance because the view from the robot is now available. As the video feed is available, an operator can perform several operations like moving forward, backward, left and right. The `MovementCommand.cpp` was integrated with the algorithm to provide manual movements to the SRV-1 using the SDL events.Table 5 shows various key functions to command the SRV-1.

UP Arrow Key	Move forward
DOWN Arrow Key	Stop
LEFT Arrow Key	Move Left
RIGHT Arrow Key	Move Right

Table 5: Commands to Move SRV-1 using the algorithm

4.2.3 Correlation Tracking Setup and Distance Calculation

When the user clicks on the screen indicating the location to which the robot should move, tracking points appear on the screen (see Figure 37). Tracking points are just the rectangles that show up when operators clicks on the video feed or the screen. A multiple point correlation tracking structure was setup because this “point and go” system is a purely visual based system and also the SRV-1 is not equipped with specialized sensors, odometers or IMU devices. Thus it was required to setup a multiple point system which can be used for various purposes like measuring the distance and calculating a stopping metric.

When the operator clicks on the screen, the two rectangles of green and white color show up on the screen. These rectangles are tracking points which are set using correlation trackers.

In this system a simple correlation technique provided by Turing Associates [14] is used. A drawback of this Correlation Technique is that it is not robust with the different light conditions and changes according to different frames.

To setup a correlation tracking in the program a `CorrelationTracker.h` file along with a correlation tracker library file are required to be added. Also a configuration setting file is also required to be kept in the corresponding directory of the system.

To add the correlation tracking to the algorithm, it is first required to convert the video frame in the image and store it. The next and most important step is to convert this image to a 2D Array. The requirement of converting the image to a 2D array ensures that each element is represented by a pixel value.

After converting the image into a 2D array it is stored for further processing. A Boolean flag was created and initialized to true. This Boolean variable will help to enable further process and if it is false the loop end. Also a wrapper function was created which takes as input the 2D array of the image and the pixel coordinate of the track point. This function is declared in `CorrelationTracker.h` file and it is pointing to the correlation tracker Class. This function enables the tracking by assigning the pixel value and the image as its parameter.

After successfully execution of the function, the system draws the tracked points by generating SDL rectangles of height and width of 4 pixels? Once they are connected, it shows up on the screen which is displayed to the operator (See Figure 37). It is required to have another tracking point for which to compare the distance to so, another rectangle is generated and tracked. This is also a provision to generate the other point while computing the relative position to the first point. So another tracking point is generated at the right side of the previous one, to achieve the multiple point correlation tracking. After all the above inputs, the algorithm is able to track the points in the video frame, using nearby tracking features to the original points.

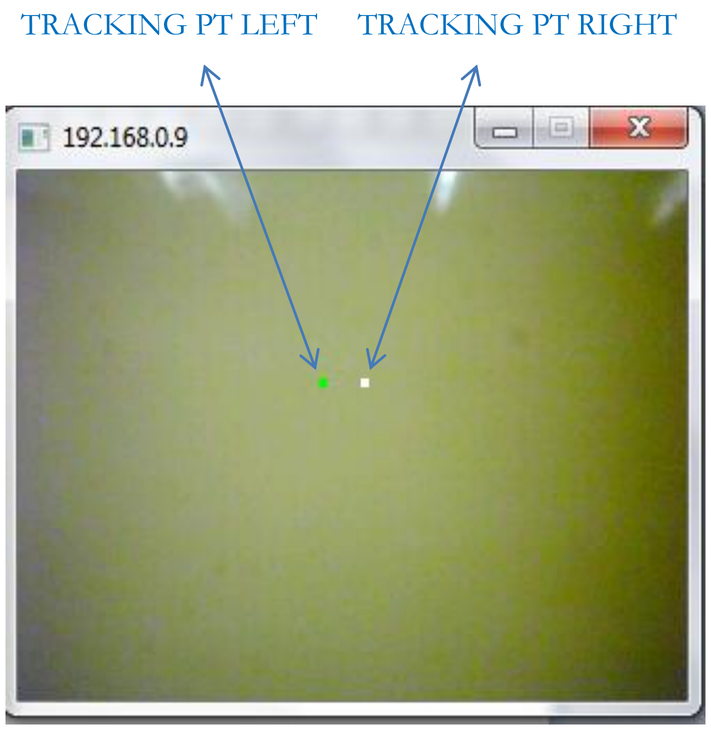


Figure 38: Once the user selects a set of points in the video view, the correlation tracker tracks these points in every video frame.

The above figure shows the two points, which are SDL rectangles as discussed above. These green and white rectangles indicate that the tracking is active.

These points remain static at their scene positions under ideal lighting conditions. However, they move or change when bad lighting and other background conditions are encountered. The main feature is that the points move near to each as the SRV-1 moves away from the target, and move apart when the SRV-1 moves closer. Hence, depending on the position and movement of the tracked points, the movement of the SRV-1 can be ascertained. This movement is related to the field of view of the camera and the relative position of the robot to the originally tracked points. Again, the algorithm works well in ideal conditions, but fails if lighting is altered. Hence, algorithm robustness is an area of future work.

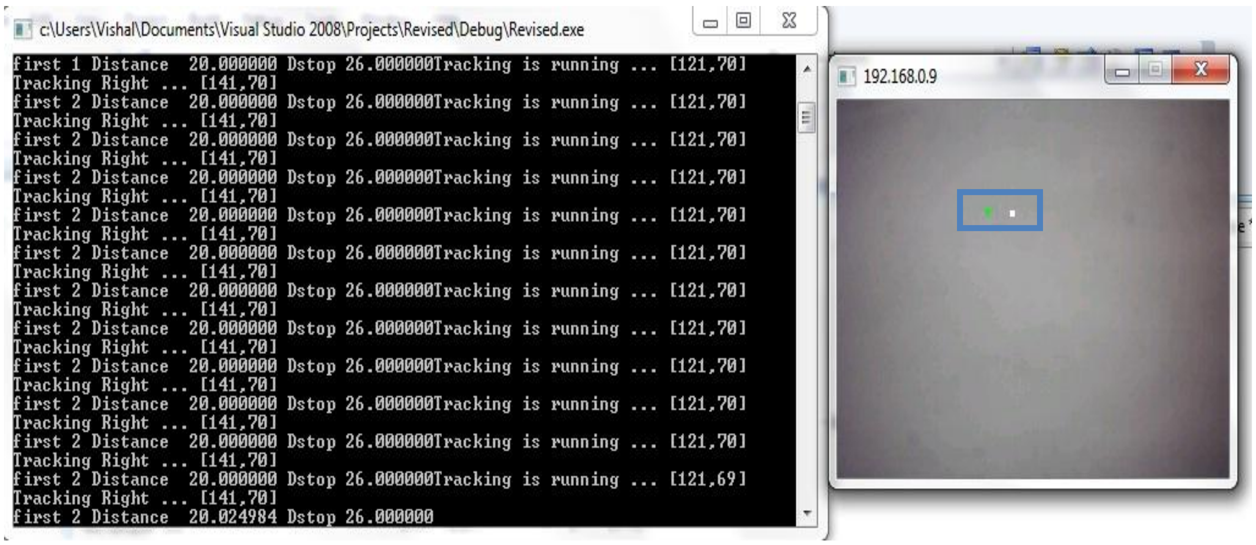
The importance of two tracking points is that, they play a lead role for calculating distance between the SRV-1 and the destination. These points move away from each other as the SRV-1 moves near to the destination. A threshold pixel distance needs to be determined. In order to determine a good stopping distance, a calibration of the required distance needs to be mapped to the pixel distance needs to be performed. Hence, the algorithm will stop the SRV-1 when the distance between the two points has reached a preset tolerance. The stopping threshold was calculated after various testing and a value was generated which is equal to distance×1.3. The value was determined by multiplying various values with distance versus the stoppage of SRV-1 from the goal point. After in depth testing the threshold value was selected as 1.3.

.

The distance is calculated using Euclidean Distance Formula

$$D = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2} \qquad (5.1)$$

The values of the coordinates of point1 and point 2 are stored in the variables, and the distance is calculated using the above formula. The following figure shows the output, after enabling the tracking, as well as the coordinates of the point and the distance between two points. Again, once the threshold distance is reached, the SRV-1 is commanded to stop.


c:\Users\Vishal\Documents\Visual Studio 2008\Projects\Revised\Debug\Revised.exe
192.168.0.9

```
first 1 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,70]
Tracking Right ... [141,70]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [121,69]
Tracking Right ... [141,70]
first 2 Distance  20.024984 Dstop 26.000000
```

Figure 39: Here you can see that the Correlation Tracking is actively computing the distance between the two points.

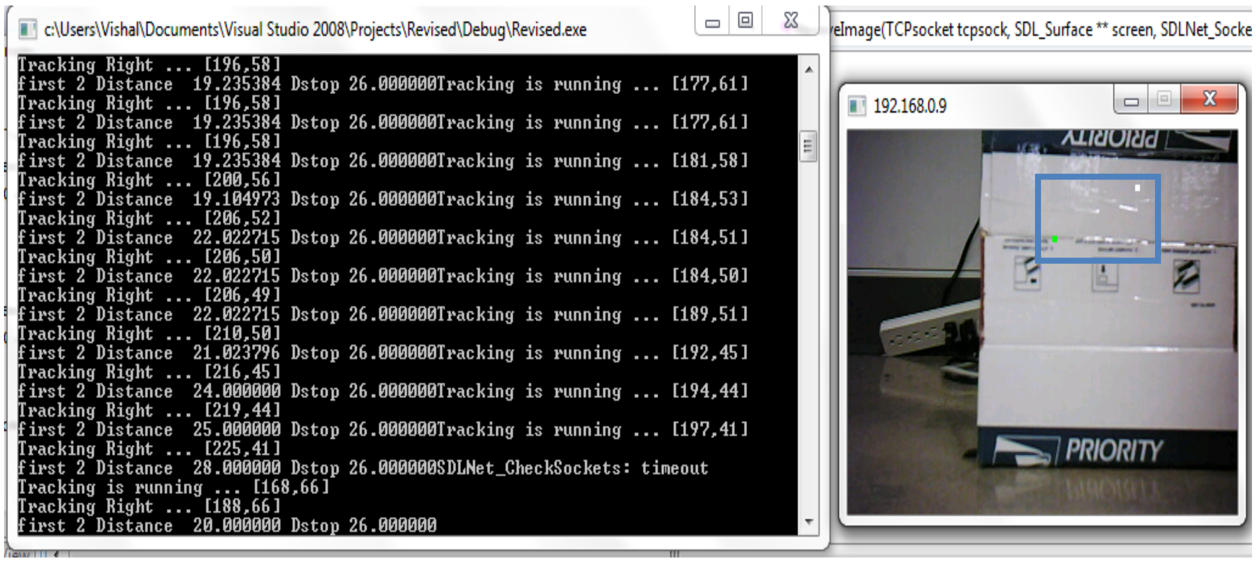
4.2.4 SRV-1 Movement Commands

After setting up the tracking points the next task is to move the robot to the desired location. The commands provided by the Surveyor Company, for the SRV-1 is used for this task. Command “8” is used for moving the robot forward. Conditional statements are generated to first checked to ensure that tracking has been activated. If the tracking is activated and the points are within the pre-specified tolerance, then the appropriate stop command is issued to the SRV-1. If not, the SRV-1 continues to move forward autonomously.

An SDL mouse event is required to sense the mouse in the algorithm and so a trigger was generated for initiating the movement and then an appropriate check loop was generated which checks for the tracking activation and also for the mouse event. If both of these are true it initiates the movement.

Once the SRV-1 starts to move forward it was required to issue a stop command when it is near to the clicked location. Here the previous calculation of the distance was taken into consideration and a stopping threshold is created so a threshold is created. It is compared with the original value of the distance. When the threshold is less than or equal to the distance a stop command could be issued to the robot. Command “5” was used to stop the SRV-1.

The figure below shows the output when the SRV-1 reaches its destination. The command window shows that it is comparing the threshold value with the original distance. The “Dstop” is the threshold value and “Distance” is the original distance calculated from the two points. Once the distance is greater than the Dstop the SRV-1 is stopped. In this window, notice that the calculated distance is 28cm where is the Dstop threshold is 26. This triggers a stop of the SRV-1.

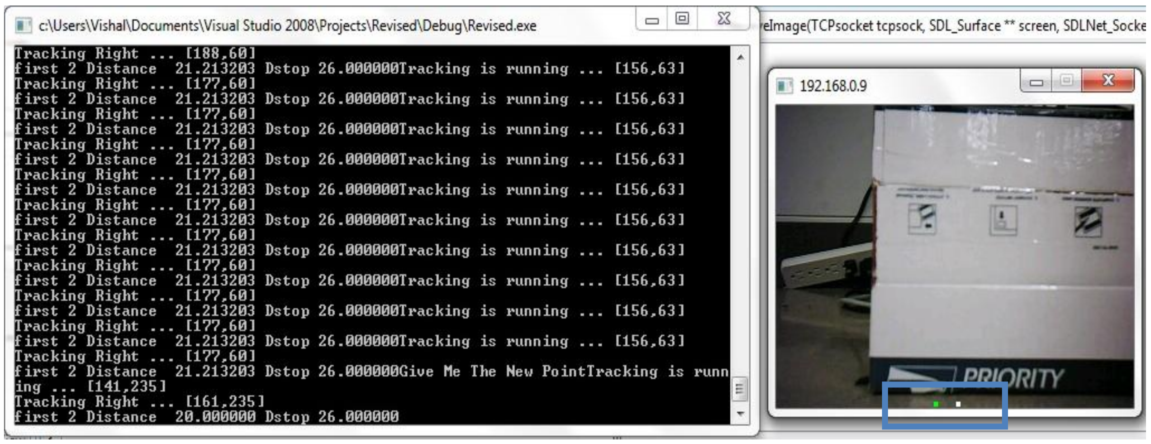

c:\Users\Vishal\Documents\Visual Studio 2008\Projects\Revised\Debug\Revised.exe
eImage(TCPsocket tcpsock, SDL_Surface ** screen, SDLNet_Socke
192.168.0.9
PRIORITY

```
Tracking Right ... [196,58]
first 2 Distance  19.235384 Dstop 26.000000Tracking is running ... [177,61]
Tracking Right ... [196,58]
first 2 Distance  19.235384 Dstop 26.000000Tracking is running ... [177,61]
Tracking Right ... [196,58]
first 2 Distance  19.235384 Dstop 26.000000Tracking is running ... [181,58]
Tracking Right ... [200,56]
first 2 Distance  19.104973 Dstop 26.000000Tracking is running ... [184,53]
Tracking Right ... [206,52]
first 2 Distance  22.022715 Dstop 26.000000Tracking is running ... [184,51]
Tracking Right ... [206,50]
first 2 Distance  22.022715 Dstop 26.000000Tracking is running ... [184,50]
Tracking Right ... [206,49]
first 2 Distance  22.022715 Dstop 26.000000Tracking is running ... [189,51]
Tracking Right ... [210,50]
first 2 Distance  21.023796 Dstop 26.000000Tracking is running ... [192,45]
Tracking Right ... [216,45]
first 2 Distance  24.000000 Dstop 26.000000Tracking is running ... [194,44]
Tracking Right ... [219,44]
first 2 Distance  25.000000 Dstop 26.000000Tracking is running ... [197,41]
Tracking Right ... [225,41]
first 2 Distance  28.000000 Dstop 26.000000SDLNet_CheckSockets: timeout
Tracking is running ... [168,66]
Tracking Right ... [188,66]
first 2 Distance  20.000000 Dstop 26.000000
```

Figure 40: Once the distance between the two points is at a predetermined threshold, the robot automatically stops at the destination.

After the SRV-1 stops, it could be moved with maual commands and assigned to a different location. Tracking could be reset and it could be send to a new location.

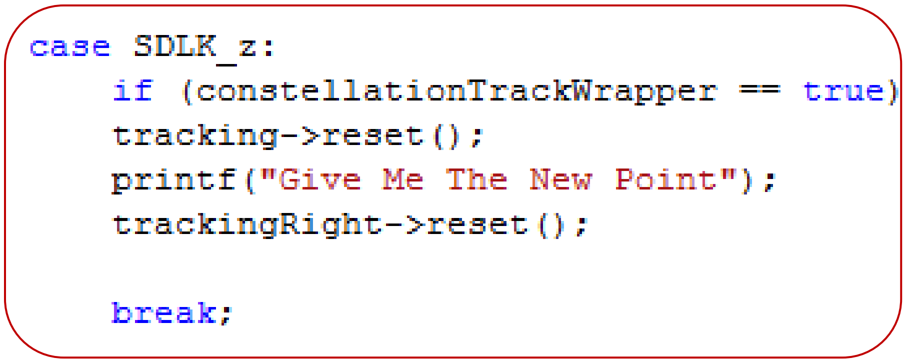
In order to reset the tracking, a “z” key on the keyboard ,was assigned for the resetting of tracking.The user or operator has to press “z” and click once on the new location and again “z” and click on the desired location, to reset both the points to the desired location. When the “z” is pressed, the user is notified on the screen asking for “*Give me the new Point*”. So now, the user has to click on the new location and the tracking points will be set to that location. The following figure shows that the tracking is reset after stopping, which is shown in the previous figure.


c:\Users\Vishal\Documents\Visual Studio 2008\Projects\Revised\Debug\Revised.exe
eImage(TCPsocket tcpsock, SDL_Surface ** screen, SDLNet_Socke
192.168.0.9

```
Tracking Right ... [188,60]
first 2 Distance  21.213203 Dstop 26.000000Tracking is running ... [156,63]
Tracking Right ... [177,60]
first 2 Distance  21.213203 Dstop 26.000000Tracking is running ... [156,63]
Tracking Right ... [177,60]
first 2 Distance  21.213203 Dstop 26.000000Tracking is running ... [156,63]
Tracking Right ... [177,60]
first 2 Distance  21.213203 Dstop 26.000000Tracking is running ... [156,63]
Tracking Right ... [177,60]
first 2 Distance  21.213203 Dstop 26.000000Tracking is running ... [156,63]
Tracking Right ... [177,60]
first 2 Distance  21.213203 Dstop 26.000000Tracking is running ... [156,63]
Tracking Right ... [177,60]
first 2 Distance  21.213203 Dstop 26.000000Tracking is running ... [156,63]
Tracking Right ... [177,60]
first 2 Distance  21.213203 Dstop 26.000000Tracking is running ... [156,63]
Tracking Right ... [177,60]
first 2 Distance  21.213203 Dstop 26.000000Tracking is running ... [156,63]
Tracking Right ... [177,60]
first 2 Distance  21.213203 Dstop 26.000000Tracking is running ... [156,63]
Tracking Right ... [177,60]
first 2 Distance  21.213203 Dstop 26.000000Give Me The New PointTracking is runn
ing ... [141,235]
Tracking Right ... [161,235]
first 2 Distance  20.000000 Dstop 26.000000
```

Figure 41: Now, a new set of tracking points can be selected. Reseting of tracking points.

The following snippet shows how the tracking is reset. The reset is a function which points to the tracker class, as we have two points the tracking must be reset of both the points. The “SDLK_z” is the key event of SDL.

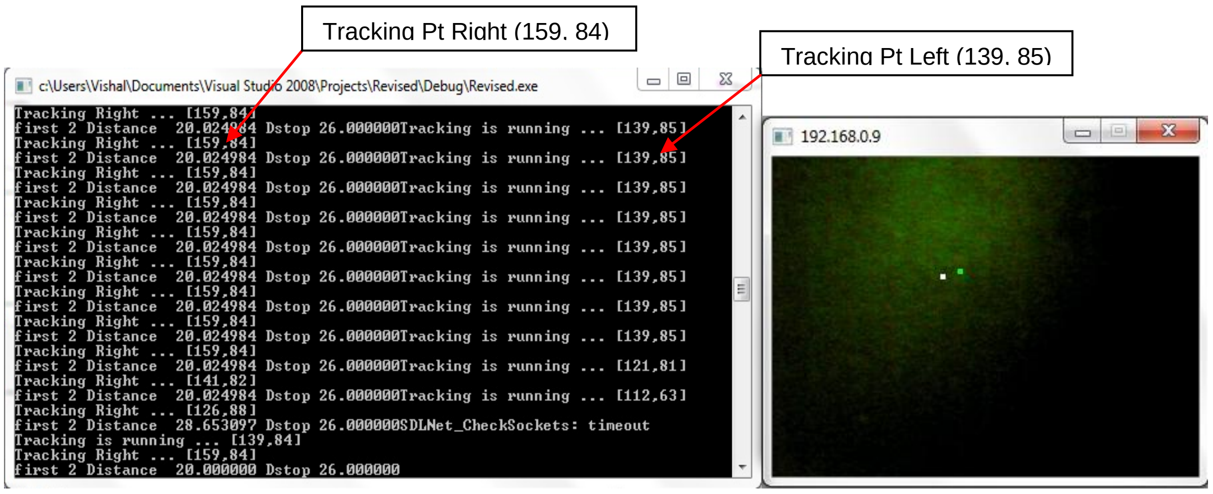


```
case SDLK_z:
     if (constellationTrackWrapper == true)
     tracking->reset();
     printf("Give Me The New Point");
     trackingRight->reset();

     break;
```

4.3 Testing and Result

After the system was created, an in-depth analysis and testing was performed to ascertain the potential problems of the system. This analysis could also lead to system-level efficiencies that could be implemented. During the testing a major problem that was encountered with respect to normal correlation tracking was the lighting condition. Poor lighting conditions drastically affected the performance of the tracking system. In addition, the system was affected by sensitive to scaling and rotation of the objects.

A test was conducted to see whether the correlation tracking reacts according to the light conditions. So the SRV-1 was sent from a bright contrast/ well lit area to a low-contrast dark area to gauge the effects of low lighting.

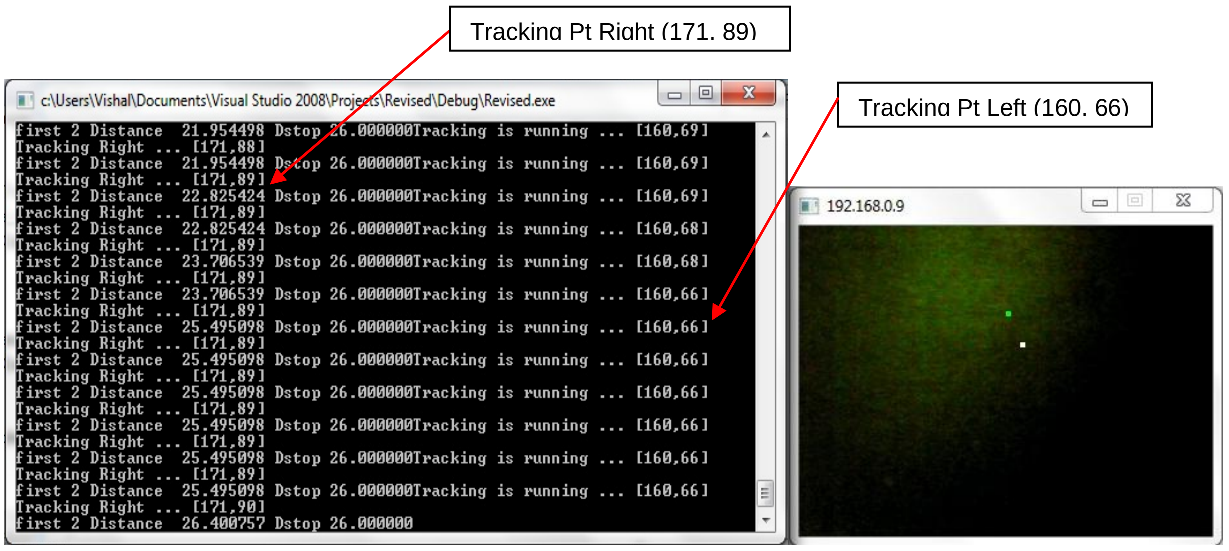

Tracking Pt Right (159, 84)
Tracking Pt Left (139, 85)
c:\Users\Vishal\Documents\Visual Studio 2008\Projects\Revised\Debug\Revised.exe
192.168.0.9

```
Tracking Right ... [159,84]
first 2 Distance  20.024984 Dstop 26.000000Tracking is running ... [139,85]
Tracking Right ... [159,84]
first 2 Distance  20.024984 Dstop 26.000000Tracking is running ... [139,85]
Tracking Right ... [159,84]
first 2 Distance  20.024984 Dstop 26.000000Tracking is running ... [139,85]
Tracking Right ... [159,84]
first 2 Distance  20.024984 Dstop 26.000000Tracking is running ... [139,85]
Tracking Right ... [159,84]
first 2 Distance  20.024984 Dstop 26.000000Tracking is running ... [139,85]
Tracking Right ... [159,84]
first 2 Distance  20.024984 Dstop 26.000000Tracking is running ... [139,85]
Tracking Right ... [159,84]
first 2 Distance  20.024984 Dstop 26.000000Tracking is running ... [139,85]
Tracking Right ... [159,84]
first 2 Distance  20.024984 Dstop 26.000000Tracking is running ... [139,85]
Tracking Right ... [159,84]
first 2 Distance  20.024984 Dstop 26.000000Tracking is running ... [121,81]
Tracking Right ... [141,82]
first 2 Distance  20.024984 Dstop 26.000000Tracking is running ... [112,63]
Tracking Right ... [126,88]
first 2 Distance  28.653097 Dstop 26.000000SDLNet_CheckSockets: timeout
Tracking is running ... [139,84]
Tracking Right ... [159,84]
first 2 Distance  20.000000 Dstop 26.000000
```

Figure 42: Change in tracking point's value.

From the above figure [42] it could be perceive that the tracking was started at the pixel values (139, 85) for the left tracking point and (159, 84) for the right tracking point. It can be concluded from the figure that the tracking points get disperse and changes. Thus rapid fluctuations in the pixel value were encountered. The fluctuations and unstable computation can also be visualized on the video as it is keeps on changing. Note on the figure that the values changed up to (112, 63) for the left tracking point and (126,88) for the right tracking point. The tracking then resets and as the tracker knows the values of the pixel it tried to set them at the same position, but the

SRV-1 was moved further to see the reaction of the low light which is shown in the following figure[42].

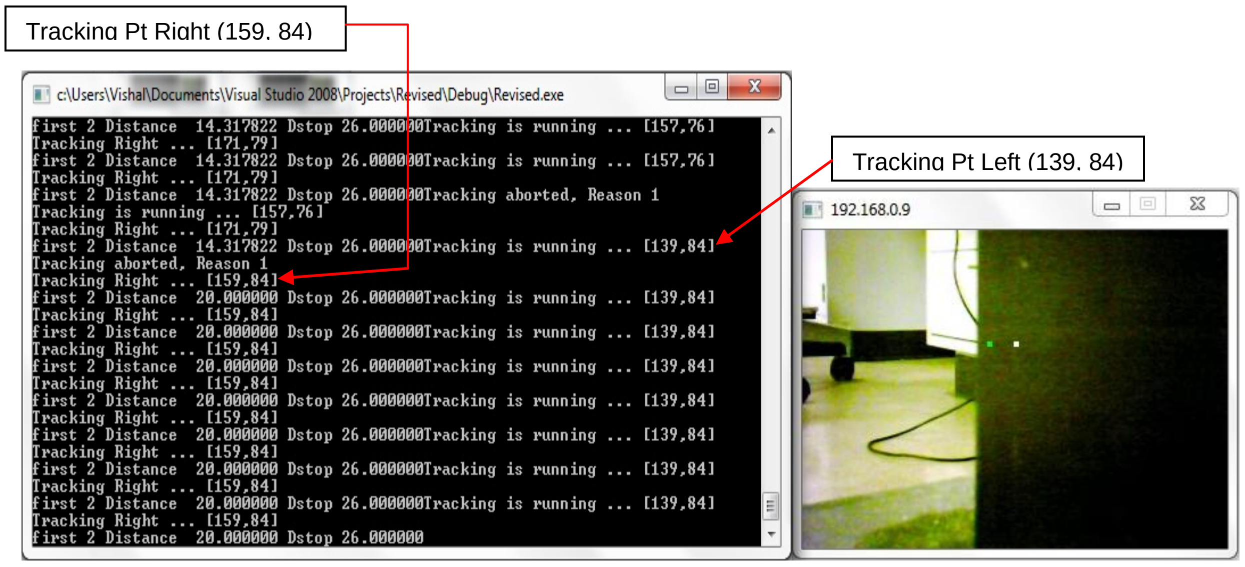

Tracking Pt Right (171, 89)
Tracking Pt Left (160, 66)
c:\Users\Vishal\Documents\Visual Studio 2008\Projects\Revised\Debug\Revised.exe
192.168.0.9

```
first 2 Distance  21.954498 Dstop 26.000000Tracking is running ... [160,69]
Tracking Right ... [171,88]
first 2 Distance  21.954498 Dstop 26.000000Tracking is running ... [160,69]
Tracking Right ... [171,89]
first 2 Distance  22.825424 Dstop 26.000000Tracking is running ... [160,69]
Tracking Right ... [171,89]
first 2 Distance  22.825424 Dstop 26.000000Tracking is running ... [160,68]
Tracking Right ... [171,89]
first 2 Distance  23.706539 Dstop 26.000000Tracking is running ... [160,68]
Tracking Right ... [171,89]
first 2 Distance  23.706539 Dstop 26.000000Tracking is running ... [160,66]
Tracking Right ... [171,89]
first 2 Distance  25.495098 Dstop 26.000000Tracking is running ... [160,66]
Tracking Right ... [171,89]
first 2 Distance  25.495098 Dstop 26.000000Tracking is running ... [160,66]
Tracking Right ... [171,89]
first 2 Distance  25.495098 Dstop 26.000000Tracking is running ... [160,66]
Tracking Right ... [171,89]
first 2 Distance  25.495098 Dstop 26.000000Tracking is running ... [160,66]
Tracking Right ... [171,89]
first 2 Distance  25.495098 Dstop 26.000000Tracking is running ... [160,66]
Tracking Right ... [171,89]
first 2 Distance  25.495098 Dstop 26.000000Tracking is running ... [160,66]
Tracking Right ... [171,90]
first 2 Distance  26.400757 Dstop 26.000000
```

Figure 43: Tracking point changed from the initial point.

The above figure [43] shows the SRV-1 was moved farther darker place to see whether the tracker resets to the normal position, but due to the normal correlation tracking it cannot get back to its starting position and ended at the pixel values (160, 66) for the left tracking point and (171,90) for the right tracking point, which is way different from the initial value ie (139, 85) for the left tracking point and (159, 84) for the right tracking point

To verify that the correlation tracking is working and to gage its efficiency, the SRV-1 was moved from bright region to the left to the low light place.


Tracking Pt Right (159, 84)
c:\Users\Vishal\Documents\Visual Studio 2008\Projects\Revised\Debug\Revised.exe
Tracking Pt Left (139, 84)
192.168.0.9

```
first 2 Distance  14.317822 Dstop 26.000000Tracking is running ... [157,76]
Tracking Right ... [171,79]
first 2 Distance  14.317822 Dstop 26.000000Tracking is running ... [157,76]
Tracking Right ... [171,79]
first 2 Distance  14.317822 Dstop 26.000000Tracking aborted, Reason 1
Tracking is running ... [157,76]
Tracking Right ... [171,79]
first 2 Distance  14.317822 Dstop 26.000000Tracking is running ... [139,84]
Tracking aborted, Reason 1
Tracking Right ... [159,84]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [139,84]
Tracking Right ... [159,84]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [139,84]
Tracking Right ... [159,84]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [139,84]
Tracking Right ... [159,84]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [139,84]
Tracking Right ... [159,84]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [139,84]
Tracking Right ... [159,84]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [139,84]
Tracking Right ... [159,84]
first 2 Distance  20.000000 Dstop 26.000000Tracking is running ... [139,84]
Tracking Right ... [159,84]
first 2 Distance  20.000000 Dstop 26.000000
```

Figure 44: Correlation Tracking Resuming with change in environment.

It is clear from the above figure that as the SRV-1 moves to the stable/good light conditions the tracking starts to work correctly, resetting itself automatically to the previous pixel values (139, 84) for the left tracking point and (159,84) for the right tracking point respectively as shown in figure[42].Thus, it is concluded from the test that the tracking system works when near ideal lighting conditions are present.

4.4 Discussion

After the overall testing of the systems some potential problems were encountered. The correlation tracking does not properly work in low light conditions and it is sensitive to the rotation and scaling of the objects. Scaling of the object means shrinking or stretching of objects that occurs when the robot moves towards objects on different terrains as the objects may stretch or shrink while moving nearer to the object.

This problem could be solved using the customized correlation tracking as per the application and the sensitivity to light, scaling and rotating of objects could be coped up by changing the algorithm as per the requirement.

The sensitivity of the light could be controlled by various methods including (1) interfacing the algorithm with the camera controls so that the scene could be better controlled (2)through an improved algorithm with filtering and image processing that enhances the images, or (3) template matching could be introduce to scale the objects nearby in the scene.

Also the SRV-1 could be enhanced using its stereovision camera capability and also an IMU (Inertial Measurement Unit) on-board. In addition, light sensors could help the system to know the position and different light conditions of the environment thus change certain image parameters dynamically. IMU board could be directly attached with the main board of SRV-1 which could help the operator to know the position of the robot and to estimate the distance required.

CHAPTER 5

CONCLUSION, DISCUSSIONS AND FUTURE WORK

5.1 Conclusion

In broad terms, the research covered in this thesis implemented the novel approach of tracking a ground robot using an aerial vehicle with the assistance of augmented reality. This dissertation has extended the previous work of Lee [17] (a “point and go” using aerial camera view), to provide independence from a static camera. This work also demonstrated the use of a two marker approach that could have a broader impact on applications with multiple robots. In addition, this research also extended the previous work of Hunt [14] (a “point and go” control algorithm using onboard camera view), to support robots without specialized location sensor. This work has shown that, the system can work using only visual servoing, allowing to control the robot to be controlled solely from its camera view. This chapter will also summarize all the aims of this research.

A tracking system was developed for the ground robot using the augmented reality. The AESOP robotic arm is used to emulate an aerial vehicle with a camera, which tracks the ground robot using the camera.

The camera view from the aerial vehicle can be used to direct the robot to a desired location using, a “point and go” control algorithm. A two marker augmented reality system enables the ground robot, to be sent to the selected location even if the camera’s position and orientation changes due to movement of the aerial vehicle.

Furthermore the ground robot can be directed to the desired location using its onboard camera view. Thus complete architecture has been established that allows the ground robot to be controlled through onboard and aerial camera views while being, tracked using the aerial vehicle.

5.2 Future Work

An unmanned aerial vehicle and AR Drone could be used as an aerial vehicle to track the ground robot. The AR Drone is a recent augmented reality enabled Wi-Fi quadrocopter manufactured by Parrot. It has a camera mounted on its bottom that could be used to track the ground robot, and it has the ability to hover in the stable position.

Figure 45: AR Drone, Wi-Fi quadrocopter manufactured by Parrot [22].

In addition marker-less augmented reality tracking could be used which would allow tracking of the robot without an affixed marker. To implement this AR Toolkit could be replaced with AR Toolkit plus which provides this capability and many other. A stereo vision camera could be used for the aerial vehicle or for the SRV-1 to provide a wide field of view as well as the 3D measurements. Thus it could be add on to any system that uses it.

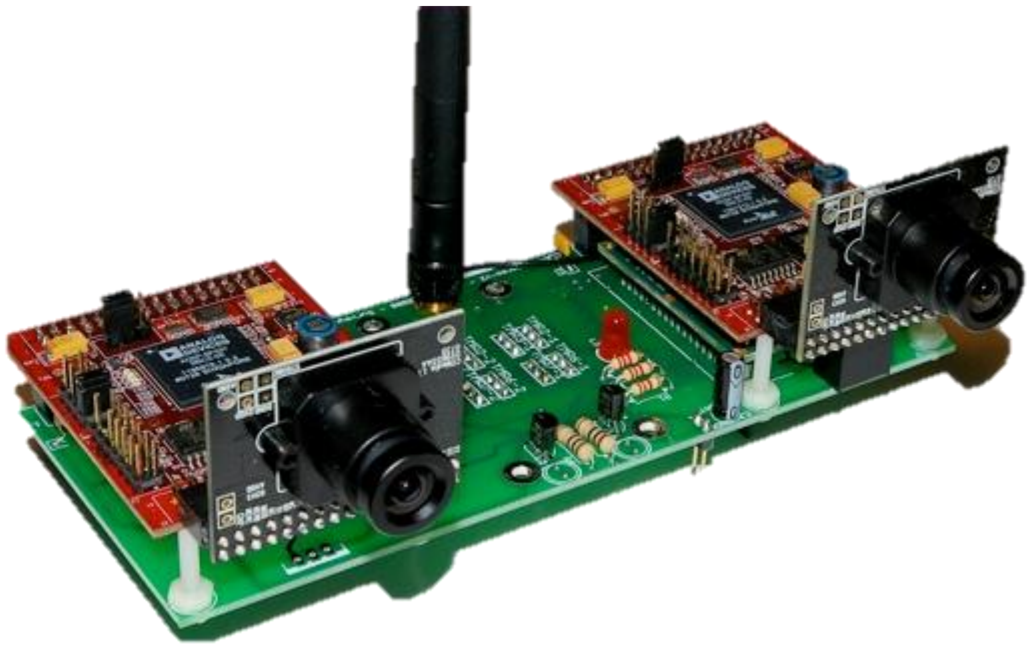


Figure 46: Stereo Vision system for SRV-1 [19].

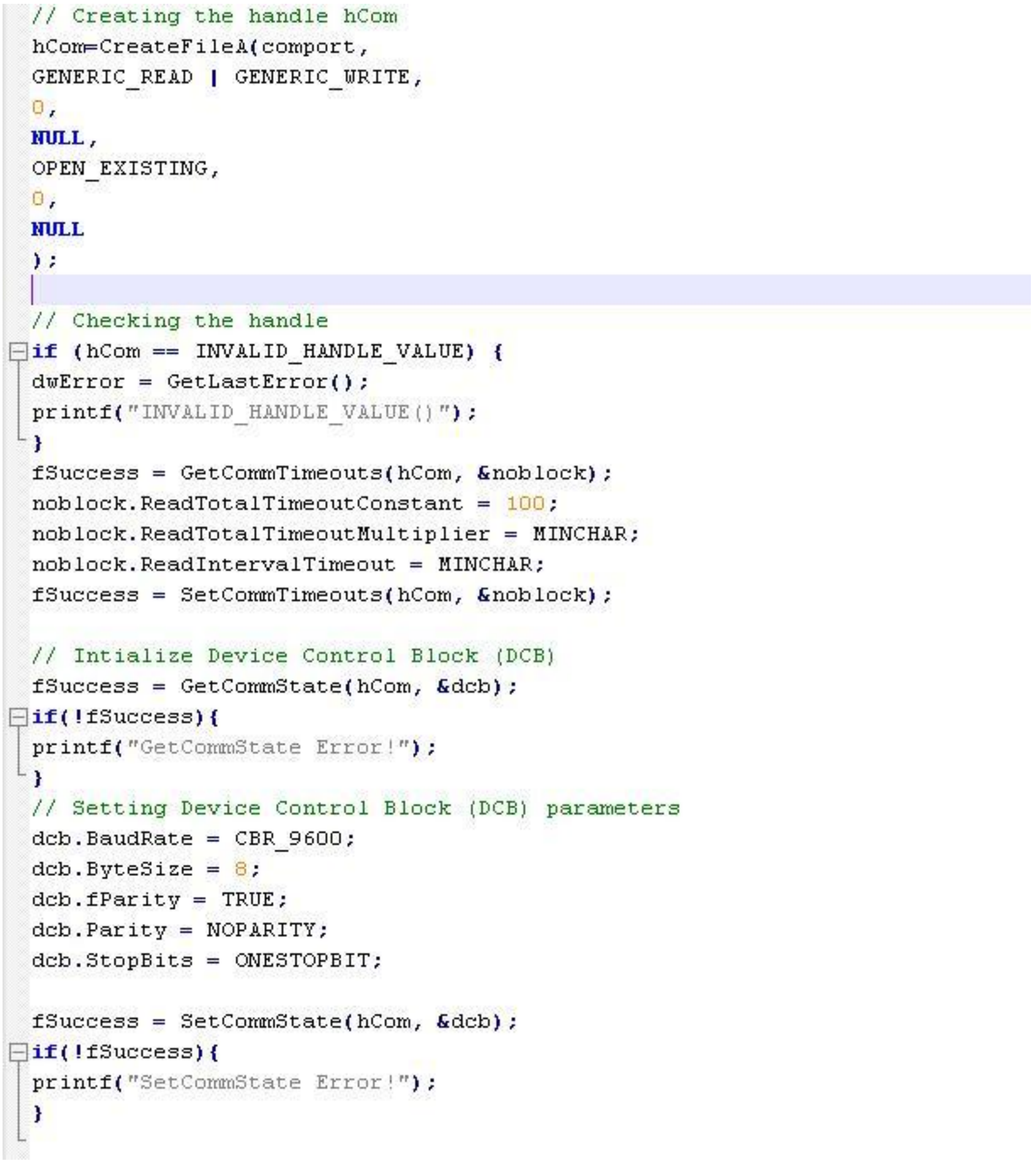
Infrared sensor, proximity sensors or inertial measurement unit (IMU) could be equipped on the SRV-1 to enhance the camera based “point and go” control capabilities. This could help to improve the estimates of the robot’s, its distance from the target and its heading. The data could be compared with the AR derived data to validate the accuracy of the system.

Furthermore the correlation tracking algorithm could be enhanced to make it more resilient to low light and scaling or rotation of the object. Thus it could work better in low light conditions and react better to fast ground vehicle movements. The entire architecture for controlling a single ground vehicle using aerial and onboard camera views could also be used for swarm robotics.

SWARM robotics can be defined as a group of robots performing different tasks collectively. The task may include intelligence or behavior. By this the multiple robots could be commanded to perform the same task. A single robot from the group could be sent to the desired location and rest of them could be deviated to that location using the data from that robot.

Appendix A: C++ SNIPPET FOR SERIAL COMMUNICATION

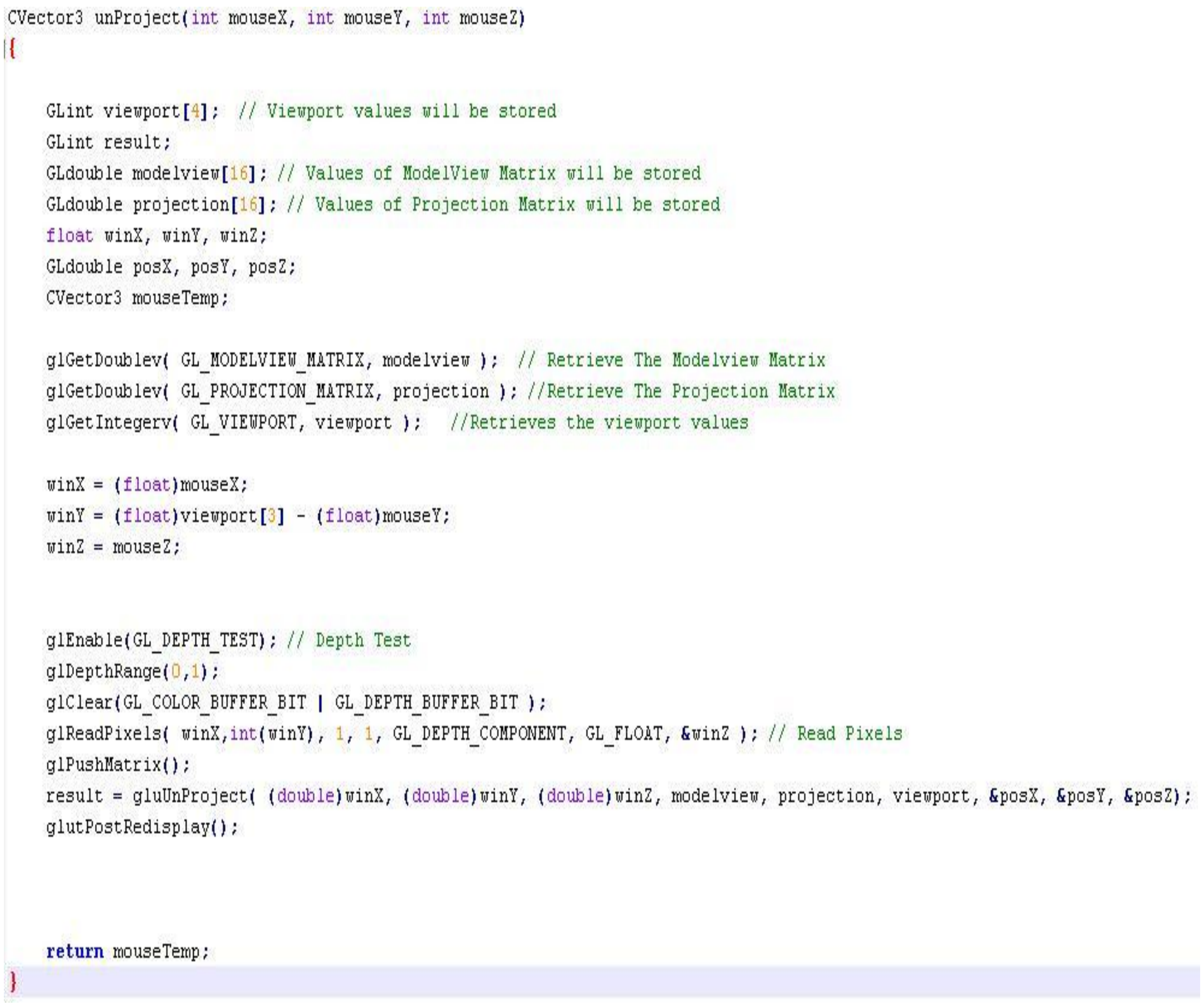
Serial Communication program used for communicating with AESOP 3000 robot.



```
// Creating the handle hCom
hCom=CreateFileA(comport,
GENERIC_READ | GENERIC_WRITE,
0,
NULL,
OPEN_EXISTING,
0,
NULL
);

// Checking the handle
if (hCom == INVALID_HANDLE_VALUE) {
dwError = GetLastError();
printf("INVALID_HANDLE_VALUE()");
}
fSuccess = GetCommTimeouts(hCom, &noblock);
noblock.ReadTotalTimeoutConstant = 100;
noblock.ReadTotalTimeoutMultiplier = MINCHAR;
noblock.ReadIntervalTimeout = MINCHAR;
fSuccess = SetCommTimeouts(hCom, &noblock);

// Intialize Device Control Block (DCB)
fSuccess = GetCommState(hCom, &dcb);
if(!fSuccess){
printf("GetCommState Error!");
}
// Setting Device Control Block (DCB) parameters
dcb.BaudRate = CBR_9600;
dcb.ByteSize = 8;
dcb.fParity = TRUE;
dcb.Parity = NOPARITY;
dcb.StopBits = ONESTOPBIT;

fSuccess = SetCommState(hCom, &dcb);
if(!fSuccess){
printf("SetCommState Error!");
}
```

Appendix B: “GLunProject” FUNCTION

“GlunProject” function for converting screen coordinate to the world coordinate.



```
CVector3 unProject(int mouseX, int mouseY, int mouseZ)
{

    GLint viewport[4];  // Viewport values will be stored
    GLint result;
    GLdouble modelview[16]; // Values of ModelView Matrix will be stored
    GLdouble projection[16]; // Values of Projection Matrix will be stored
    float winX, winY, winZ;
    GLdouble posX, posY, posZ;
    CVector3 mouseTemp;

    glGetDoublev( GL_MODELVIEW_MATRIX, modelview );  // Retrieve The Modelview Matrix
    glGetDoublev( GL_PROJECTION_MATRIX, projection ); //Retrieve The Projection Matrix
    glGetIntegerv( GL_VIEWPORT, viewport );   //Retrieves the viewport values

    winX = (float)mouseX;
    winY = (float)viewport[3] - (float)mouseY;
    winZ = mouseZ;


    glEnable(GL_DEPTH_TEST); // Depth Test
    glDepthRange(0,1);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
    glReadPixels( winX,int(winY), 1, 1, GL_DEPTH_COMPONENT, GL_FLOAT, &winZ ); // Read Pixels
    glPushMatrix();
    result = gluUnProject( (double)winX, (double)winY, (double)winZ, modelview, projection, viewport, &posX, &posY, &posZ);
    glutPostRedisplay();



    return mouseTemp;
}
```

REFERENCES

1. Chen, J.Y.C.; Haas, E.C.; Barnes, M.J.; , "Human Performance Issues and User Interface Design for Teleoperated Robots," Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on , vol.37, no.6, pp.1231-1245, Nov. 2007
2. *“*EyeDrive Multi-Purpose Mini-Robot - Technological Solutions Meeting Counter-Terror Indoor Challenges*”*. Defense Update, n.d. Web. 7 Sep 2010. Available from: http://defense-update.com/features/2008/november/11208_multipurposeminiroboticplatform.html
3. Fong, T. and C. Thorpe (2001). "Vehicle Teleoperation Interfaces." Auton. Robots **11**(1): 9-18.
4. J.Y.C, Chen. *EVALUATION OF STEREOSCOPIC DISPLAYS FOR INDIRECT-VISION DRIVING AND ROBOT TELEOPERATION*. Orlando
5. Ballantyne, G. H. (2002). "Robotic surgery, telerobotic surgery, telepresence, and telementoring - Review of early clinical results." Surgical Endoscopy and Other Interventional Techniques **16**(10): 1389-1402.
6. F. Smith, D. Backman, and S. Jacobsen, “Telerobotic manipulator for hazardous environments,” in *Journ. of Rob. Syst.*, vol. 9, NO. 2, pp. 251–260, 1992.
7. J. P. K. Kim, H. Lee and M. Yang, “Robotic contamination cleaning system,” in *IEEE Conference on Intelligent Robots ans Systems*, Lausanne, Switzerland, October 2002.

8. Azuma, R. T. (1997). A survey of augmented reality. *Presence*, *6*, 355–385.

9. Milgram, P., S. Zhai, et al. (1993). Applications of augmented reality for human-robot communication. Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on.

10. Green, S. A., M. Billinghurst, et al. (2008). "Human-Robot Collaboration: A Literature Review and Augmented Reality Approach In Design." International Journal of Advanced Robotic Systems **5**(1): 1-18.

11. Se, S., D. Lowe, et al. (2001). Vision-based mobile robot localization and mapping using scale-invariant features. Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on.

12. Wolf, J., W. Burgard, et al. (2002). Robust vision-based localization for mobile robots using an image retrieval system based on invariant features. Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on.

13. Espiau, B., F. Chaumette, et al. (1992). "A New Approach to Visual Servoing in Robotics." IEEE Transactions on Robotics and Automation **8**(3): 313-326.

14. Hunt S. “Robotic Goal-Based Semi-Autonomous Algorithms Improve Remote Operator Performance” Phd Dissertation 2010.

15. Morris, Stephen. *Cooperative Tracking of Moving Targets by Teams of Autonomous Unmanned Air Vehicles*. (2005):

16. Ariyur, K. B. and K. O. Fregene (2008). Autonomous tracking of a ground vehicle by a UAV. American Control Conference, 2008.

17. Lee, Sam. Single Operator Virtual Interface Control of Heterogeneous Robots Executing Cooperative Algorithms for Contaminant Localization Phd Prospectus 2010
18. MARKHAM, K.C. "A Multiple Point Correlation Tracker for TIME-TO-GO Estimation." British Aerospace (Dynamics) Ltd, UK,
19. "Surveyor SRV-1 Open Source Mobile Robot. Surveyor", n.d. Web. 4 May 2011. Available from: http://www.surveyor.com/SRV_info.html
20. "*Quickcam* Orbit AF", n.d. Web. 4 May 2011. Available from: http://www.logitech.com/en-us/webcam-communications/webcams/devices/3480.
21. Buss, S. R. (2004). Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods.
22. "AR Drone" Web. 4 May 2011. Available from: http://www.amazon.com/Parrot-AR-Drone-Quadricopter-Controlled-iPhone/dp/B003ZVSHB0
23. "ARToolKit Coordinate Systems." N.p., n.d. Web. 22 Jun 2011. Available from: http://www.hitl.washington.edu/artoolkit/documentation/cs.htm
24. Wood, G., Realities of automatic correlation problem. Photogrammetric Engineering and Remote Sensing, 1983. 49(4): p. 537-538.
25. "OpenCV: Image Processing and Computer Vision Reference Manual" Web. 4 May 2011. Available from: http://www.seas.upenn.edu/~bensapp/opencvdocs/ref/opencvref_cv.htm

ABSTRACT

CONTROLLING OF A SINGLE GROUND VEHICLE USING AERIAL AND ONBOARD CAMERA VIEWS

by

VISHAL LOWALEKAR

August 2011

Advisor: Dr. Abhilash K. Pandya.

Major: Computer Engineering.

Degree: Master of Science.

Controlling a ground vehicle using only onboard and aerial camera views, can be a tedious task, as it is challenging for the operator to estimate information about the robots location and orientation using only these view. However the visual data can be sufficient for computerized calculation of this information even in the absence of other sensors. This approach is of interest to the military because the use of passive sensors instead of active sensors (e.g. LIDAR) would be easier and more reliable and would not give off detectable signatures.

The goal of this research was to control a ground vehicle using onboard and aerial camera views. No other sensors like an Infrared camera or IMU were used to track the robot. In addition an augmented reality-based tracking capability was developed for the ground robot using an aerial vehicle. Analysis and efficiency testing was conducted on the implemented tracking and control algorithms demonstrating that

the ground vehicle can be tracked by the aerial robot using augmented reality. Moreover the augmented reality control system enabled the ground vehicle to be accurately directed to different locations using the onboard and aerial camera views. Although the research used a specific robotic platform the Surveyor SRV1, the concept can be implemented on any other ground vehicle or potentially be applied to a swarm of robots.

AUTOBIOGRAPHICAL STATEMENT

VISHAL LOWALEKAR

My life began in the famous city of Vadodara, which is located at the western part of India and also famous as a kingdom of Shri. SayajiRao Gaekwad, well known for his work. I finished my entire school in Vadodara (Guj.) and received a Bachelor’s degree in Information Technology [I.T.] from Punjab Technical University, Jalandhar.

I was fascinated with the robots especially humanoids. So I decided to pursue Master’s degree in Computer Engineering field at Wayne State University, where I can use my previous knowledge of the programming and could work on the hardware platform. I had an opportunity to work in SSIM-CARES lab as a graduate researcher. I also worked as a student assistant in Breast Cancer Robot Project of Angott Medical Product. I had also worked as a team member in Intelligent Ground Vehicle Competition (IGVC) from Wayne State University. While pursuing this research, I got an opportunity to work as a software developer intern at Emergent Systems Corporation. I feel highly privileged for all the opportunities I have had in my life.