April 2014

Intelligent Vision-Driven Robot For Sample Detection And Return

Peng Ren
Worcester Polytechnic Institute

Tianci Zhao
Worcester Polytechnic Institute

MXC-380

Intelligent Vision-Driven Robot For Sample Detection And Return

-A Study in Artificial Intelligence

A Major Qualifying Project Report:

Submitted to the faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

by:

Tianci Zhao (RBE)

Peng Ren (CS)

Approved:

Michael J. Ciaraldi

Abstract

This project explores various vision methodologies to locate and return a user-specified object. The project involves building an automatic robotic unit with an all-terrain chassis vehicle and integrated camera. The high level vision control system uses serial communication to direct the low level mechanical parts. The chosen approach for vision analysis is comparison of color thresholds. This solution provides generally accurate detection even in an environment which is noisy but has good color contrast.

Table of Contents

Table of Figures

Chapter 1: Introduction to the project

During recent years, there is an increasing need for autonomous robots in research and military areas. For tasks such as sampling and investigating, objects and videos need to be taken while moving for analysis purposes. Often times it is inconvenient or impossible for humans to finish such tasks due to environmental restrictions. A feasible solution to this problem is a device able to autonomously find a path to the desired location with the capability of taking images and videos, identifying and carrying the desired objects, and responding to various commands given by a human. Such a device could be used in an extreme environment like Mars to collect and return samples for scientific analysis, or it could be used in warfare for investigation purposes.

Project Goal

The goal for this project is to create a four wheel mobile robot that would be able to autonomously find a path to the desired location by using an efficient computer vision method. The robot should have a robust mechanical support that allows it to travel through rough terrain. The robot also needs to have accurate detection and a user-friendly interface program. Ideally the robot would be able to recognize the desired item using the camera, retrieve the object, and navigate a path to the destination. The robot needs to be able to switch between autonomous finding and teleportation control, and send video back to the user while traveling.

This goal was broken down into various objectives. First a graphic user interface needed to be created to give the user the option to control the robot remotely. Video analyzing is very important; to accomplish autonomous path finding the robot must be able to recognize the item and drive toward it. After these two objectives are accomplished, the resulting functions can be merged into the next objective, which is to have the robot find and return the sample autonomously.

Report organization

The organization of the report is as follows. In Chapter 2, background information is presented on the research topic. Relevant external study is discussed, which includes a brief history of robots, introduction to the Mars Sample Return Mission, the NASA-WPI Sample Return Robot Challenge, and a summary of the history of computer vision. Chapter 3 introduces hardware and software used in the project. Chapter 4 expresses the methodology of how the project was implemented to fulfill the requirements. Chapter 5 illustrates the results of the system design, including testing of the final product. Conclusions are drawn and a summary of the project is given in Chapter 6. Possible improvements and future work are given in Chapter 7.

Chapter 2: Background Research

History of Robots

Although robotics are considered a cutting-edge technology in the 21st century, the basic idea of robots has occupied the human mind for a long time before the technology had been developed. Novelists and movie directors have dreamed and written about stories of robotic technology, and many of these science fiction stories have become classics in literature. One example is *Frankenstein,* which is a fiction novel about an artificial man who goes out of control and become a monster. Other examples such as the movie *The Day the Earth Stood Still* in the 1950s, and the movie *WALL.E* in 2008 both show a robot as advanced saviors for human. So what is a robot? The Robot Institute of America defines a robot as “A reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through various programmed motions for the performance of a variety of tasks.”

The earliest robots were built in the early 1950s by George C. Devol, an inventor from Louisville, Kentucky. He invented and patented a reprogrammable manipulator called "Unimate," from "Universal Automation."[12] Devol attempted to sell his robots to industry, but did not find success. In the late 1960s, Joseph Engleberger, a American businessman and engineer, acquired Devol's robot patent and modified it into an industrial robot; he formed a company called Unimation to produce and market the robots. [12] For his successes, Engleberger is considered the **father** of the modern **robotics** industry.

As computer technology became more advanced and affordable, scientists became more interested in developing robotic technology that could eventually create robots which think as humans do. Today, robotic applications are used in various fields such as military, medical, space exploration, industry, research task and police work. They are used to perform tasks that are undesirable or too dangerous for humans, such as repetitive assembly work or bomb disposal. Estimated by the International Federation of Robotics, the number of industrial robots world-wide will reach over 1,659,500 units by the end of 2016. [13]

Mars Sample Return Mission

Mars sample return has been studied as back as far as the 1960s by the National Aeronautics and Space Administration. The idea of the mission was to send a robot to Mars for the purpose of collecting and bringing back to Earth samples of rock, soil and dust from Mars. A Mars sample return mission would give scientists the capability to explore, analyze and study Mars to a high level of detail in Earth laboratories. The mission would help to develop new technologies since there are always new things to be studied during the development phase. It would also allow us to explore for signs of life on Mars. Until now, several missions have been planned but none of the proposed missions have gone beyond the planning phase. The last three Mars sample returns proposed are a NASA-ESA proposal (Mars 2020 rover mission), a Russian proposal (Mars-Grunt), and a Chinese proposal [1].

In the year 2012, NASA's Mars Science Laboratory successfully sent the rover Curiosity to the surface of Mars and the rover began sending back images immediately after it landed safely. From images Curiosity sent back, there is evidence of an ancient streambed where water once flowed knee-deep [2], which could be a sign of life in the past. The rover Curiosity is a car size rover; it can travel up to 300 feet per hour on its six-wheeled rocker-bogie system. Curiosity also has 17 cameras [3], which gives it the capability to capture stunning image of the red planet. A picture of the rover could be seen in Figure 2.1, which shows the front view of Curiosity.

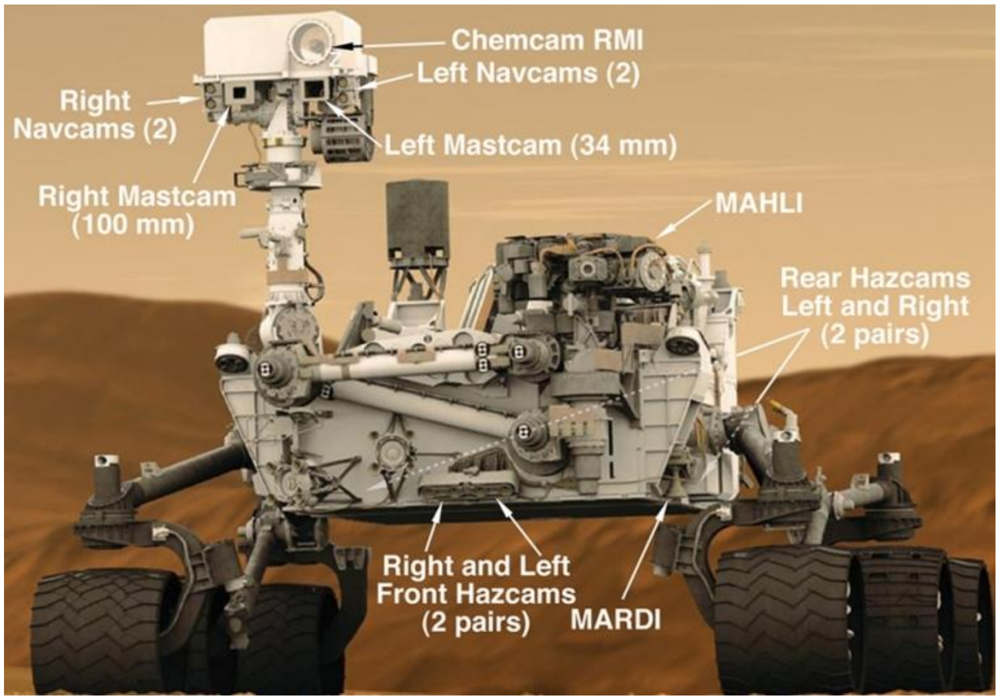

Figure 2 - 1: Curiosity [3]

In order to go further in investigating the ancient environment, past habitability and geological processes on Mars, NASA has announced the Mars 2020 rover mission in December 2012. The Mars 2020 rover mission is a Mars planetary rover mission study by NASA with a possible launch in 2020 [4]. The mission is not only designed to accomplish the science research goal but also be an important step to prepare sending humans to Mars in the 2030s. The rover’s design for the Mars 2020 rover mission will be driven from the Curiosity rover, which landed on Mars in August 2012. The entire rover chassis would be essentially same as the Curiosity rover, which reduces overall technical risk and saves time and funds for the project. The basic structure given by NASA could be seen in Figure 2.2.

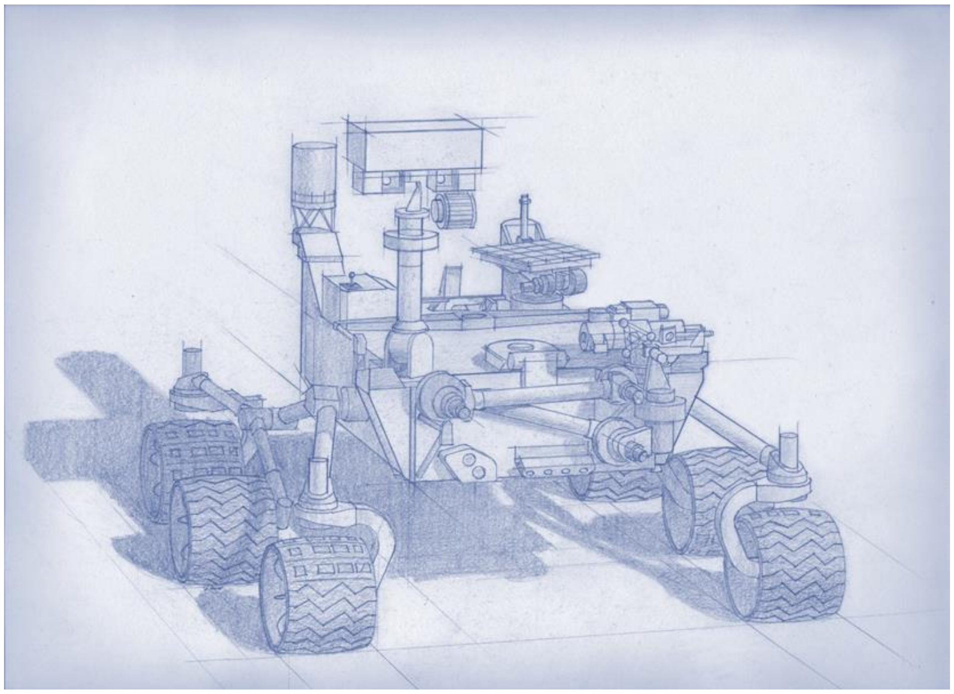

Figure 2 - 2: Basic Concept of Mars 2020 Rover [5]

NASA-WPI Sample Return Robot Challenge

In order to find innovative solutions in autonomous navigation and robotics technologies for NASA's space program, the NASA-WPI Centennial Challenge has been held since 2012. The challenge required each team to create a robot that can locate and collect geologic samples from varied terrain, operating without human control [6]. The first running of the Sample return Robot Challenge took place in 2012, at WPI, where six team showed up to the competition but none of them succeeded. The second competition was held in 2013, at WPI. Ten teams competed for level one, which required the robot to leave the platform, retrieve a sample and return to the platform within the 15-minute limit [7]. There was one team which successfully completed level one, but no teams advanced to level 2. Figure 2.3 shows a robot used in the previous year NASA-WPI Sample Return Robot Challenge.

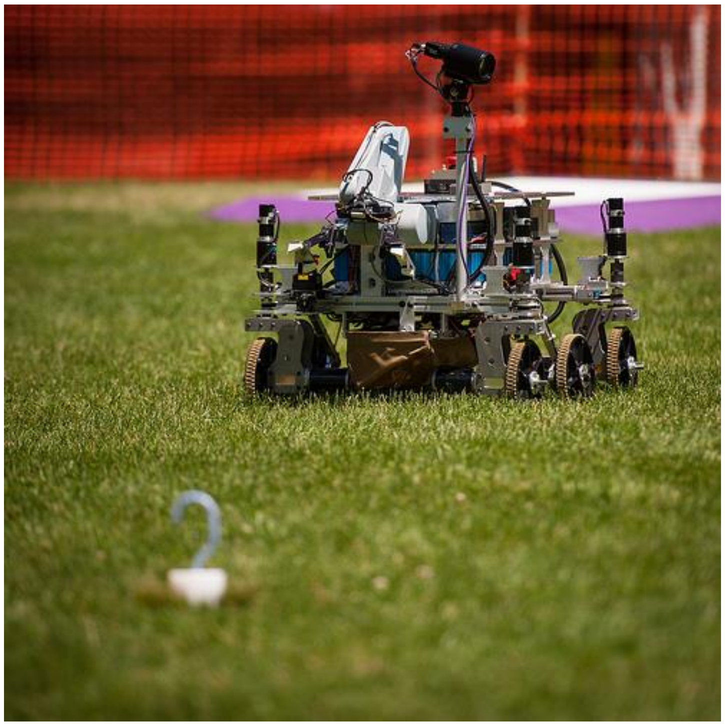

Figure 2 - 3: Robot used in the 2012 NASA-WPI Sample Return Robot Challenge

Computer Vision

Computer vision is the ability of computers to see. From the engineering point of view, computer vision aims to build recognition systems which could mimic or even surpass some of the tasks which the human visual system can perform. Computer vision first started in the 1960s at MIT as a graduate thesis, which discussed the possibilities of extracting 3D geometrical information from 2D perspective views of blocks. [8] Many researchers followed this work and interpreted computer vision in synthetic worlds. Later in 1970s, scientists realized the importance of analyzing images from the real world. Thus, research was focused on interpreting selected images such as edge detecting and segmentation. In the 1980s, computer vision technology shifted toward geometry and increased mathematical rigor. In the 1990s, face recognition technology came out, which is widely used in security and many other areas. [9] In recent years, computer vision technology has grown at a fast rate. Large annotated datasets are available for analyzing and video processing has also started. This is now being applied to robot vision.

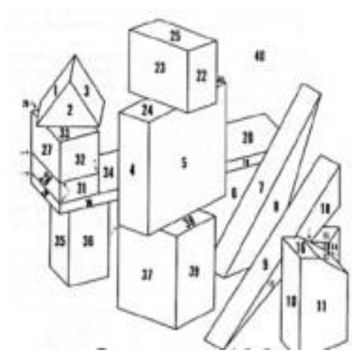

Figure 2 - 4: Guzman'68 [9]

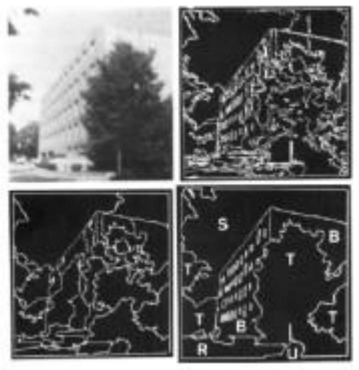

Figure 2 - 5: Ohta Kanade'78[9]

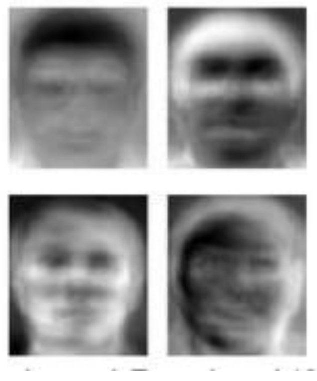

Figure 2 - 6: Turk and Pentland'91[9]

Nowadays, computer vision has become part of our daily life. Computer vision can be used in the medical field to support medical research by image data, which provides various information such as blood flow, the structure of an organ, or assessing the quality of medical treatments. It is also largely used in industry to support manufacturing processes. Tasks such as measuring the position and orientation of an object to be picked up by robot arm and inspecting the final product in order to find defects are all done by computer vision. Military applications are one of the largest areas for computer vision. One example is to use computer vision for missile guidance, which uses locally acquired image data to reduce complexity and to fuse information from multiple sensors to increase reliability. [10]

One of the upcoming fields is autonomous vehicles, which includes aerial vehicles, land-based vehicles, and unmanned aerial vehicles (UAV). Autonomous vehicle technologies can be classified by the level of automation, ranging from using a computer vision based system to support a driver, to more autonomous vehicles, and fully driverless vehicles. Fully autonomous vehicles typically use computer vision for navigation. One example is the Turtlebot robot used in the course RBE 3002: UNIFIED ROBOTICS IV, where the robot sensor data and computer vision technology are used to produce a map of its environment (SLAM) and detect obstacles. In 2012, Google has developed the Google driverless car, where the car takes several maps generated by a range finder mounted on the vehicle and combines the generated map with maps of the world to produce different types of data models that allow it to drive itself. [11] Autonomous vehicles have already being used in space exploration, such as ESA's ExoMars Rover and NASA's Mars Exploration Rover.

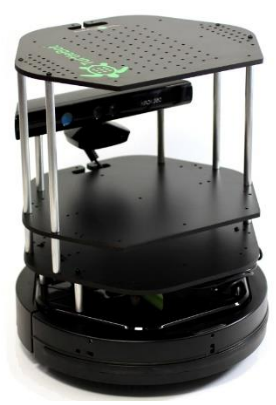

Figure 2 - 7: Turtlebot 2

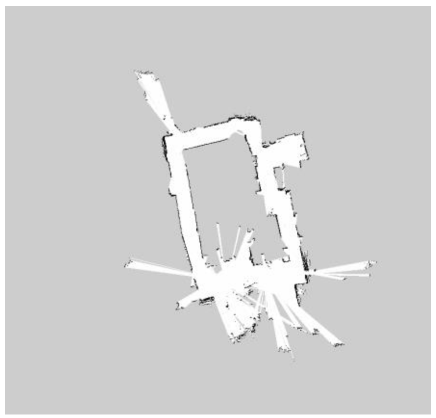

Figure 2 - 8: map generated by Turtlebot 2 by using gmapping

Chapter 3: Methodology

For this project, a robot had to be constructed that had enough power to grab and move the aimed object as well as to power the necessary surveillance equipment. After discussion, we decided that the size of the robot should be relatively small because it would require less power and construction expense. The robot needed to use a computer vision method to navigate a path to the destination, without using encoders or sensors. For testing purposes, the robot needed to be able to run for at least an hour before recharging the battery.

Robot Hardware

Several different solutions were considered in the process of designing this robot; the primary concerns were the ease of construction and time consumption since the team only contains two team members. In this section, robot hardware, including chassis, robot controller and camera, will be introduced. The implementation of this hardware will be introduced in the next chapter.

Robot Chassis

The first choice for the team was to make a custom six wheel rover. The design focused on modularity in order to easily keep a consistent design for the base while still being able to add specialized features depending on the terrain that the rover will have to traverse. A rigid chassis would initially be used, but suspension could easily be added to the rover if testing proved it would be helpful. The top plate would have a grid of tapped holes to allow for easy interfacing of other components such as the arm, camera boom, and sample storage container. It was later decided that the design would be time consuming, and using plastic would make it susceptible to breakage in an outdoor environment.

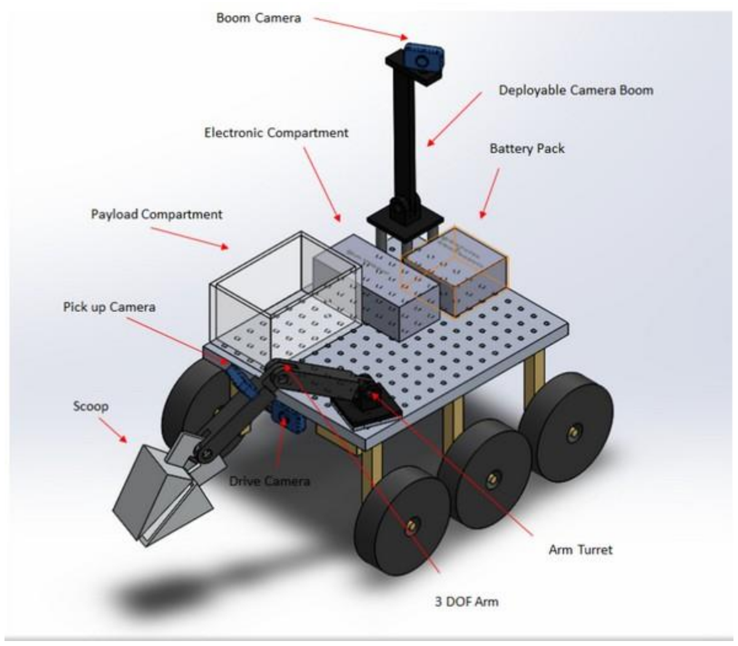

Figure 3 - 1: A representative model of the first rover design

Later in the term, the team found a four wheel drive chassis from Dagu Electronics which was at a reasonable price and decided to purchase it. The rover is designed to drive on rough terrain, which makes it a great platform for robot that needs to perform tasks in an outdoor environment. The size of the robot is 11 by 12 by 5 inches, which was big enough to carry all the necessary equipment. The movement of the robot was provided by four DC motors with brass brushes and 75:1 steel gearboxes that drive 120mm diameter spiked tires. These motors are designed to run between 2V and 7.2V. A rechargeable 7.2 volt Nickel-Metal Hydride battery pack would supply power to these motors. When powered at 7.2v, the chassis can reach a top speed around 3 kilometers per hour. The stall torque of each motor is roughly around 11kg-cm, which is far from powerful. The robot also has a spring suspension system which keeps each wheel in contact with the ground for maximum traction. The chassis is made from a 2mm thick corrosion-resistant anodized aluminum plate; nuts and bolts are made from stainless steel, which made the chassis robust enough for all terrain. Specifications of the robot would be seen in the Appendix, Table 1.

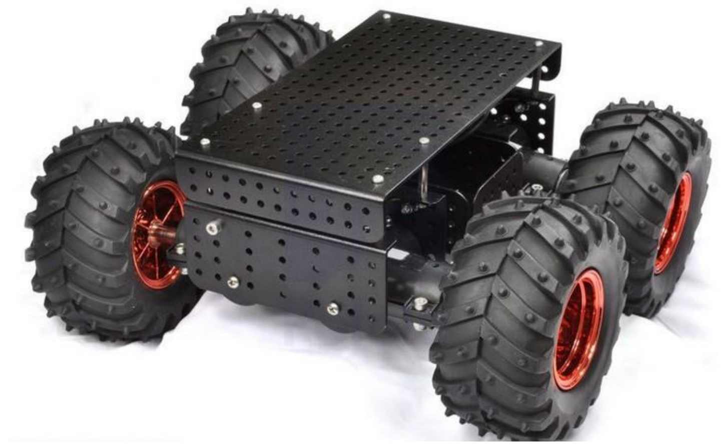

Figure 3 - 2: Dagu Wild Thumper 4WD All-Terrain Chassis

Robot Controller

There were primarily two choices for onboard processor: the T'ReX Robot/Motor Controller or the Raspberry Pi Controller. The team first considered the Raspberry Pi controller. This controller was served with a reasonable price and is programming compatible with Arduino IDE. After more researching, the team found that the Raspberry Pi Controller would not be powerful enough for this project. Later in the term, the team found the T'ReX Robot/Motor Controller and decided to use it since the portability, compactness and power were all desirable.

The T'ReX Robot/Motor Controller is an Arduino compatible robot controller that allows up to 2 analog input, and 4 digital input devices to be controlled from a single USB port. The controller also has motor input and output, which allows motors to be directly connected to it; the motor drivers can supply up to 40 A. A Bluetooth module could also be connected to it to achieve wireless control. Attached to this controller are the following:

One servo mounted on the bottom of the robot arm to raise the arm, one servo mounted on the end of the robot arm to control the gripper, and four DC motors which provided movement to the robot. The specification of T'ReX Robot/Motor Controller can be seen in the Appendix, Table 2 and 3.

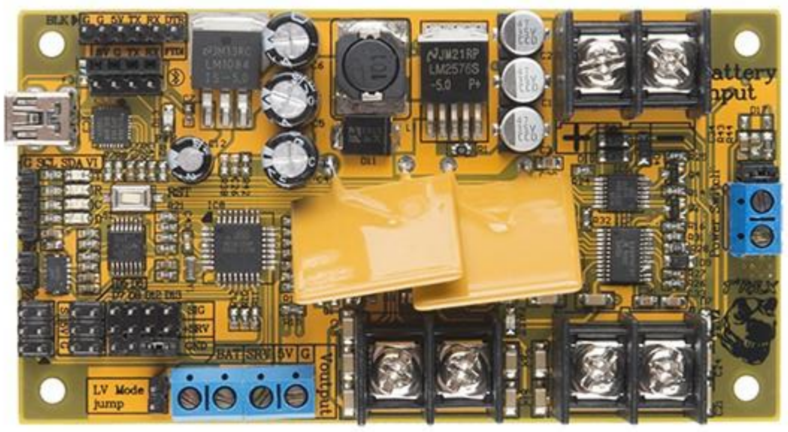

Figure 3 - 3: T'ReX Robot/Motor Controller

Camera

This robot also needed an "eye" that would allow it to recognize the world around it and be able to detect the desired object. Three different cameras were used in this project in order to achieve better performance. The first camera used by the team was the Axis M1033-W Network Camera. This camera used a Wi-Fi connection which allowed the team to achieve wireless control. However, the video quality was heavily dependent on the network speed. So the team tried the second camera, which was the Logitech WebCam C610 HD. A USB cable was used to connect between camera and computer. The detection range of the robot was limited since the camera required a wire for the connection. The last camera the team tried out is Pixy, also known as CMUcam5. It is a fast vision sensor that the user can quickly "teach" to find objects, and it can connect directly to an Arduino-compatible controller. Since Pixy does vision processing itself, this reduces the load on the other computers in the system. However, Pixy is a brand-new product, and our Pixy only arrived during mid-April. The team did not have enough time to change the program design, so we used the Logitech WebCam C610 HD instead.

Figure 3 - 4: Axis M1033-W Network Camera, Logitech WebCam C610 HD

Robot Software

Programming Language

Our team decided to use the Java programming language as our primary programming interface to build the software. Java is a very powerful language, which promotes portable applications. Specifically, Java offers different types of portability: the source code portability, the CPU architecture portability and the OS and GUI portability. The portability is very important to our project since robotic research always involves programming on several different platforms. A software system with high portability enables it to be exported from one platform to another. For instance, when we have finished the software system under Windows platform and want to import it to the Linux, and maybe integrate with the ROS framework, we could probably do it without rewriting everything for Linux, due to the high portability of the Java language.

Another important reason for our team to choose Java is that it is an object-oriented language with concurrent safety. The object oriented feature could make our code more reusable and make the code more maintainable. Also, by taking advantage of design patterns, we could easily make the coding more flexible and sustainable. With the object oriented design, we could easily add new modules to the existing software without modifying large amounts of code.

Embedded Programming and Communication Toolkit

The low level robot control software is a small embedded software system on the T'ReX microcontroller. Since the controller supports the Processing language interface, we choose the Arduino IDE to perform the software development. Also, the sample code of the controller provided by the manufacturer is written in Arduino; therefore it is convenient if we choose Arduino to do the task.

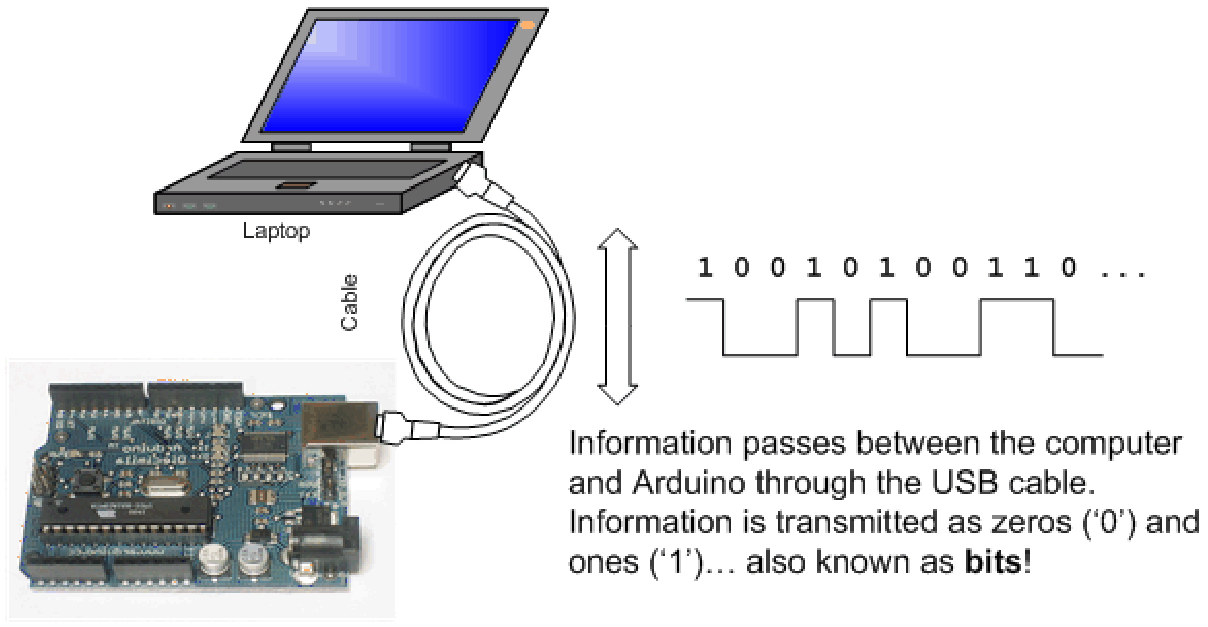
Laptop
Cable
1 0 0 1 0 1 0 0 1 1 0 . . .
Information passes between the computer and Arduino through the USB cable. Information is transmitted as zeros ('0') and ones ('1')… also known as **bits**!

Figure 3 - 5: Axis M1033-W Network Camera, Logitech WebCam C610 HD

In the software system, the high level part of the software on the laptop needs to communicate with the robot connected to a serial port. The serial port communication is usually connected through the USB cable such as shown in figure 3.5. To make sure the high level part could easily communicate with the low level Arduino code, we decided to use the RXTXcomm library. With RXTX, our team could access serial ports through the RXTXcomm.jar file. Along with the jar file, the pre-compiled binaries are readily available to provide the interface to the operating system's virtual UART ports. With the help of the RXTX, the amount of work to establish the serial communication is greatly reduced, so that we could focus our attention on the vision analysis programming. [15]

Vision Software

Since our team concentrated on computer vision to solve the problem, having suitable vision software is critical for the team to achieve success. There are many vision software systems already in the market for both commercial and educational usage, and all of them have their own advantages and disadvantages. After researching along with consultation from the WPI Sample Return Challenge team, our team decided to choose OpenCV as the primary computer vision tool to complete the task.

Figure 3 - 6: OpenCV Logo

Before heading to details about the vision analysis and algorithms, we will first briefly introduce OpenCV. OpenCV, short for Open Source Computer Vision, is a project launched in 1999 by the Intel Corporation. It was first announced as an optimization research aiming for advancing CPU-intensive application, part of a series of projects including real-time ray tracking and 3D display walls [14].

The initial public release of OpenCV was in 2000 at the IEEE Conference on Computer Vision and Pattern Recognition. Until 2005, there were another five beta versions released. However, the first major version of OpenCV was released in 2006, labeled as the 1.0 version of OpenCV. The second primary release, i.e. OpenCV 2, was on October 2009, which contained major changes to the C++ interface. The official release of OpenCV now occurs about every 6 months and the OpenCV has already been taken over by OpenCV.org, a non-profit organization.

The reasons for our team choosing OpenCV are:

1. OpenCV is released under a BSD license so that it is free for both commercial and educational use. Our project is an academic research focusing on the reusable and improvable solution to a real world problem. The limited budget suggests that open source software should be our first choice. Moreover, by choosing an open source software as our toolkit, it is easy for the successor teams to improve or add modules built on our project.
2. OpenCV is a multi-platform software system and supports different programming language interfaces. OpenCV has C and C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. The powerful portability of OpenCV offers us a good way to integrate our software system to different platforms. And our team could free to choose the programming language to develop the vision system.
3. OpenCV is open source and has more than 7 million downloads by users. OpenCV is still in development and has a user community of about 47 thousand people. The large user pool guarantees that we will always have technical support from the official software community. Moreover, it is easy for us to find resources and tutorials if our team encounters any problems in the process of development.

Chapter 4: Implementation

Physical Construction

Structure

The structural components of the robot are made up mainly of pieces from a Dagu Wild Thumper kit. These pieces are held together with bolts and hex nuts which are also part of the kit. On top of this frame rides the T'ReX controller which provides processing power for the gripper and the arm. The movement of the robot is driven by four Dagu Wild Thumper wheels. Each wheel is directly driven by a DC motor which is built inside the frame. The four DC motors are powered by a rechargeable 7.2 volt Nickel-Metal Hydride battery pack. The first camera (Axis M1033-W Network Camera) the team used is mounted directly to the frame with 6-32 machine screws and nuts. The camera is mounted in the center of the top frame and it is powered by the T'ReX controller through a custom made power cable. Later in the term, the team changed to using the Logitech WebCam C610 HD. The team used pieces from a Vex robotics kit to build a stand to provide enough height for the camera to "see" the ground and then used a zip tie to secure the camera on the stand.

Custom Components

Since the robot used several different units that were not designed to work together, some custom components were made to adapt from one system to another. The custom pieces were the following: a connector that connects two rechargeable 7.2 volt Nickel-Metal Hydride batteries, and a power cable that connect between the Axis M1033-W Network Camera and T'ReX motor controller.

The battery used to provide power to the robot was the standard rechargeable 7.2 volt Nickel-Metal Hydride battery. The voltage of the battery was within the lower end of the acceptable range of the motor controller, but the controller board may be damaged due to the low voltage. Thus, a connector was needed to connect two batteries together in series. A positive end of a battery and a negative end of the other battery are wired together by using wire nuts.

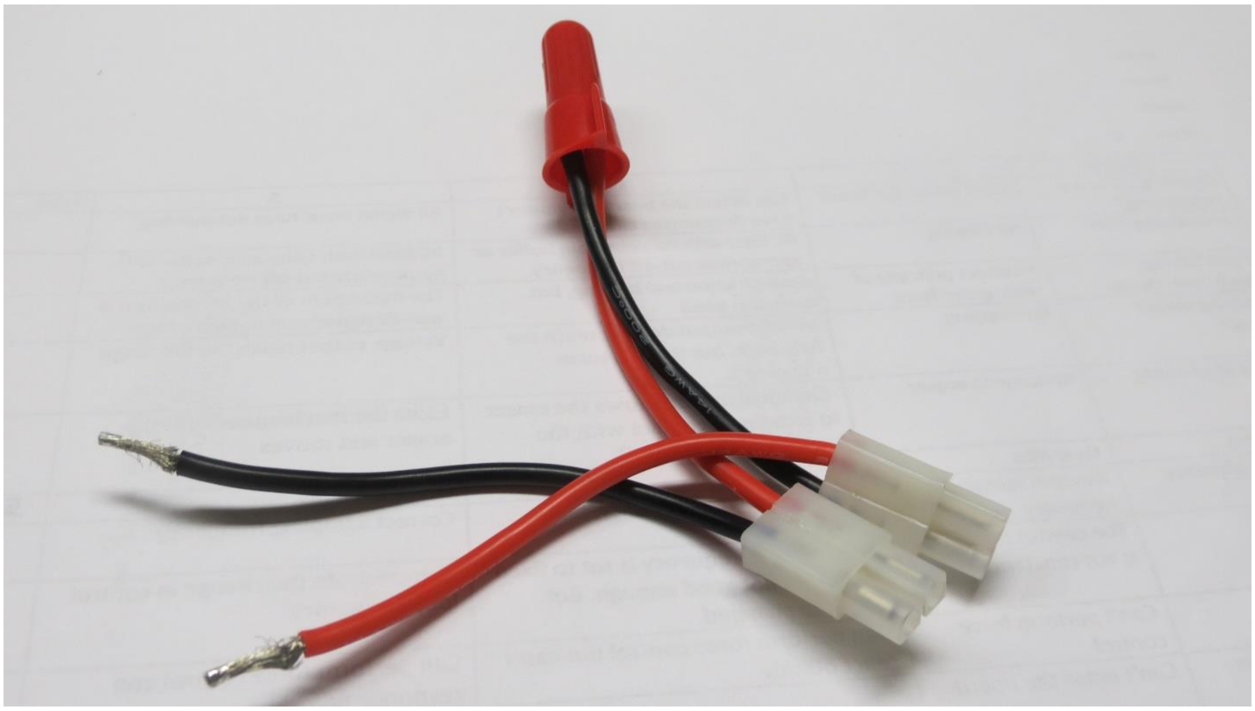

Figure 4 - 1: Battery to T'ReX controller adaptor

The first camera the team used was the Axis M1033-W Network Camera. This camera provides a Wi-Fi connection, but the power cable it uses needs to be wired. In order to achieve wireless control, the team decided to change the power cable to allow it to directly connect to T'ReX motor controller. We cut the power cable and connected it to the leads by using electrical tape.

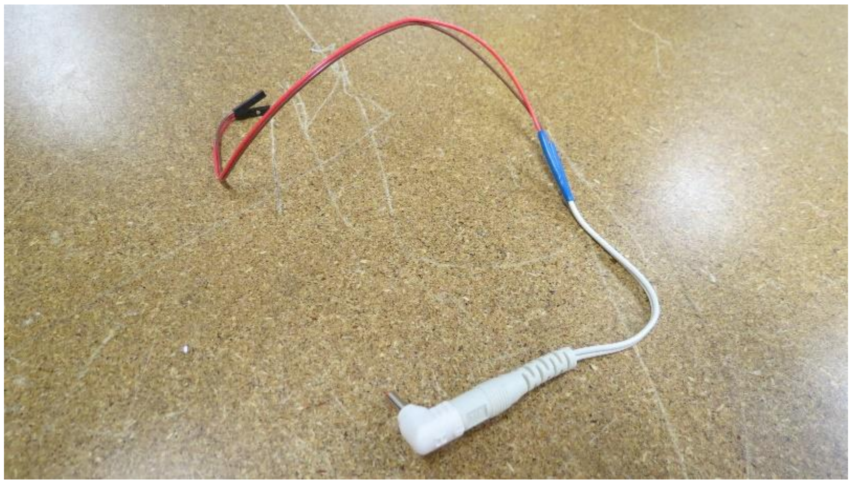

Figure 4 - 2: camera to T'ReX controller adaptor

Wiring

In order for the software to correctly interface with the hardware, components need to be connected in a particular manner. The two Vex batteries were connected to the custom made battery to controller adaptor. The battery to controller adaptor was connected to the motor input and motor output on the T'ReX motor controller. The servo that controlled the arm was connected to pin D7 and the motor that controlled the gripper was connected to pin D12. The T'Rex motor controller board and Logitech WebCam C610 HD were each connected to the laptop via USB.

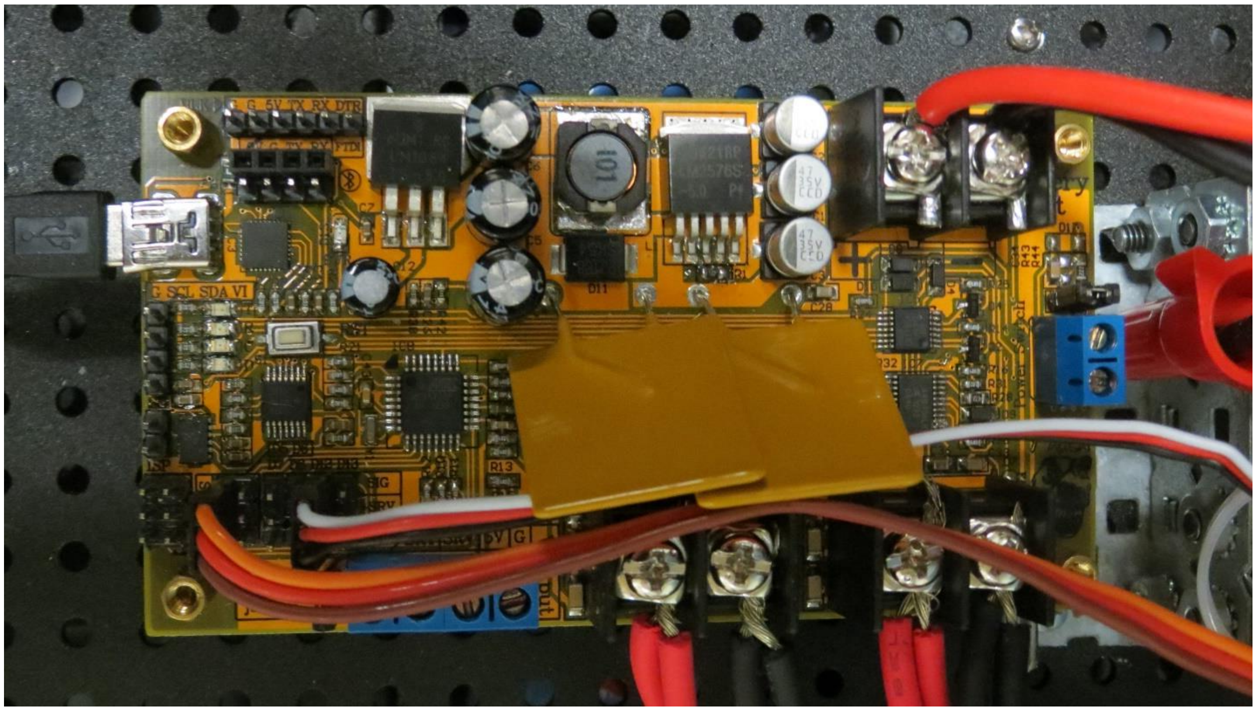

Figure 4 - 3: Top view with components

Software Design

Software Architecture

A design diagram of the software architecture is presented in Figure 4.4. It describes how each software unit integrates with the robot. In this section, we will discuss what each of these nodes does, and explain why such software design decisions were made.

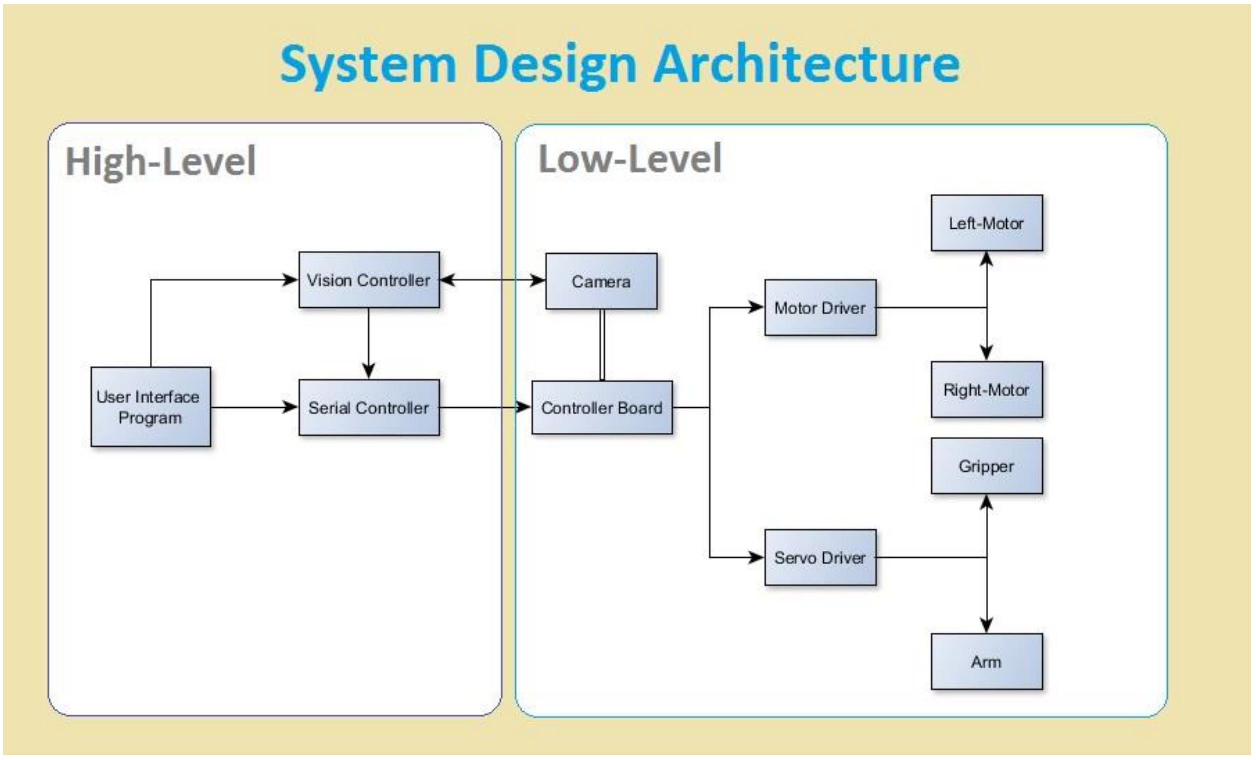

Figure 4 - 4: Software architecture

Software Design and Relationship

This section will review the software design for our robot through discussing all major software component and their functions. While there are some very powerful robotic software platforms such as Robot Operating System (commonly known as ROS), we still decided to develop our own framework for the robot. As shown in the picture above, the software system consists of two parts, the high level part that sits on the laptop and the lower level part that is on the robot. The communication bridge of the two parts is through serial port communication. Our team uses the external java library RXTXcomm to do the serial communication. While the RXTXcomm library provides a convenient communication framework it was still necessary to develop our own organization of tasks to facilitate teleoperated control and autonomous sample return functionality.

User Interface Program

The user interface program is the place where the user could interact with the robot. It provides the user a graphic user interface to control the robot or let the user know what happens when the robot is automatically undertaking the sample return mission. The GUI is programmed using Java Swing to guarantee the portability of the program. The user interface shown in the figure 4.5 is very foolproof and simple to use. We cut all the cumbersome features on the GUI to reduce confusion and only maintained the key features to provide full functionalities. The user could control the robot from the GUI by clicking the buttons and check the robot status through the consoles on the GUI. By using this GUI, the user could teleoperate to drive the robot and control the robot arm and gripper. On the console, the user could see the moving direction of the robot, angle of the robot arms and the gripper status, i.e. if the gripper is open or closed. The left side of the GUI provides a video frame to show the raw image from the camera so that the user could monitor the vision of the robot. Although our final goal is to make the robot accomplish the sample return task autonomously, we still decided to provide the user a GUI to control the robot for some other usage.

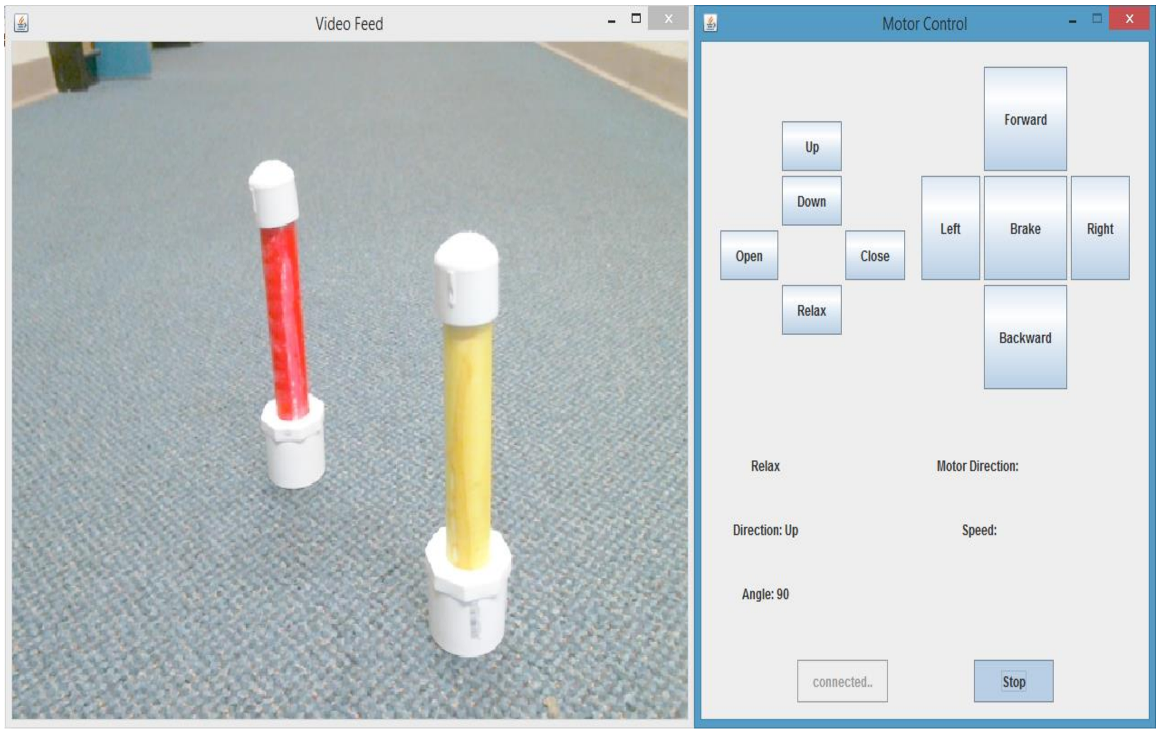

Figure 4 - 5: Graphic User Interface for teleoperation

Since the primary object detection algorithm is the color tracking method, we provide a computer vision GUI for the user to specify the target color that he or she wants the robot to detect. The vision control GUI is shown in figure 4.6. Basically, the vision GUI consists of three components: two window frames and one control panel. The window frame called “Video Feed” will always display the raw video captured from the camera on the robot so that the user could see the original image from the camera in this window. The window named “Object Tracking” will show the processed image by the vision controller. We will describe our vision algorithm in detail in a later section of this report. The image processed by the vision controller will mark the center of the detected object as a yellow point. If the detected object moves on the frame, the program will draw a line between the previous center and the center of object after moving so the user could see the moving track of the target object. In figure 4.6, since we did not provide a specific color for the program to detect, the program will just draw a point in the center of the screen. The control panel is the place where the user could specify the color to detect. However, the user must provide the hue, saturation and value of the target color. The reason for this will be fully explained in the vision algorithm section. Simply speaking, the HSV color space is the most suitable color space for segmentation of colors. When the user articulates the color to detect via the control panel, the robot will target to detect the object, and mark the position of the target in the "Object Tracking" window.

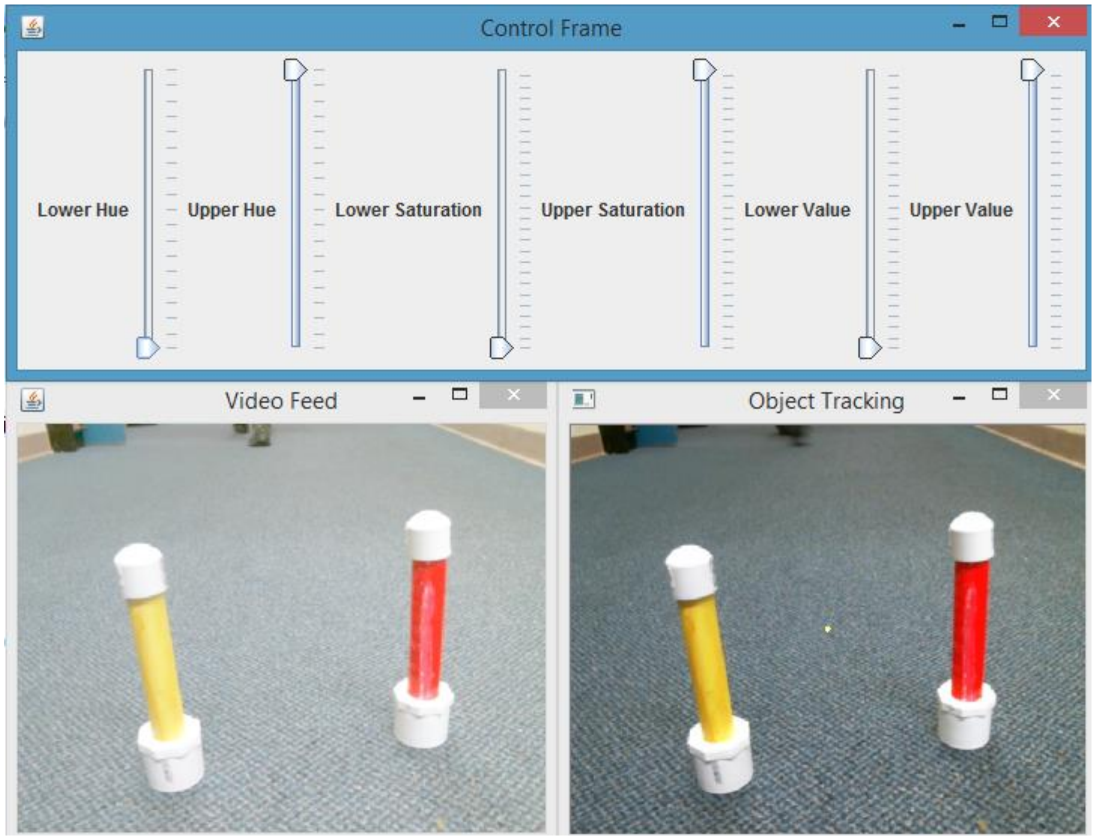

Figure 4 - 6: Computer Vision Graphic User Interface

Vision Controller

The vision controller is the control center for handling the computer vision task. Since the primary goal of the robot is to complete the sample return mission, the vision controller, as the vision analyzing center, plays the crucial role in the software system. The proper functionality of the vision controller directly determines the success of the project. In other words, the vision controller is the heart of the software system. Moreover, our team added some useful functionality to the vision controller to do some processing of the image even though the sample return mission may not need these features. For example, the vision controller could draw lines or polygons on the screen to help the user markup a particular screen area or to help the robot to navigate the path. The robot navigation part of the report will explain how we use lines to navigate the robot. Also, the vision controller could process the source image to convert it to a binary image, or blur the image, or convert it to a greyscale image, and so on.

Serial Controller

The serial controller is the bridge connecting the vision controller and the low level robotic software. When the vision controller finishes analyzing the image captured from the camera, it will send a command to the serial controller and let the serial controller be the delegate to convey the command to the low level. The serial controller will convert the command from the vision controller to the corresponding serial signals and establish serial communication with the low level part through the serial port. And finally, the serial controller will send the processed serial signal to the microcontroller in the low level part of the software system via the USB cable. After that, the microcontroller will control the robot according to the command given to it. When the microcontroller completes the task, it may want to send the operation feedback back to the high level to notify the user. At this time, the serial controller will again act as the messenger between the high level and the low level software system. It gets the signal from the serial port, decodes the signal, and eventually sends the processed feedback to the user interface program.

When the user interface program receives the feedback, it will update the GUI to print out human readable status of the robot on the console.

Navigation System

Our system provides the user a graphic user interface to control the robot; however, our primary goal is to let the robot autonomously complete the sample return task. Since our robot is completely camera driven, it is crucial for us to figure out a suitable navigation algorithm to guide the robot to find the path to the targeted sample. In this part of the report, we will explain the navigation system of our robot.

Since our robot does not have any sensors and encoders -- it is totally driven by the camera -- it is hard to use conventional methods to navigate the robot. Also, traditional camera driven robots always have more than two cameras to assist them in finding the angles and distance to the object. However, there is only one camera installed on our robot. Therefore, the robot navigation once became a hard problem for the team.

The first idea we considered was to separate the camera and the robot. By putting the camera in a position where both the robot and the targeted object are in the vision, our camera could probably direct a way for the robot to find the target. However, this approach turned out to be difficult to implement. First of all, the camera must have the capability to recognize the robot and the targeted object at the same time. After that, the camera must send correct commands to the robot to let it approach the object. Even though it is achievable to have the camera detect the robot and object simultaneously, it is technically hard to tell which direction the robot is heading. Also, as the robot changes its position on the screen, the corresponding command for the camera to send would be different, which is hard to be learned by the machine. Finally, because the camera is fixed and cannot rotate, when either the robot and the object happens to fall out of view of the camera vision, there is no good way for the system to address this situation. Therefore, we had to give up this method of navigation.

After several trials and failures, the final navigation algorithm for our team is the vision line driven method. In this algorithm, the camera is installed on the robot and faces the same direction as the robot head. Since the gripper of the robot is in front of the robot and also faces front, the camera and the gripper are on the same line and facing the same direction. This is an important feature that makes it possible for us to implement the method. As the camera is facing the same direction as the robot and the gripper, the camera view in this case is the robot vision -- what is seen in the camera is just in front of the robot. The overall navigation guide picture is Figure 4.5.

To help the robot find the path to the targeted object, we draw some assistant lines on the camera frame to guide the robot. In figure 4.5 you could see there are two vertical lines in the middle of the window and three horizontal lines in between the two vertical lines at the bottom of the window. The two vertical lines in the middle are the robot direction guide lines. When the target appears in the camera vision, there are only three possible situations regarding the two lines — the object must be on left of the left vertical line, or on right of the right vertical line or just in between the left vertical line and the right vertical line. The robot will use the position of the object in the screen to adjust the moving direction. When the target is on the left of the left vertical line, such as the situation presented in figure 4.6, the vision program will ask the robot to turn left. Similarly, when the aiming object is on the right of the right vertical line, the robot will be told to turn right. The robot will keep turning right or turning left until the targeted object falls between the right vertical line and the right vertical line.

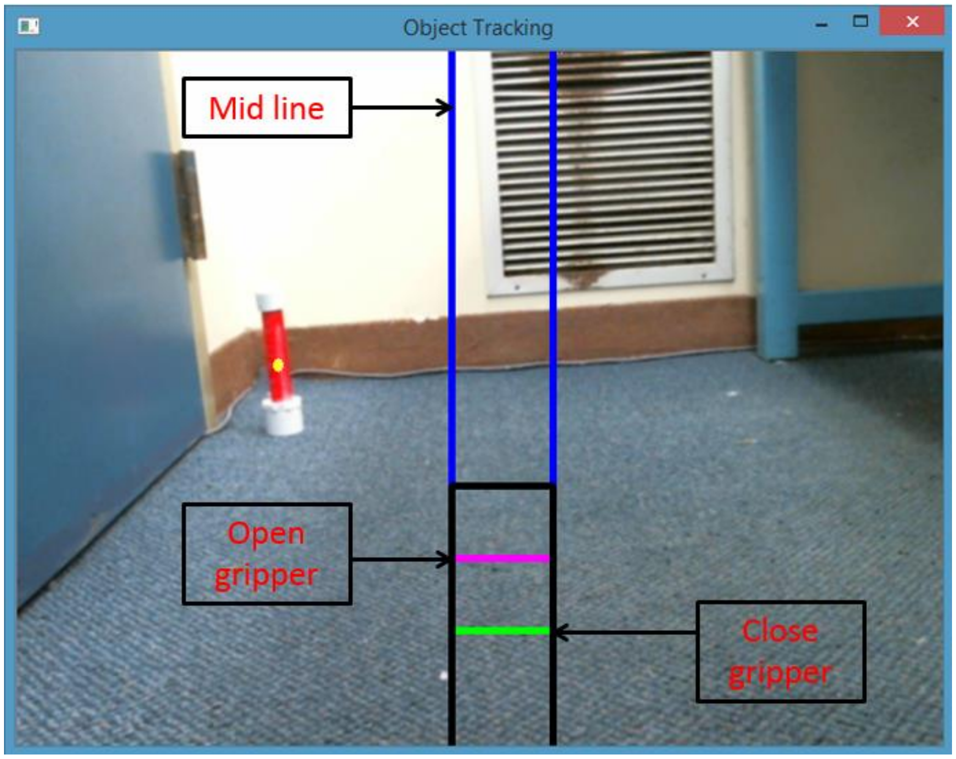

Figure 4 - 7: Overall Robot Navigation

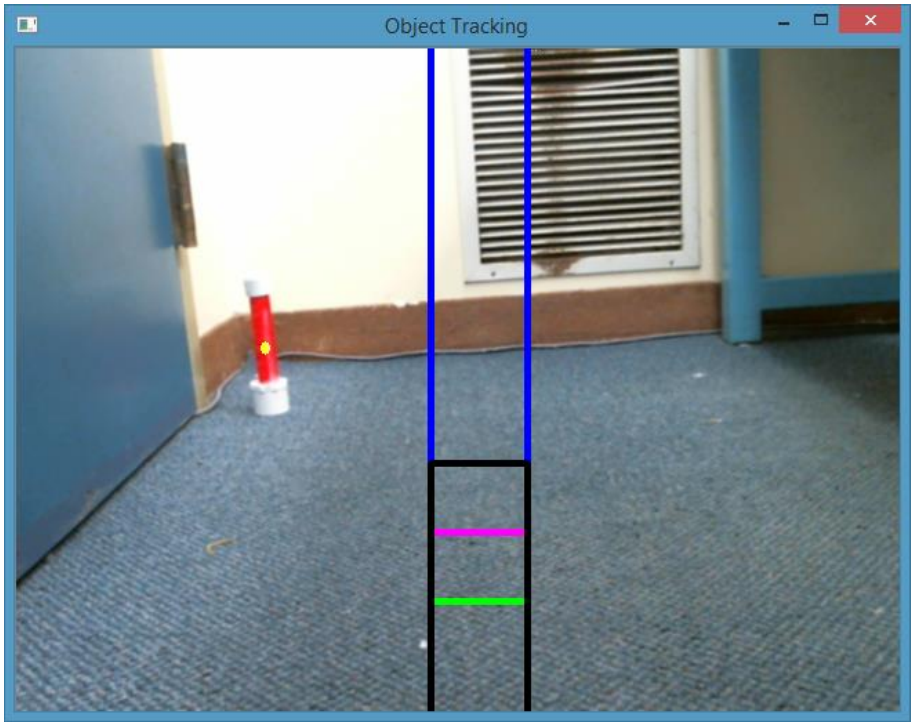

Figure 4 - 8: Robot Navigation 1

When the target in between the two middle lines, as shown in the figure 4.7, the robot will take action according to the Y coordinate of the center of the target. In figure 4.7, even though the object is in the range of the two middle vertical lines, the center of the object (marked as the yellow point by the program) is above all of the three horizontal lines (the black, purple and green horizontal lines drawn between the two vertical lines). In this case, the program will know that the robot, although facing towards the target, is still far away from the object, so it will just let the robot keep moving forward. Whenever the target is out of the range of middle lines, the robot will automatically change the moving direction again.

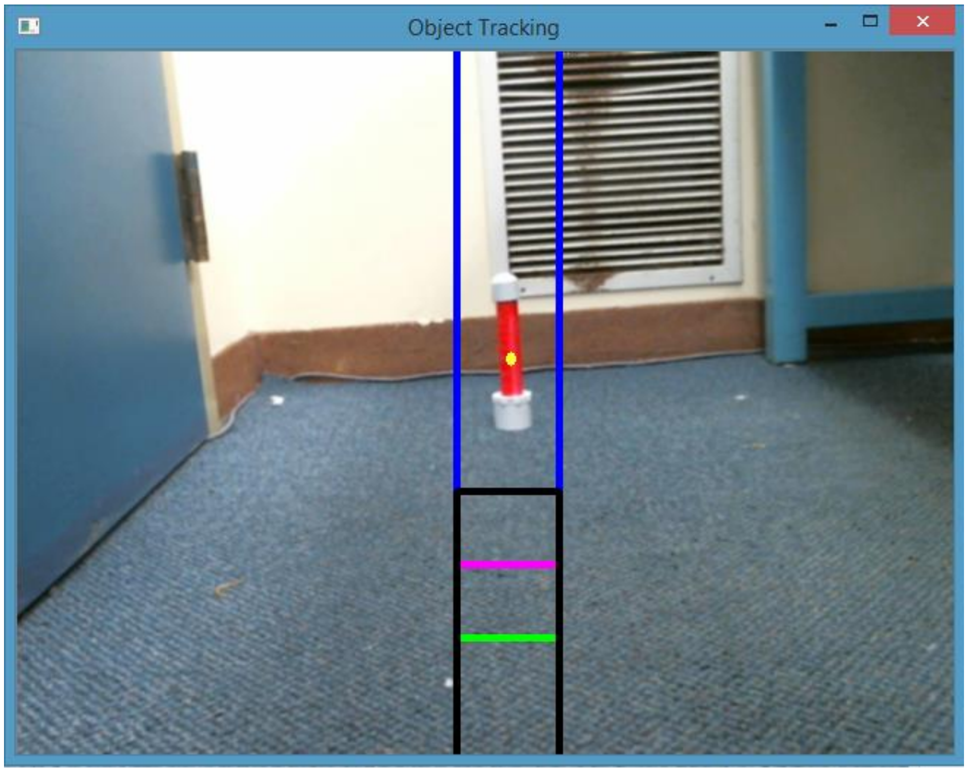

Figure 4 - 9: Robot Navigation 2

As the robot moving forwards, it will become closer and closer to the target. The object position in the window therefore will gradually move downwards as the robot approaches it. When the center of the object is below the black horizontal line, the program will print a message on the console to express that the robot is approaching the target and should be prepared to take actions later. In fact, in this case, the program will not send commands to the robot do something special; however, we set a check point here to help us the debug the software system. As the center of the object is below the purple line (figure 4.8), the program will tell the robot that it is very close to the object and ask the robot to open its gripper. As we mentioned early, the camera, robot and gripper face the same direction on the same line; therefore when the gripper is open, it is still facing towards the target. Now, the robot is ready to grab the target.

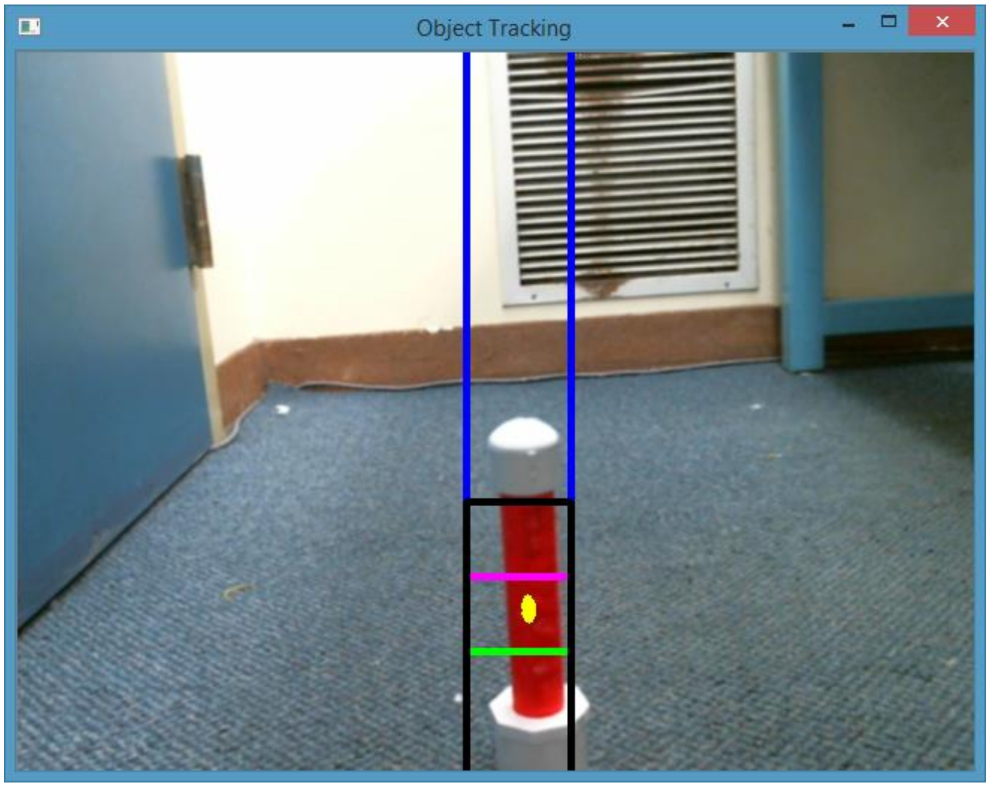

Figure 4 - 10: Robot Navigation 3

Eventually, when the robot has finally reached the appropriate distance to the targeted object, i.e. the center of the object in the screen is just below the green line, the program will command the robot to close the gripper and lift up the object. The final status of the navigation is presented in figure 4.9.

One thing to be noticed is that when the targeted object is not in the vision of the camera, or the camera happens to not detect the object in front of it for some reasons, the program will ask the robot to turn around to attempt to find the target. When the target appears in the window and has been successfully captured by the camera, the ordinary navigation process that we described above will begin to operate.

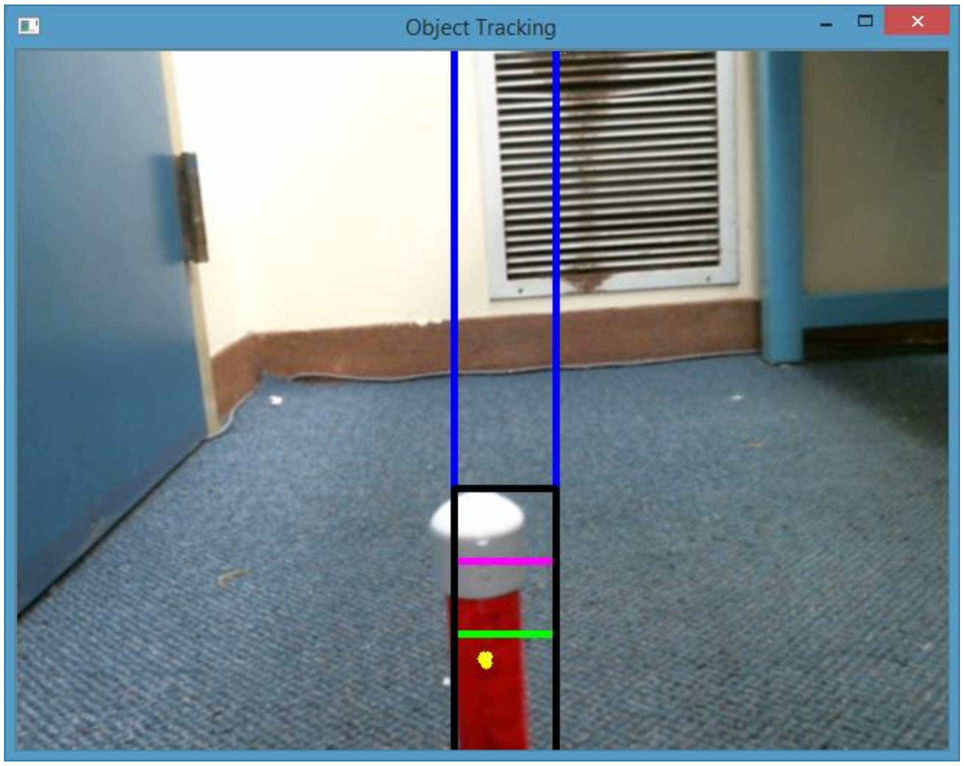

Figure 4 - 11: Robot Navigation 4

Robot Vision

In this section of the report, we will articulate the vision analysis process of the robot. Our robot is fully driven by the camera, so the computer vision processing is the critical part of the project to guarantee the functionality of the robot. In the early part of the report, we already explained the choice of the camera that is used for this project. The network condition turns out to be an issue that has a great impact on the performance of the video frame. In computer vision analysis, a stable frame capture rate is important—obviously the robot must get a sense of what is happening now, but not what happened in the past. If the frame transfer rate is too low from the camera, it will probably cause delay in the robot action. For instance, if the camera could only capture a frame every two seconds, the vision controller may end up analyzing an image captured two seconds ago and cannot make correct judgments based on what it sees. Therefore, the Logitech webcam is the best camera to accomplish the vision task.

Our team attempted several different vision algorithms and compared the performance of these algorithms to choose the best one. There are many powerful computer vision algorithms that used by people to detect specified objects and all of them have their own advantages and disadvantages. What the team needed to do was choose the best one that is most suitable to solve the problem in this particular situation. Now, we will briefly introduce each of the algorithms we used in the project and explain why or why not we chose to use it.

SURF Algorithm

SURF, Speeded Up Robust Features, is one of the most powerful and popular vision algorithms to detect targets. SURF is a robust local feature detector and an improved version of Scale Invariant Feature Transform (SIFT) algorithm. The underlying theory of SURF is based on the summary of 2D Haar wavelet response and making an effective implementation of integral images. And SUFR depends on the determinant of the Hessian matrix for location and scales. SURF could quickly calculate integral images with three integer operations by using an integer approximation to the determinant of Hessian blob detector. [16]

OpenCV offers built in SURF functionalities to help the user implement a SURF algorithm in the project. Our team took advantage of this feature of OpenCV and tried SURF as our detection algorithm in the beginning. In the procedure, we first provide a clear image of the target object to program. The program will then instantiate a SURF object and set up the initialization according to the parameters provided by user, such as Hessian threshold. After that, the program will use the built in functions SURF.detect() and SURF.compute() to find all the descriptors and keypoints in the image frame, and according to the computed value figure out the location and the matching rate of the object. Figure 4.11 shows how we tried to use a SURF algorithm to detect a mobile phone through the webcam on the laptop. The left side on the picture is the real photo for the phone as the targeted object, while the right side is the video feed from the webcam. As shown in the figure, the SURF algorithm succeeded in recognizing the phone held in a human hand.

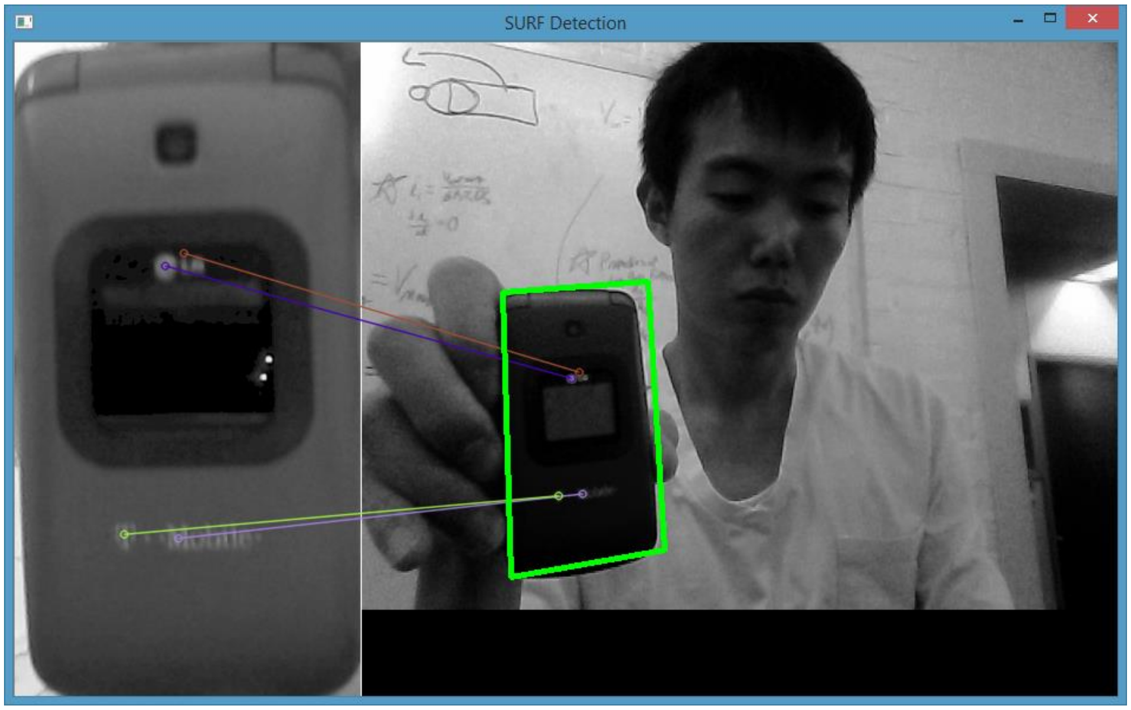

Figure 4 - 12: Using SURF to detect phone

Although SURF is a very powerful object detection algorithm with many successful applications, we found that this method is not suitable for our project. The reason is that due to the environment noise and distraction, SURF could not always correctly detect the target. In the experiment, our team found that the detecting rate by using this algorithm in the real world circumstances was relatively low. Since to accomplish the sample return mission the program must be able to keep track the target position, SURF algorithm would not be a good choice for the consideration. Therefore, we decided to give up the exploration of SURF.

Haar Cascade Classification

Another vision algorithm that our team attempted to use is the Haar cascade classification algorithm. It is also one of the traditional object detection algorithms used by many other academic researchers and commercial organizations. Similar to SURF algorithm, the Haar classification method uses the Haar-like features to do the object recognitions. The so-called Haar-like features are based on the Haar wavelet. It considers the adjacent polygons regions of the particular areas on the detecting window, adding up the pixel intensities of the regions and computing the difference between the intensity summaries. After that, the Haar classification algorithm uses the difference between the intensity sums to classify the subsections of the analyzing image. The Haar classification algorithm is usually used to recognize human faces — the eye area of the face is usually darker than the cheek area, which could form a Haar feature for face detection. Figure 4.12 shows the usage of the Haar classification algorithm to detect human faces.

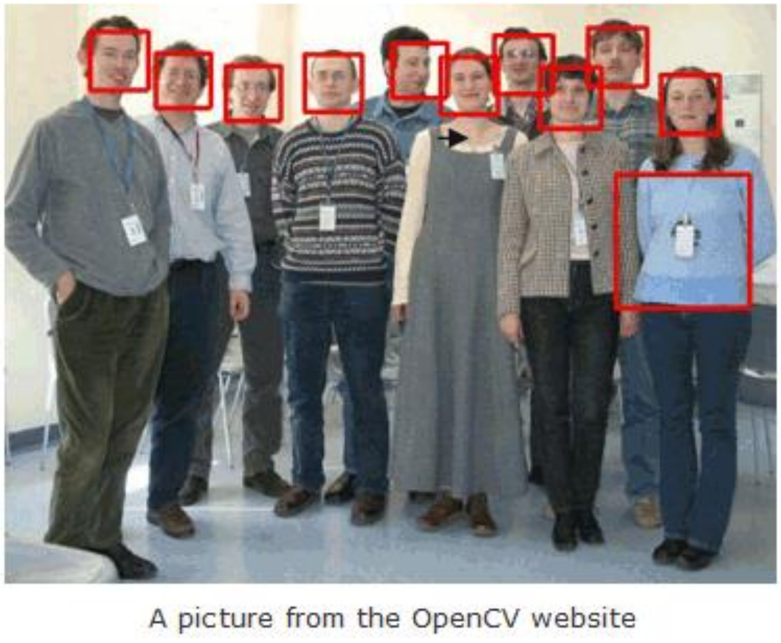

Figure 4 - 13: Usage of the Haar cascade classification algorithm

In OpenCV, there is a trainer as well as a detector built into OpenCV to help the user utilize the Haar cascade algorithm. To use this algorithm, the user must provide the target object classifier to deal with the detection. The classifiers themselves are XML files containing the Haar feature of the target object and some useful parameters to do the object tracking. OpenCV offers many pre-trained classifiers for faces, eyes, smiles, etc. However, if we want the program to detect a specified object for the project, we must train our own classifier for the object through the cascade classifier training. The cascade classifier training is in general a very complex process and they did not provide us several important pieces of information such as what images and parameters they used for training. Also, the Haar feature detection is not angle independent, which means, when the same object appears in the camera with a different angle, the program may not detect the object any more. Therefore, the Haar classifier detection is also not our choice.

Color Filtering Algorithm

The final choice for the vision algorithm is the color filtering/tracking method. This algorithm utilizes the fact the target object has a different color from the other objects in the environment to quickly locate the position of the targeted object. In real world experiments, our team finds that unless the object is put into an environment with many same color distractions, the algorithm works pretty well in detecting the object. Also, the accuracy and the success rate of the method is much better than the two other algorithms we mentioned earlier. Therefore, our team eventually decided to use this vision algorithm to be our primary object detection algorithm. The only drawback of the method is that the detection distance is relatively close due to the capability of the camera we installed on the robot. Now, the robot could only detect the targeted object within 5 meters. However, our team could always easily improve the detection distance by using more advanced devices later. Now we will explain the color detecting algorithm in details.

The color based object tracking method first assumes that there is a significant difference between the color of the target and the color of the environment. If the color contrast is not large enough, the performance of this method will be largely limited. As to the algorithm itself, the program first will capture an image from the camera to be the source image. All vision analyzing and image processing be will based on this image. When the image has been captured from the camera, the program will blur the image to reduce the color noise. As the image is being blurred, some small areas of the distracting colors will be removed from the image; therefore the accuracy of the detecting will be increased.

OpenCV by default captures an image from the camera in a format of 8-bit, unsigned and RGB format. In other words, OpenCV captures an image consisted of 3 matrices—the red matrix, the green matrix and the blue matrix, each matrix element with integer values ranging from 0 to

255. As to the image, it consists of many small boxes, i.e. pixels that are too small to be distinguished by human eyes. However, since our algorithm needs to do the color segmentation and isolation, the program needs to achieve a color space that is easy to do the separation. In fact, HSV color space is actually the most suitable color space for the color based isolation and segmentation. For this reason, the program will therefore convert the source image from RGB image to HSV image to convert the color space from the RGB color space to the HSV color space. As to the HSV image, it still consists of three matrices: the hue matrix, saturation matrix and value matrix. The ranges of the elements of the three matrices are 0 to 180, 0 to 255, and 0 to 255 respectively. As to the meaning of each component, hue indicates the specific color, saturation represents the extent to which the specified color is mixed with white color, and value indicates the extent to which the specified color is mixed with black color. To detect a specified color, the program must know the range of the hue, saturation and value of the target color to separate the color from the image. For instance, to detect a yellow object, the hue range for yellow is usually set to 20 – 40. Even though we could easily figure out the hue range for a specific color, the saturation and value of the color usually rely on the object materials and environment lighting; therefore, the specified saturation and value of the color must be determined according to the environment. Therefore, in our program, we provide a color panel for the user to specify the hue, saturation and value of the target color, so that the user could adjust the parameters according to the specific environment.

After converting the image to an HSV image, the program will limit all the colors that do not match the color criteria according to the parameter ranges given by the user. Then the program will check if the areas detected are large enough be our target object. Our team set a threshold value for the area of the target object so it will again reduce the color noise from the environment. As presented in figure 4.13, all the other color is eliminated by the color based method and only the detected area is represented as the white area in the window. The shape of the detected area is just the shape of the targeted object.

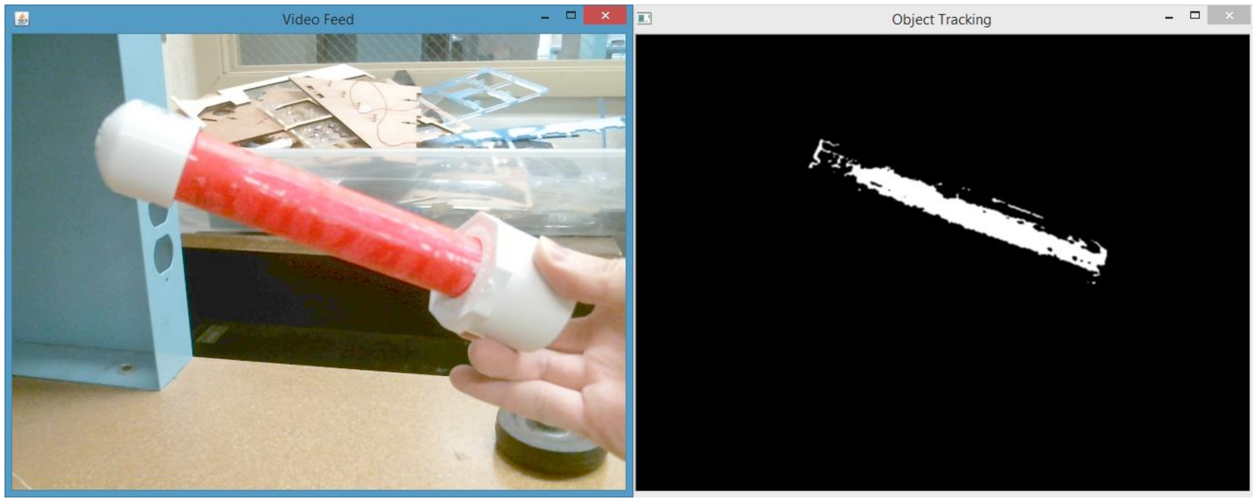

Figure 4 - 14: Color tracking method to find the area of the target color

After the area is detected, the program will calculate the X and Y coordinate of the center of the object by using moments. The program must compute the first order spatial moments around X axis and Y axis and also compute the zeroth order central moments of the binary image. In this case, the zeroth order central moments of the binary image is equivalent to the white pixels area in the binary image shown in figure 4.13. The X coordinate of the center of the detected area is therefore equal to first order spatial moment around X axis divided by zeroth order central moment. And similarly, the Y coordinate of the center of the recognized area is equal to first order spatial moment around Y axis divided by zeroth order central moment. As we set a threshold value for the area, if the detected area is less than the threshold value, the program will choose to ignore the area. This is because there may be many noisy color pixels to be detected and we do not want to let the noise colors reduce the accuracy of the color algorithm. In our case, we set the threshold value to be 200 pixels so that the program will easily detect the target object without being distracted by the noise while it could still detect the object in a relatively far distance.

Finally, the program will copy the original image to another image with the same size and combine the threshold image with the copied image. Also, the program will mark the center of the detected area as a yellow point. When the object moves, the detected center coordinate will change; the program will draw a line between the original position of the detected center and the current center position so that the user could easily see the moving track of the targeted object in the window. In figure 4.14, the program is detecting the red color object. As shown in the picture, as the object moves from one position to another position, the program will draw the moving track as yellow lines. The user could easily see what happened after the algorithm is applied via the yellow mark in the window. Also, if the yellow marks become too noisy for user to see the image, user could press ‘r’ key to clear the screen.

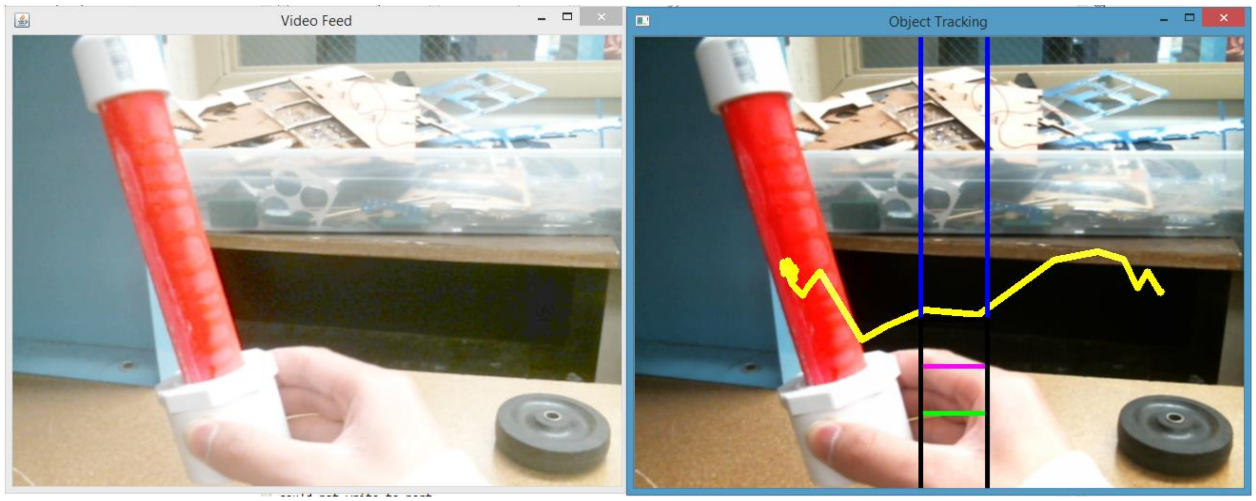

Figure 4 - 15: Color tracking red object and marking the moving trace

The color tracking algorithm is very powerful as we implemented it on the robot to complete the sample return mission. As long as the target object is within the view of the camera, the program could quickly detect the object. Therefore, our team eventually decided to use this method to be our primary vision analysis algorithm. The disadvantage of the method is that if multiple sample objects appear in the vision simultaneously, the program could not figure out the correct X and Y coordinates of the center of the objects. However, since our goal is to detect one specified sample and return it to another position, the algorithm is suitable for the situation and therefore a good choice for our project.

Chapter 5: Results

The criteria for judging our project is to see if our robot could complete the sample return mission autonomously, since it is the primary goal for this project.

Our team chose to do a field experiment to test the performance of the robot. In the test, our team decided to use two different objects with different colors to be the sample and the destination mark respectively. The objects chosen have the same shape and materials—plastic sticks with white hubs on the ends for supporting them. The color of the sample is red while the destination object is yellow. The robot needed to detect and navigate its path to the sample, grab it using its gripper, and return it to the position of the destination mark. At the beginning, the sample was about 5 meters away from the robot and not in the view of the robot camera. The destination mark was about 5 meters away from the sample and also not in the vision of the camera when the robot should pick up the sample. To complete the task, the robot must first detect the sample and find its way to pick it up; after that, it should find and detect the destination object and navigate a path to reach it and put the sample at that point.

We chose the testing location as the hallway outside the AK 116 room. The reason to choose this place as the experiment location is that the colors of wall and ground here are contrasting to the target colors. In this case, our team could largely ignore the color noise from the environment and focus on testing the performance of the vision algorithm itself. Also, the space here is large enough for placing the sample object and the destination object 5 meters away from each other. Last but not least, the floor here is smooth therefore it would not stop the robot due to the high friction as the robot turns around.

The experiment began with the robot facing toward a direction without the sample object in the camera range. In figure 5.1, the window named “Video Feed” shows the raw video from the robot camera (part of the window is clipped by the video from the hand-held camera). As you can see, the red sample object is not in view. The window named “Object Tracking” shows the processed image after the vision algorithm is applied to the source image. The lower left is the video frame from the hand-held camera so that the user could see from a 3rd person point of view.

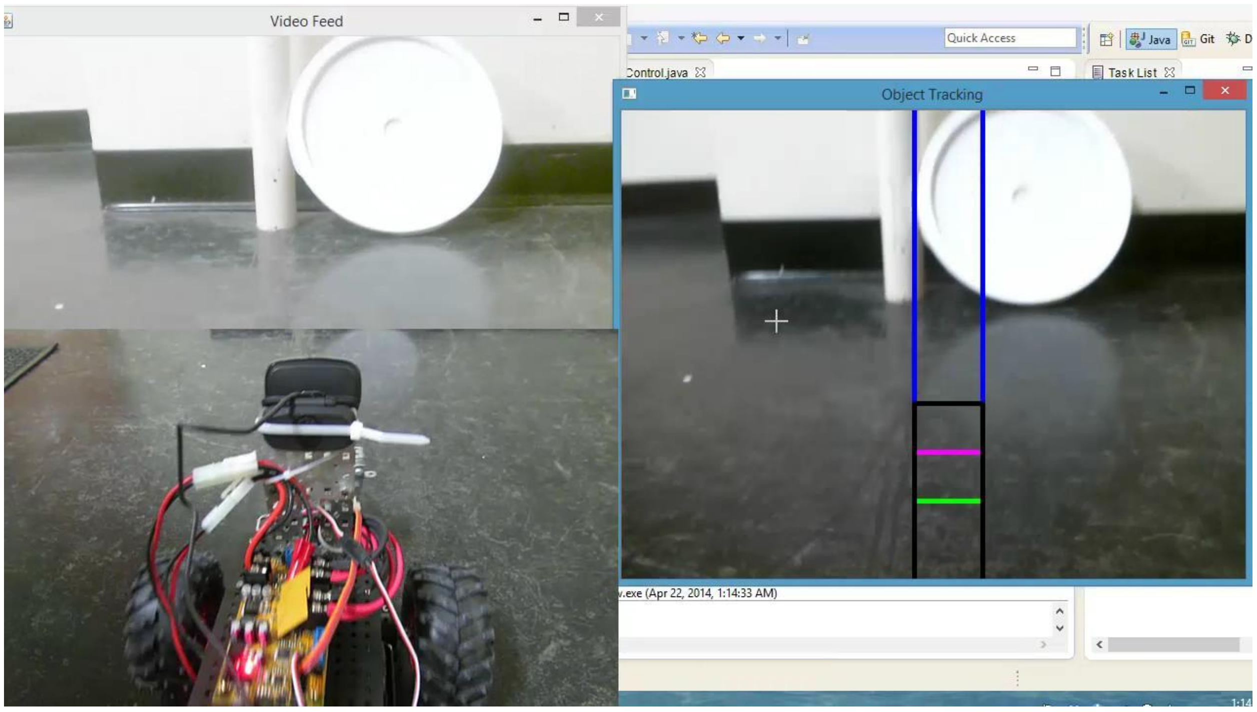

Figure 5 - 1: Testing Sample Return Robot 1

As the robot did not see the object in the vision, the robot started to turn around to try to find the object, just as we expected.

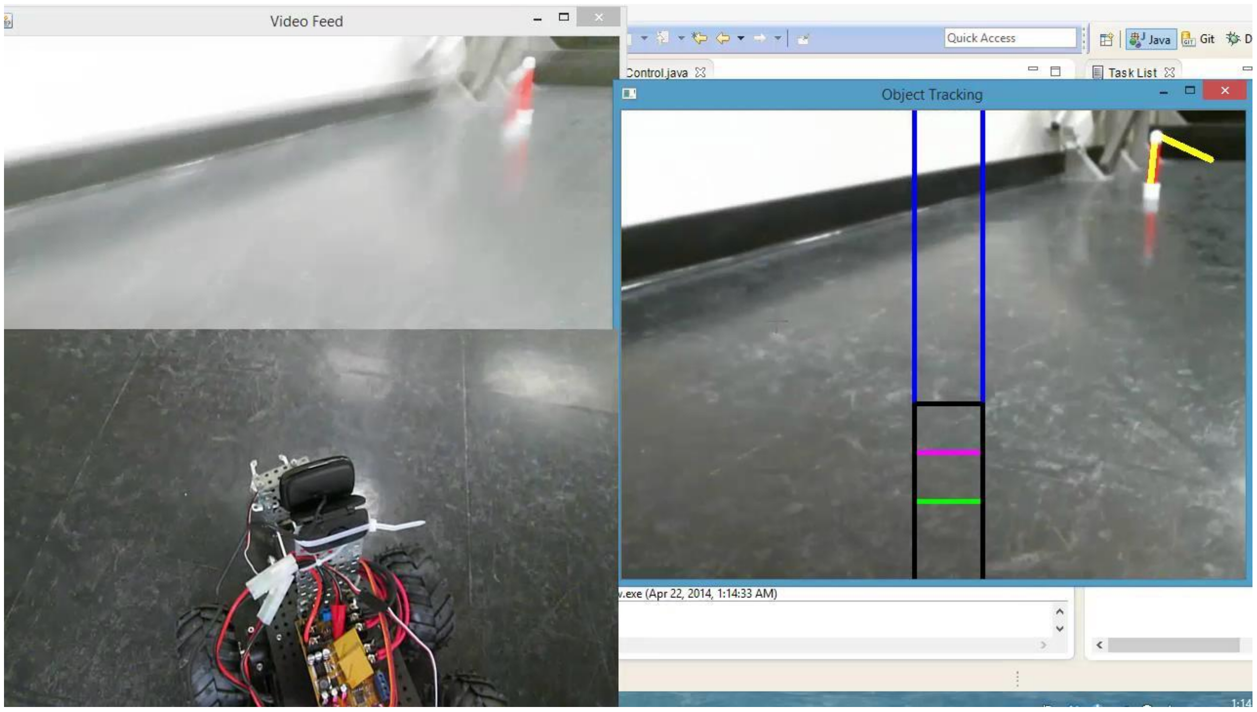

Figure 5 - 2: Testing Sample Return Robot 2

As soon as the object appeared in the range of the camera, the robot accurately detected the object and drew a yellow point at the center of the target to mark the position of it on the screen. In figure 5.2, the yellow lines in the "Object Tracking" window are the moving track of the detected object. In fact, the sample object itself was not moving, but as the robot was moving, it seemed that that the object was moving on the screen. As the object first appeared to the right side of the middle navigation line, the robot knew that the object was on the right side of it, so it kept turning right until the center of the object on the screen was between the two middle navigation lines (refer to the robot navigation part to learn more about the robot navigation). In this case, the robot knew that it was now facing towards the object. Because the object center was on the upper part of the screen, it was still far away from the robot. As the robot moved forwards, the center of the target gradually moved downwards. When the robot was close enough to the target (the center of the target was just below the purple line), the robot opened its gripper and moved forwards to the object slowly. When the object center was finally below the green line, the robot closed its gripper and lifted the sample up. At this point in the experiment, the sample detection and collection part has been successfully completed (figure 5.3).

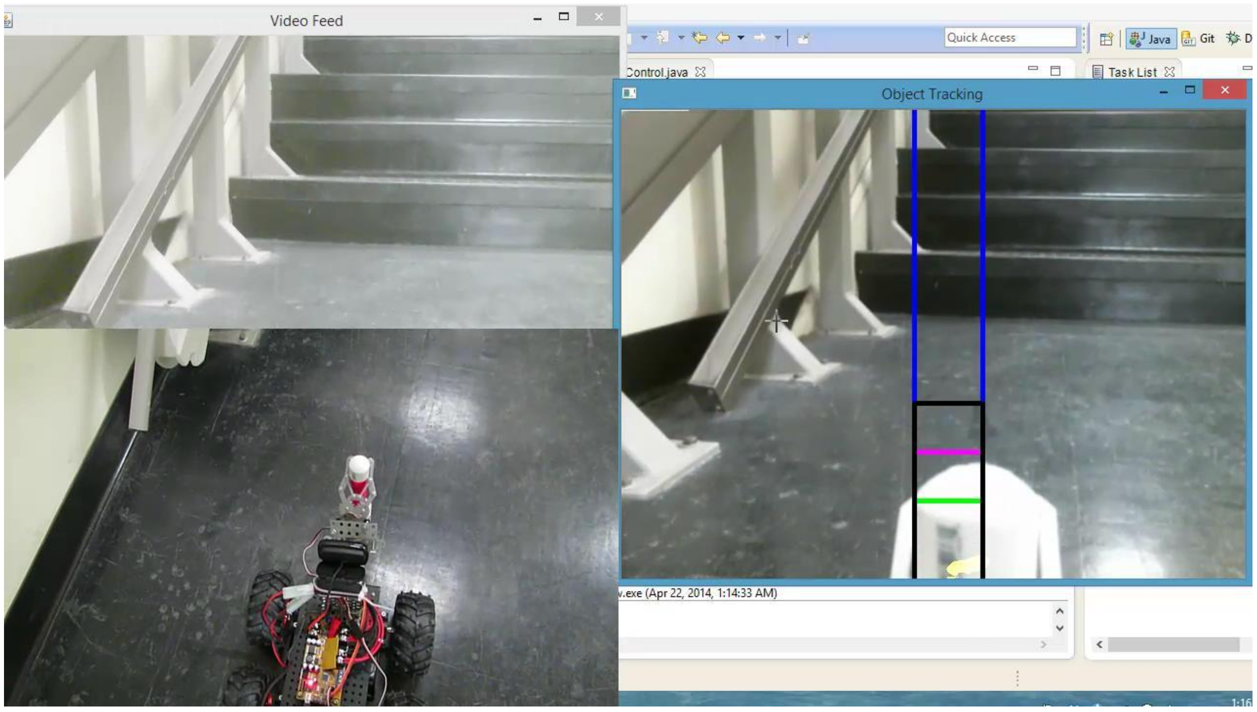

Figure 5 - 3: Testing Sample Return Robot 3

After the robot finished getting the sample, it started the sample return part of the mission. As the destination point was not in the camera range, the robot again turned around to try to find the sample. The consequent process was almost the same as before. Once the robot detected the destination (yellow object), it navigated a path to approach it and when the robot was close enough to the yellow object, it lowered the arm and released the object. As shown in figure 5.4, the robot succeeded in returning the sample to the correct position. At this point, our experiment is complete, with the robot successfully accomplishing the sample return task.

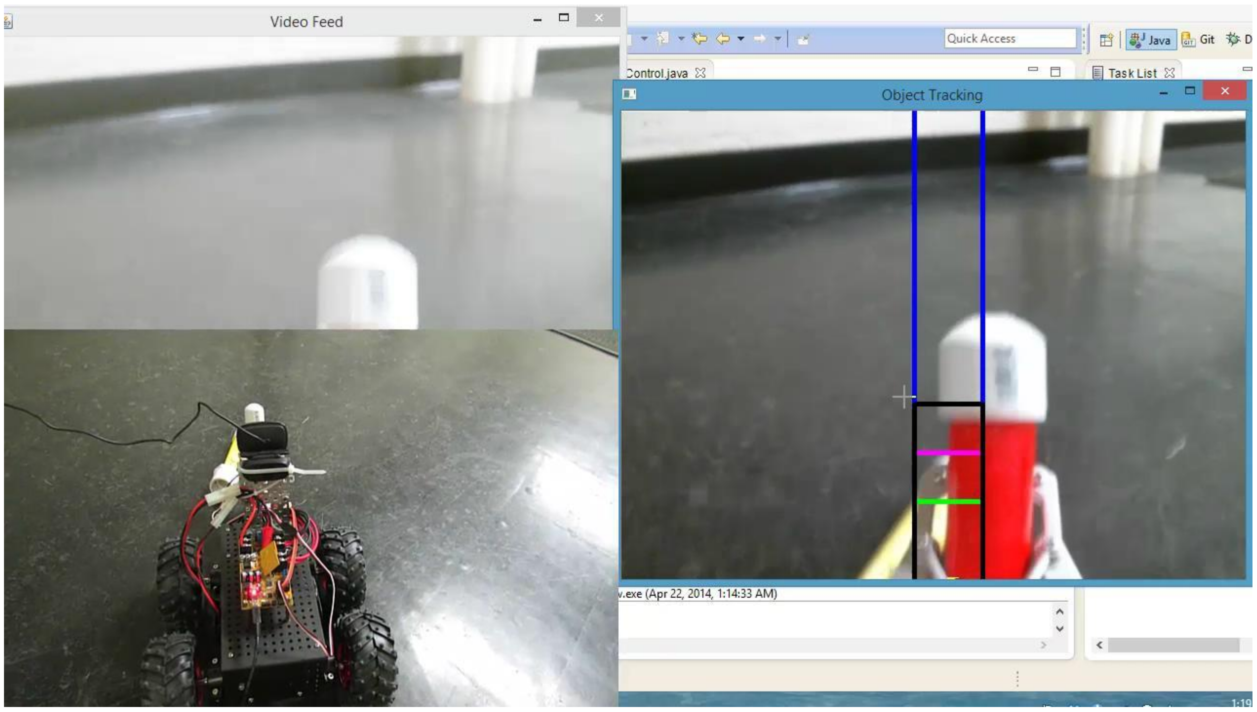

Figure 5 - 4: Testing Sample Return Robot 4

Chapter 6: Conclusion

Based on the result of the experiment, it shows that accuracy of the robot vision detection is competent to complete the task. The distance of detection is about 5 meters — if the target object is further away from the robot than 5 meters, the accuracy of the detection will be largely reduced. Also, there should not be much color noise in the testing environment; otherwise the performance of the robot will be affected greatly.

To improve the detection range of the robot, one possible solution is to use a more powerful camera that could zoom in and out to see further. To reduce the environmental influence on the robot performance, combining two different vision algorithms together to complete the job may be a good choice. For instance, the vision controller could switch between the color tracking method and the SURF algorithm to detect the target object. When it uses the color tracking method to look for the target and finds more than one target in the range, it means that the color noise is too large to be ignored by the color tracking algorithm. In this case, the program would automatically switch to the SURF algorithm to detect the target. However, since now the success detection rate of the SURF and Haar cascade classification algorithms are very low, our team must first improve the performance of those two algorithms to make the combination detection possible.

Overall, the robot succeeded in accomplishing the sample return task autonomously only the camera, which means possible applications could be implemented by using the vision driven technology used in this project. However, there are still many restrictions and drawbacks in the implementation and we expect to improve the performance of the robot in future work.

Chapter 7: Future Work

After completing the intelligent vision driven robot for sample detection and return project, we have some recommendations. Despite having met all of our set goals and successfully implementing the user interface program, there is still much room for future improvements. Completion of these recommendations would help to increase the detection range and add more functionality to the robot. We have divided our recommendation into two categories: mechanical recommendations and software recommendations.

Mechanical Recommendations

Our mechanical recommendations are:

1. Change the gripper design to allow the robot to be able to grab objects such as rocks.
2. Change the camera to a wireless camera and add a Bluetooth module in order to increase the detection range of the robot and achieve wireless control.

The first mechanical recommendation is to change the gripper design of the robot. The current gripper design can only allow the robot to grab objects of a certain shape, such as cylinders. It would be impossible for the robot to grab rock or dust since the gripper cannot hold such objects. For this reason, we suggest designing a new gripper that would allow the robot to grab objects that have different shapes. One possible design is a scoop design that has a large upper scoop and a flat bottom that is controlled by a motor. The flat bottom opens and closes in order to hold a rock. Another possible design is to have a scoop design that could open and close horizontally. The robot arm is then maneuvered near the rock, and then the scoop closes and picks up the object.

Second, we recommend using a Bluetooth module for the robot to achieve wireless control. Currently, the laptop communicates with the T'ReX thorugh a USB cable. A Bluetooth module could be easily added to the T'ReX motor controller to solve the problem. The team found out the DAGU Serial Bluetooth Module can directly connect to the current controller through the TTL level Serial Port. Having the Bluetooth module implemented will increase the current detection range of the robot. Similarly, the current camera we use needs a wired connection; switching to WiFi would eliminate this.

Software Recommendations

Our software recommendations are:

1. To use Pixy (also known as CMUCam 5) to improve user experience and provide more accurate detection

2. Add A* algorithm to find the best path.

The current system requires the user to provide the hue saturation value of the object to be detected. By using Pixy, the user can teach it to find an object by pressing the white button located on the top, and holding the camera in front of the desired object for a few seconds to let the sensor see the object. While doing this, the LED under the camera provides feedback by changing to the color of the object it is looking at. When the color changes, the user would release the button and Pixy would store the detected color in flash memory. It will use this information to find objects with similar color in its frame. Using Pixy will improve the user experience and provide a more accurate detection of color.

The second recommendation is to add the A* algorithm to the system. This would also require adding sensors and encoders to the robot. The current system would go straight towards the detected position and would not avoid obstacles. By adding a laser sensor and encoders, and implementing the A* algorithm in the system, the robot would be able to achieve autonomous searching in an unlimited area.

Appendix

Size	280 × 300 × 130 mm (11" × 12" × 5")
Weight	1.9kg (4.1 lb)
Ground clearance	60 mm (2.5") when lightly loaded
Recommended motor voltage	2 – 7.5 V
Stall current at 7.2 V	6.6 A per motor
No-load current at 7.2 V	420 mA per motor
No-load output shaft speed at 7.2 V	160 RPM
Stall torque at 7.2 V	11 kg-cm (160 oz-in) per motor

Appendix - 1: Specifications for Wild Thumper All-Terrain Chassis

MCU	ATmega328P
Clock speed	16MHz
Logic voltage	5V
FLASH	32K
SRAM	2K
EEPROM	1K
Bootloader	Arduino Nano w/ ATmega 328

Appendix - 2: Specifications of T'ReX Robot/Motor Controller Processor

Supply voltage	6V – 30V
Power switching FET maximum current	20A continuous – 110A peak
Switch mode regulator	6V, 3A maximum, 52kHz
+5V output current	2A continuous – 3A peak

Appendix - 3: Specifications of TReX Robot/Motor Controller Power Supply

References

[1] "Mars Sample Return Mission." *Wikipedia*. Wikimedia Foundation, 22 Mar. 2014. Web. 24 Mar. 2014.

[2] "Solar System Exploration: Missions: By Name: C: Curiosity." *Solar System Exploration: Missions: By Name: C: Curiosity*. N.p., n.d. Web. 24 Mar. 2014.

[3] *NASA*. NASA, n.d. Web. 24 Mar. 2014.

[4] "Mars 2020 Rover Mission." Wikipedia. Wikimedia Foundation, 24 Mar. 2014. Web. 24 Mar. 2014.

[5] "Space Images: Artist's Concept of Mars 2020 Rover - NASA Jet Propulsion Laboratory."Space Images: Artist's Concept of Mars 2020 Rover - NASA Jet Propulsion Laboratory. N.p., n.d. Web. 24 Mar. 2014.

[6] "Sample Return Robot Challenge." | NASA Centennial Challenge. N.p., n.d. Web. 24 Mar. 2014.

[7] "Centennial Challenges." Wikipedia. Wikimedia Foundation, 24 Mar. 2014. Web. 24 Mar. 2014.

[8] T.S. Huang 'Computer Vision: Evolution and Promise', University of Illinois at Urbana-Champaign, Abstract

[9] http://cs.brown.edu/courses/csci1430/lectures/01.pdf

[10] "Computer Vision." *Wikipedia*. Wikimedia Foundation, 30 Mar. 2014. Web. 30 Mar. 2014.

[11] "How Google's Self-Driving Car Works." - *IEEE Spectrum*. N.p., n.d. Web. 27 Mar. 2014.

[12] "Robotics: A Brief History." *Robotics: A Brief History*. N.p., n.d. Web. 28 Mar. 2014.

[13]"IFR Press Release - Executive_Summary_WR_2013.pdf." *IFR RSS*. N.p., n.d. Web. 30 Mar. 2014.

[14] OpenCV Wikipedia (http://en.wikipedia.org/wiki/OpenCV)

[15] RXTX: The Prescription for Transmission (http://users.frii.com/jarvi/rxtx/index.html)

[16] SURF: Speeded Up Robust Features, Herbert Bay, Tinne Tuytelaars, and Luc Van Goo, Web. 2006