**VISION GUIDED MOBILE ROBOT FOR AUTOMATED SURVEILLANCE**

ECE 487: Final Proposal

**Prepared by:**

Robert Prestridge (EE)

Donald Stith (CpE)

Daniel Sciortino (CpE)

**Faculty Advisor:**

Dr. Khan Iftekharuddin

**Old Dominion University**

*Department of Electrical and Computer Engineering*

*December 04, 2017*

**ODU Honor Pledge:**

“I [we] pledge to support the Honor System of Old Dominion University. I [we] will refrain from any form of academic dishonesty or deception, such as cheating or plagiarism. I [we] am aware that as a member of the academic community it is my responsibility to turn in all suspected violation of the Honor Code. I [we] will report to a hearing if summoned.”

**Abstract:**

The MARCbot IV platform is currently being used by U.S. military personnel in combat situations to remotely observe IEDs from a safe distance. This proposal introduces the vision guided mobile robot for automated surveillance project. The proposed project requires students to construct a program to control the MARCbot autonomously with the assistance of object/facial recognition. This proposal shows the progress toward facial recognition, and the combination of MATLAB and LabVIEW for autonomous operation of the MARCbot IV platform. The use of fully automated robots to identify objects or facial recognition from a pre-determined database will be critical toward a continued surveillance environment. With the use of automated robots in combat situations, ethical questions are raised over the decision-making capabilities, errors produced by the robots, and the responsibility of the manufacturers.

**Table of Contents:**

[List of Figures: iv](#_Toc500147763)

[List of Tables iv](#_Toc500147764)

[I. Introduction 1](#_Toc500147765)

[II. Design Approach 1](#_Toc500147766)

[A. Project Summary 1](#_Toc500147767)

[B. Summary of Design Method 1](#_Toc500147768)

[i.) MARCbot IV 3](#_Toc500147769)

[ii.) LabVIEW 4](#_Toc500147770)

[iii.) OpenCV 4](#_Toc500147771)

[iv.) MATLAB 4](#_Toc500147772)

[C. SUMMARY OF PROGRESS 5](#_Toc500147773)

[a.) Hardware Upgrades 5](#_Toc500147774)

[i.) MARCbot IV Robot Upgrades 5](#_Toc500147775)

[ii.) MARCbot IV Controller Upgrades 5](#_Toc500147776)

[b.) Facial Recognition Algorithm 7](#_Toc500147777)

[c.) LabView Modification 9](#_Toc500147778)

[d.) Driving Toolbox Design 11](#_Toc500147779)

[D. Realistic Constraints 12](#_Toc500147780)

[a.) Electrical Constraints 12](#_Toc500147781)

[b.) Computer Programming Constraints 12](#_Toc500147782)

[E. Overall Summary of Current Operation 13](#_Toc500147783)

[F. Alternative Design 15](#_Toc500147784)

[III. Project Considerations 16](#_Toc500147785)

[A. Ethical Implications 16](#_Toc500147786)

[B. Lifelong Learning 16](#_Toc500147787)

[C. Engineering Standards 17](#_Toc500147788)

[a.) IEEE.1872.2015 Standard Ontologies for Robotics and Automation 17](#_Toc500147789)

[b.) IEEE.15288.1-2014 Standard for Application of Systems Engineering on Defense Programs 17](#_Toc500147790)

[c.) ISO/IEC.14882.2011 Standard for the C++ Programming Language 17](#_Toc500147791)

[d.) ISO/IEC.8802.11 IEEE-STD 802.11 Information technology 18](#_Toc500147792)

[IV. Broader Impact 18](#_Toc500147793)

[A. Global Impact 18](#_Toc500147794)

[B. Economic Impact 18](#_Toc500147795)

[C. Environmental Impact 18](#_Toc500147796)

[D. Societal Impact 19](#_Toc500147797)

[V. Project Contributions 19](#_Toc500147798)

[VI. Conclusion 19](#_Toc500147799)

[Appendix A Executive Summary A-1](#_Toc500147800)

[Appendix B Budget B-1](#_Toc500147801)

[Appendix C Face Recognition and Driving Toolbox Code C-1](#_Toc500147802)

# **List of Figures:**

[Figure 1-MARCbot IV 2](#_Toc500148058)

[Figure 2-MARCbot Controller with DAC interface 2](#_Toc500148059)

[Figure 3-Video Communication Diagram 2](#_Toc500148060)

[Figure 4-Movement Control Flowchart 3](#_Toc500148061)

[Figure 5-Dry Rotted Tires 5](#_Toc500148062)

[Figure 6-Controller Wiring prior to Correction 6](#_Toc500148063)

[Figure 7-As Found Wiring Connections 6](#_Toc500148064)

[Figure 8-Controller Wiring after Corrections 7](#_Toc500148065)

[Figure 9-Wiring using NASA-STD-8738.3 with Heat Shrink Insulation 7](#_Toc500148066)

[Figure 10-Facial Recognition Initial Attempts 8](#_Toc500148067)

[Figure 11-Flowchart of Facial Recognition with LBPH 8](#_Toc500148068)

[Figure 12-Facial Recognition Tested Satisfactorily 9](#_Toc500148069)

[Figure 13-KLT Flowchart 9](#_Toc500148070)

[Figure 14-LabVIEW Manual Control Front Panel 10](#_Toc500148071)

[Figure 15-LabVIEW Manual Control Block Diagram 10](#_Toc500148072)

[Figure 16-LabVIEW using UDP Input 11](#_Toc500148073)

[Figure 17-Distance Based on Pixilation 11](#_Toc500148074)

[Figure 18-Driving Toolbox Flowchart 12](#_Toc500148075)

[Figure 19-Example of Video Feed Distortion 13](#_Toc500148076)

[Figure 20-Use of Facial Recognition with Analog Camera 13](#_Toc500148077)

[Figure 21-Overall Operational Flowchart 15](#_Toc500148078)

[Figure 22-MARCbot IV Robotic Platform and Controller A-1](#_Toc500148079)

# **List of Tables**

[Table 1-Budget for MARCbot IV Upgrades B-1](#_Toc500147825)

# **Introduction**

The headline of news agencies covering combat is rarely positive. The headline of soldiers killed in action can be minimized through the advancement of autonomous robots performing higher risk tasks. Robots can detect IEDs, high value enemy combatants in hostile environments, survey the aftermath of chemical or biological weapons to search for survivors, and potentially administering first aid on the frontline thru the advancement of robotics. This paper will show the advancement of the vision guided mobile robot for automated surveillance toward full autonomous robotics. A previously modified MARCbot IV platform was upgraded with new hardware and software. The use of MATLAB will control the face detection algorithm to acquire the target. Control of the robot is accomplished using LabVIEW, and will interact with the facial recognition algorithm through a UDP (User Datagram Protocol) and driving toolbox. The project has attempted to fully implement an autonomous MARCbot IV robot with the capability of face/object recognition. This robot will have the ability to follow the face/object throughout the space. This proposal will discuss the design approach, project considerations, broader ethical impacts, and project contributions from each member of the team.

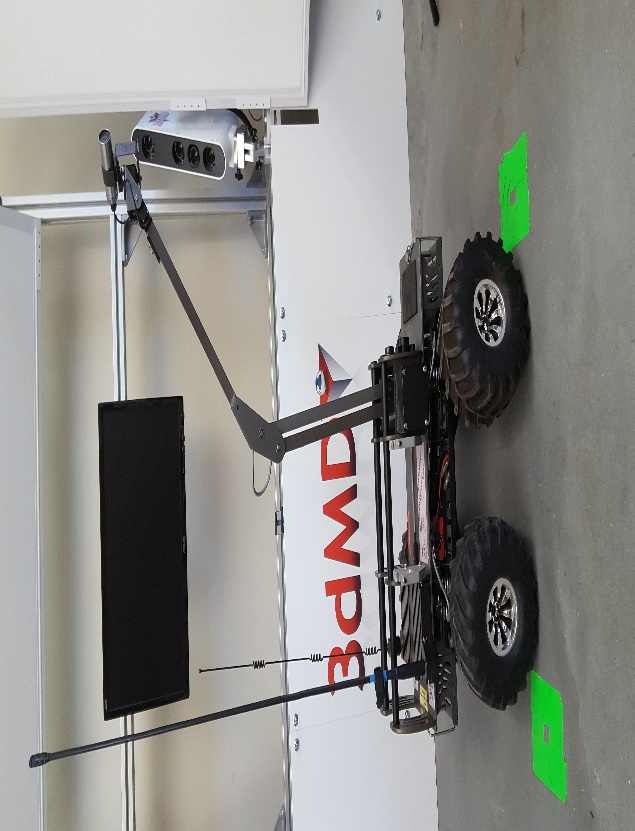
# **Design Approach**

## **Project Summary**

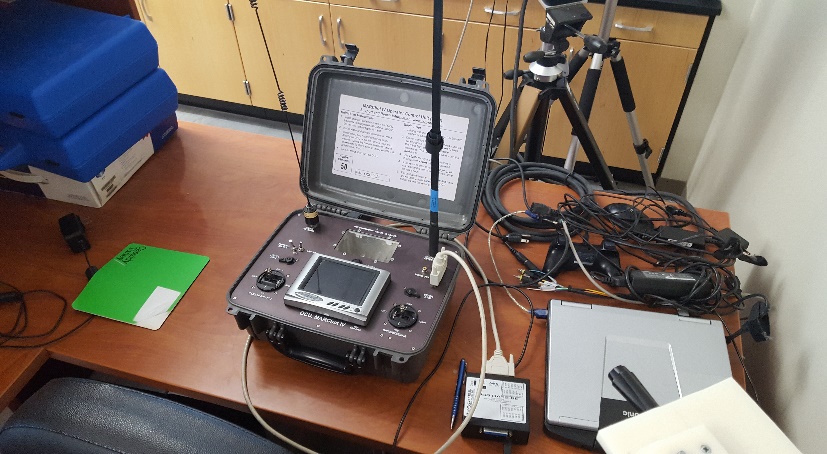
The vision guided mobile robot for automated surveillance uses an all-terrain robot with real-time automated object detection capabilities. The original MARCbot IV platform captures video images, and sends the data to a remote station manned by an operator. This project requires students to develop a program capable of object/facial recognition using computer programs with the ability to communicate navigation controls to the robot for improved object recognition and tracking. The robot will need the ability to navigate toward an object of interest to obtain improved confirmation of the object. Previous experiments using the MARCbot with software programs consisted of corrupted data. Students will be required to re-create the software programs, upgrade existing components, and test the working prototype for continued success.

## **Summary of Design Method**

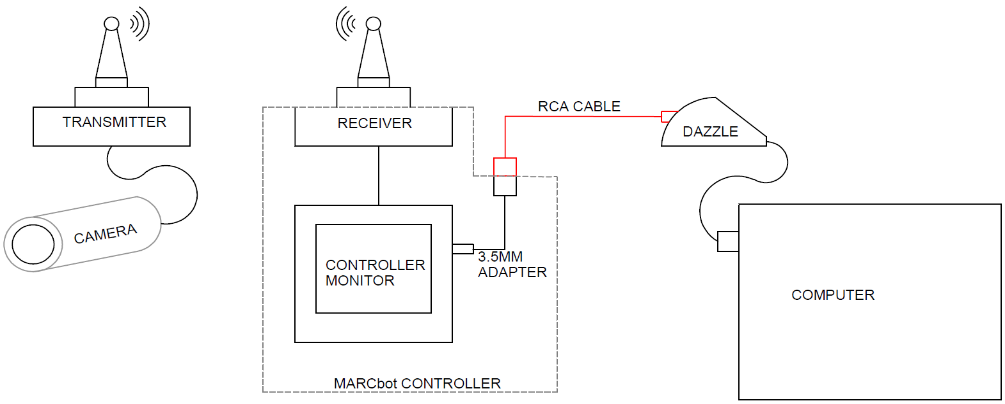
The initial design of the MARCbot IV robotic platform consists of an external control unit to control the robot and view the video feed. Previous projects bypassed the controls of the controller via a DB25 serial port. The control signals are sent via the DB25 serial to a DAC for digital control by the host computer. The automated mechanical controls are operated through a LabVIEW (Laboratory Virtual Instrument Engineering Workbench) program with rheostat DAC controls. The video feed is relayed from the onboard camera to the control unit and displayed on a video monitor attached to the 1.2GHz receiver. During previous projects, the output video receiver was converted from 3.5mm video port to RCA video connector, and then converted from analog to digital with the use of a Dazzle converter for use on the host computer. The video feed can be automatically viewed with C++ programmed OpenCV (Open Source Computer Vision) library or MATLAB (matrix laboratory), allowing for face/object recognition.



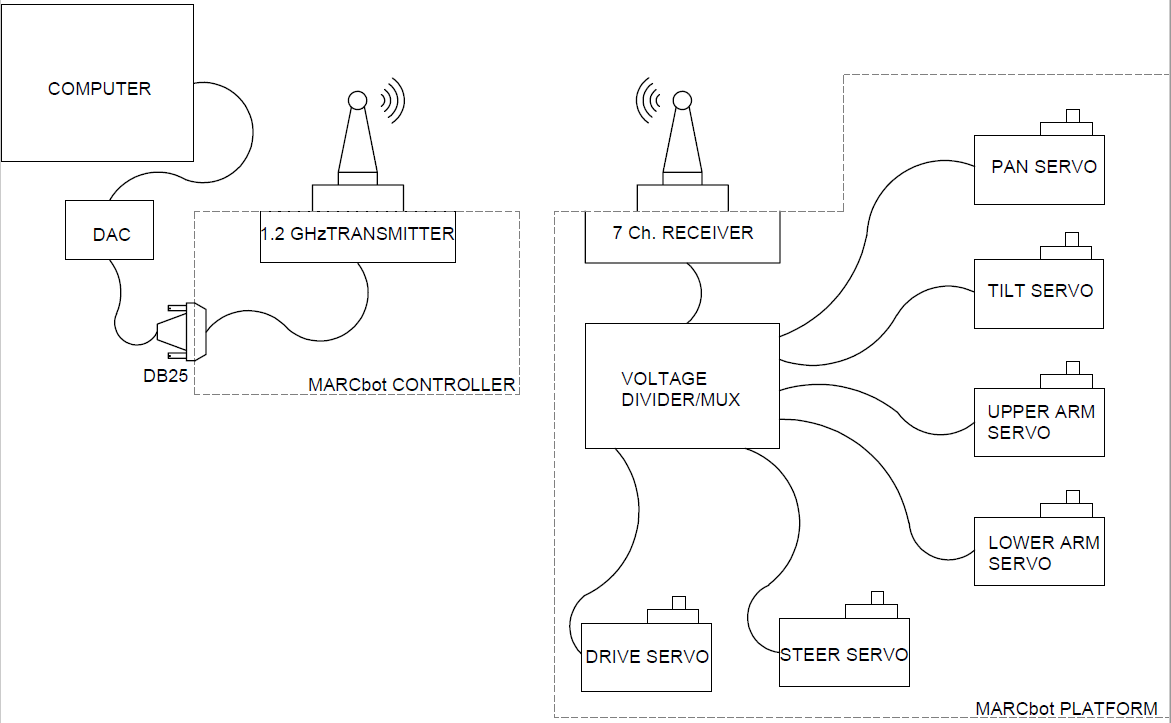
*Figure 1-MARCbot IV*



*Figure 2-MARCbot Controller with DAC interface*



*Figure 3-Video Communication Diagram*



*Figure 4-Movement Control Flowchart*

The first objective of the project is to develop a facial recognition software using C++ with OpenCV libraries, or MATLAB. Facial recognition programs considered are Eigenface, LBPH (Local Binary Pattern Histograms), and Fisher. The video feed will be utilized to lock onto facial features and determine if the face is recognized from a pre-determined database. Continuous facial recognition will allow the robot to lock onto the face within the video frame. To increase the ability of the facial recognition algorithm to recognize the face will require upgrading the onboard robot camera. Due to the age of the robot, other components will also need to be upgraded to ensure operability of the robot and provide backup components to provide reliability when components fail due to age.

The final portion of the project will require developing a driving toolbox based on the facial recognition algorithm. Prior to developing the driving toolbox, LabVIEW will need to be developed to control the robot remotely. The previous project had configured LabVIEW for remote control of the robot, but does not control the robot fully. Modification to the LabVIEW program will need to be made to understand all of the inputs for the DAC to control each servo on the robot. Once the remote controlling program is developed, then the driving toolbox can be developed using a UDP to interface between the facial recognition algorithm and the driving toolbox. When a face is recognized by the facial recognition algorithm, the data is relayed to the driving toolbox which will need to interface with LabVIEW to control the robot via the DAC to maintain the face in the frame at a predetermined depth and location within the video frame. The driving toolbox will need to determine the location of the face within the video feed and provide commands to the robot to move forward and reverse, and turn the robot to center the face in the video frame. This will require maintaining the camera centered on the robot for optimal viewing.

### ***MARCbot IV***

Multi-function Agile Remote-Controlled Robot (MARCbot) was designed for counter improvised explosive device (IEDs) operations in the Iraq War by the United States Army [1]. The initial cost of $10,000 for the MARCbot allowed the design to be a low-cost alternative to available robotic platforms of the time [1]. The MARCbot was developed in 2004 by Exponent, and has seen numerous upgrades since [1]. The MARCbot utilizes a single ACTi IP camera attached to an extendable arm [2]. The adjustable arm allows for a remote viewing platform up to 39 inches in height for viewing into vehicles and around corners [2]. As shown in figure 1, the MARCbot is an all-terrain vehicle capable of remote movement, but is limited in capability due to only having the camera attached to the extendable arm. The remote-control station, as seen in figure 2, can allow the MARCbot to operate at a range of 300m line of sight [2]. The MARCbot controller was modified on a previous project to include a DB25 serial port. The DB25 serial port allows signals to be sent from a controlling computer, through a digital to analog converter, to control the robot operations. Additionally, an RCA input was installed allowing for remote viewing of the video feed being visible from an external source, such as a computer, via RCA cables. The MARCbot is equipped with three separate BT-70791A Li-Ion batteries capable of operating for up to 6 hours [2]. Due to the limited capabilities of the MARCbot, the U.S. Army has discontinued its use, and the robot platform is currently being modified for use within local police principalities [2].

### ***LabVIEW***

Laboratory Virtual Instrument Engineering Workbench (LabVIEW) was developed by National Instruments in 1986 [3]. LabVIEW was developed for engineers and scientists using graphical programming language “G” [3]. LabVIEW uses a dataflow model to produce programs vice text. LabVIEW can be utilized to acquire and analyze measurement data, instrument controls, embedded control and monitoring systems, and automated test and validation systems [3]. LabVIEW interfaces with instruments, cameras, sensors, proximity detectors, and other devices through direct bus commands, or utilizing device-specific drivers from the National Instrument libraries [3].

### ***OpenCV***

Open Source Computer Vision (OpenCV) was created by Intel Corporation in 1999 as a library of functions for real-time computer vision [4] [5]. OpenCV is written in C++ with numerous interfaces with C, Python, Java, and MATLAB [5]. OpenCV can be used for multiple applications, such as facial recognition, gesture recognition, motion tracking, etc. [5].

### ***iv.) MATLAB***

MATLAB (matrix laboratory) is a proprietary programming language developed by MathWorks [6]. Cleve Moler started development of MATLAB at the University of New Mexico in the late 1970s [6]. Moler along with Jack Little, and Steve Bangert founded MathWorks in 1984 [6]. MATLAB was originally written in FORTRAN and C programming languages, but was later updated to include C++ and Java source code [6]. MATLAB has updated numerous libraries to include facial recognition and driving toolboxes [7].

## **SUMMARY OF PROGRESS**

### **a.) Hardware Upgrades**

#### *MARCbot IV Robot Upgrades*

The installed camera was determined to be a ACM-1311 ACTi camera. The camera specifications have a maximum resolution of 350 TVL (~0.35MP), with standard low light sensitivity for indoor use only. Maximum frame rate of the camera was 30 fps at 720x480. Replacement cameras were researched and a decision was made to order a Sony 8 infrared no glow night vision bullet camera. The Sony 8 uses a SC-IR8 DSP camera, with eight no glow infrared LEDs for night vision capability up to 25ft. The increased night vision capability allows for increased accuracy indoors and outdoors, with low level lighting. The Sony 8 has a maximum resolution of 420 TVL, and a frame rate of 45fps at 720x510. Due to size requirements of the mounting bracket on the MARCbot IV, a larger resolution camera was not compatible for purchase. Also, the transmitter operates at 1.2GHz, and is only capable of 525 TVL limiting the resolution of the replacement camera. The current BT-70791A Li-Ion batteries are capable of continuous use up to 6 hours. To increase the operating time of the robot, three BT-70791CK Li-Ion batteries were purchased, and increased the operating time to 7.5 hours. After the new batteries arrived, one of the previous BT-70791A Li-Ion batteries failed to hold a charge. The purchase of the batteries provided additional backup for failure of the previous operating batteries. Due to the age of the robot, the tires show severe signs of dry rot, see figure 5 below. Replacement of the tires was accomplished along with the camera replacement in July.



*Figure 5-Dry Rotted Tires*

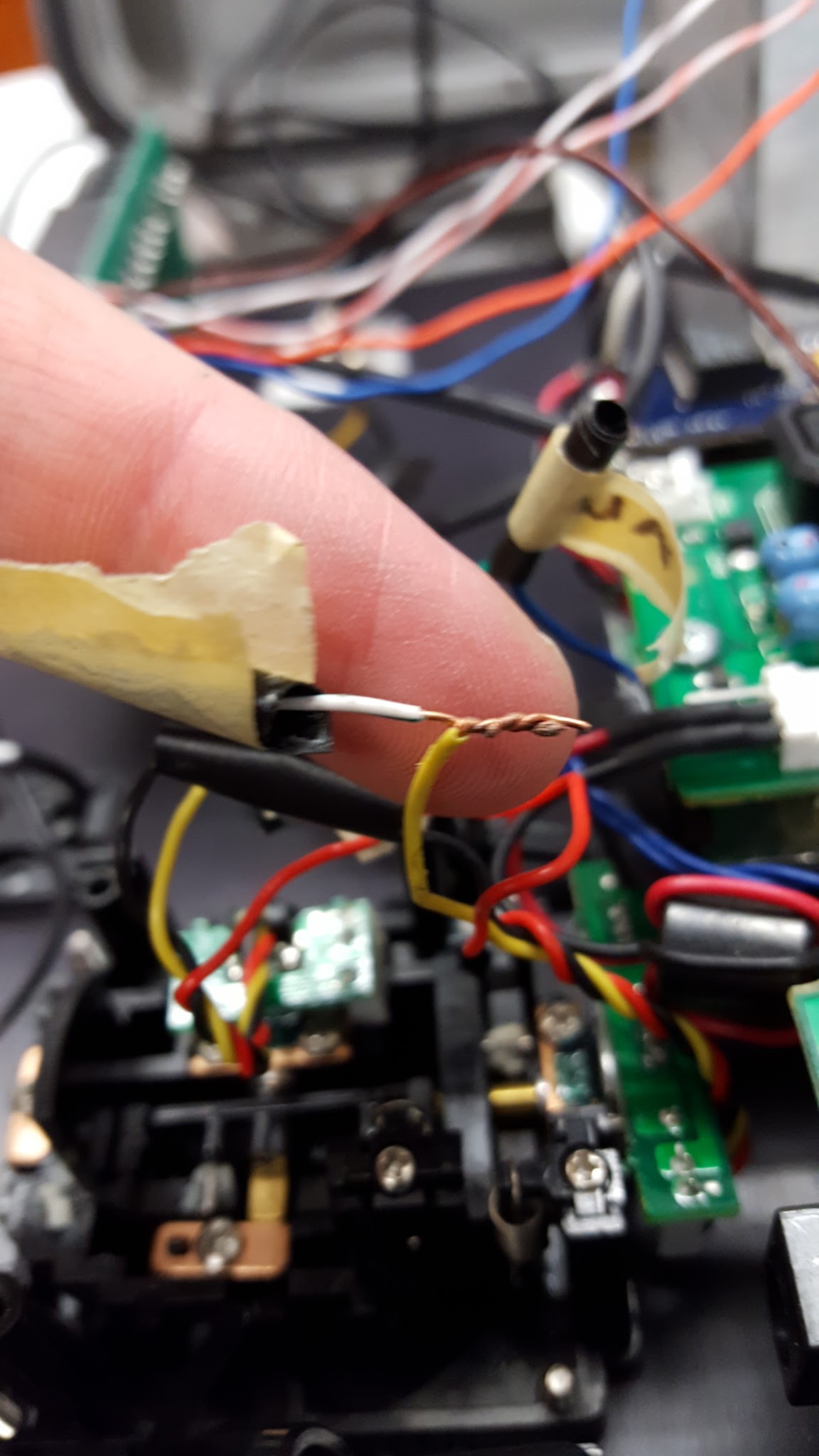
#### *MARCbot IV Controller Upgrades*

The replacement of the digital analog converter (DAC) was a concern, due to the age of the current DAC. Additionally, the benefit of providing backup components for the robot assist in reliability for continued operation. The replacement DAC will only be used for backup operations.

During testing of the facial detection algorithm, control of the MARCbot IV was lost. During troubleshooting of the controller, it was noted that poor workmanship was used to connect the DB25 serial port. Materials were acquired from a local electronics store, and was not be included in the overall project budget. As shown in figure 6 and 7 below, the DB25 serial port was wired directly to the controller, bypassing the controller components. The poor workmanship was visible as two wires were disconnected, including the system ground. The wires were connected by twisting the wires and secured with electrical vinyl tape. Additional markings were made using masking tape, and permanent markers.

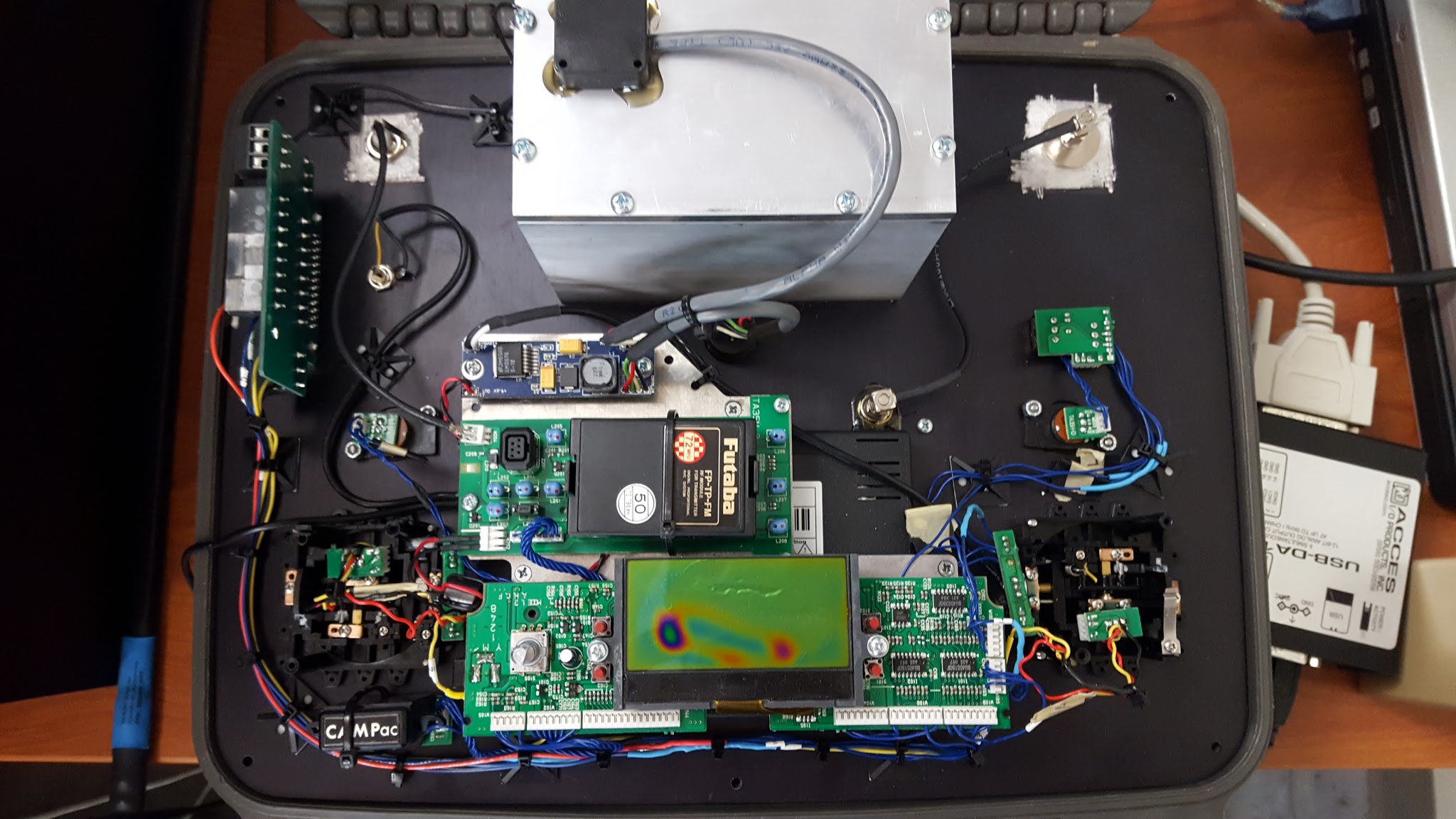


*Figure 6-Controller Wiring prior to Correction*



*Figure 7-As Found Wiring Connections*

The wires were connected and soldered in accordance with NASA-STD-8738.3, for a lineman splice. The connections were then protected, using 3M heat shrink insulation, and routed neatly around the board using ziotek zip tie mounts, and zip ties. The wires were numbered using wire numbering tape to the numbered pin on the DB25 serial port, see figure 8 and 9 below. Also, the 3.5mm to RCA adapter used on the MARCbot controller was found to be TRS (tip, ring, sleeve). This means the 3.5mm adapter only utilizes Left Audio, Right Audio, and ground. The correct 3.5mm to RCA adapter was installed, and consists of a TRRS (tip, ring, ring, sleeve) configuration, which allows for Left Audio, Right Audio, Video, and ground. The video image produced was slightly improved for human eye, but considerably improved the computer recognition capabilities.



*Figure 8-Controller Wiring after Corrections*



*Figure 9-Wiring using NASA-STD-8738.3 with Heat Shrink Insulation*

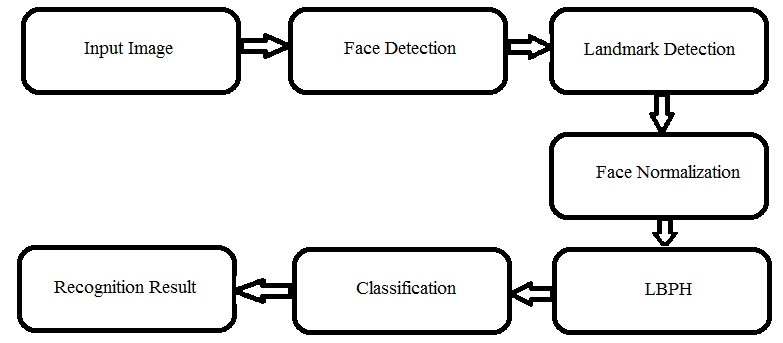
### **b.) Facial Recognition Algorithm**

MATLAB programs used for facial recognition were researched. Additional libraries were used to run facial recognition software in MATLAB. Initial facial recognition software was tested to determine accuracy of the software. Initial attempts at facial recognition failed to distinguish faces from random objects in the room, such as the image in figure 10. Additional attempts using the same program, but with the installed laptop camera, showed that the program could recognize faces more accurately only with increased resolution. The students continued to research how to create a facial database within MATLAB by establishing positive and negative images from a stored database, but accuracy was not acceptable. The decision was made to create a facial database to be shared with competing programs. The first program was being developed by Dan in MATLAB using fdlibmex library, and the second program was being developed by Donald using the LBPH model of the FaceRecognizer in OpenCV. LBPH with OpenCV was finally selected due to many factors, including speed, minimal required resources, and increased knowledge in C++ programming.



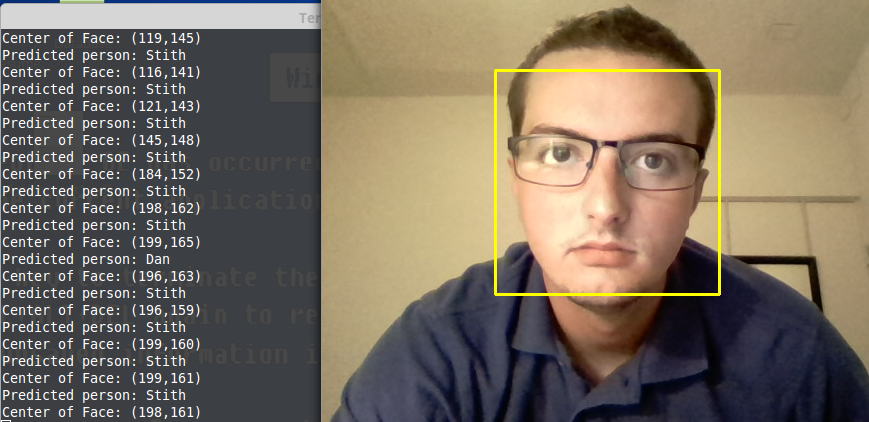
*Figure 10-Facial Recognition Initial Attempts*

The facial recognition algorithm consists of two main components: OpenCV’s CascadeClassifier class, and FaceRecognizer model. The CascadeClassifier is loaded with a “haarcascade\_frontalface\_alt.xml” file provided by OpenCV, which trains the classifier to look for frontal faces. The FaceRecognizer model is also trained with a database of images of the project group members’ faces. The main disadvantage of using LBPH is that the images must be of equal size. To accomplish this, we are using the largest sized image in each facial database, and expanding the other images in the database to match the dimensions of the largest image. When the camera detects a face, the program will search for specific features, such as the eyes, and rotate the image until the eyes are horizontal. Once the eyes are horizontal, the program will take the distance between the eyes and adjust the image, so an equally sized image can be cropped for comparison.



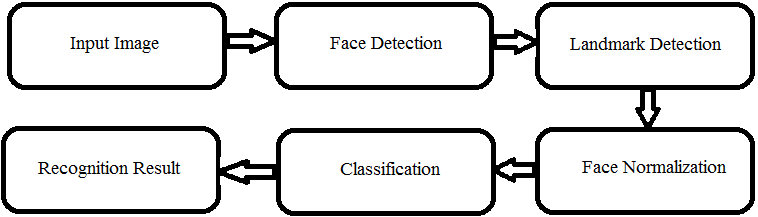
*Figure 11-Flowchart of Facial Recognition with LBPH*

Upon initializing the startup of the program, the FaceRecognizer is trained with the database of the project group members’ face images. After the FaceRecognizer has been trained, the program will then take a snapshot of the webcam feed, and pass it to the CascadeClassifier’s “detectMultiScale” function. This function detects a face within that snapshot, and returns the coordinates of the face in the form of a “Rect” object. That object contains an x and y coordinate for the top-left of the face in the image, as well as height and width of the face. This information is used to crop the face out of the frame, and pass it to the FaceRecognizer model. The FaceRecognizer can then identify the image of the face and label it as one of the three group members, or unknown.



*Figure 12-Facial Recognition Tested Satisfactorily*

Once it was determined that OpenCV did not work with the analog camera feed, the program was converted to MATLAB, using the vision package. The move from OpenCV with LBPH to MATLAB created some problems. The program was very similar, but the algorithm changed from LBPH to KLT (Kanade-Lucas-Tomasi). As shown in figure 13, the change in operation was minimal, as KLT does not use the LBPH to determine facial features for classification. The operation overall is faster.

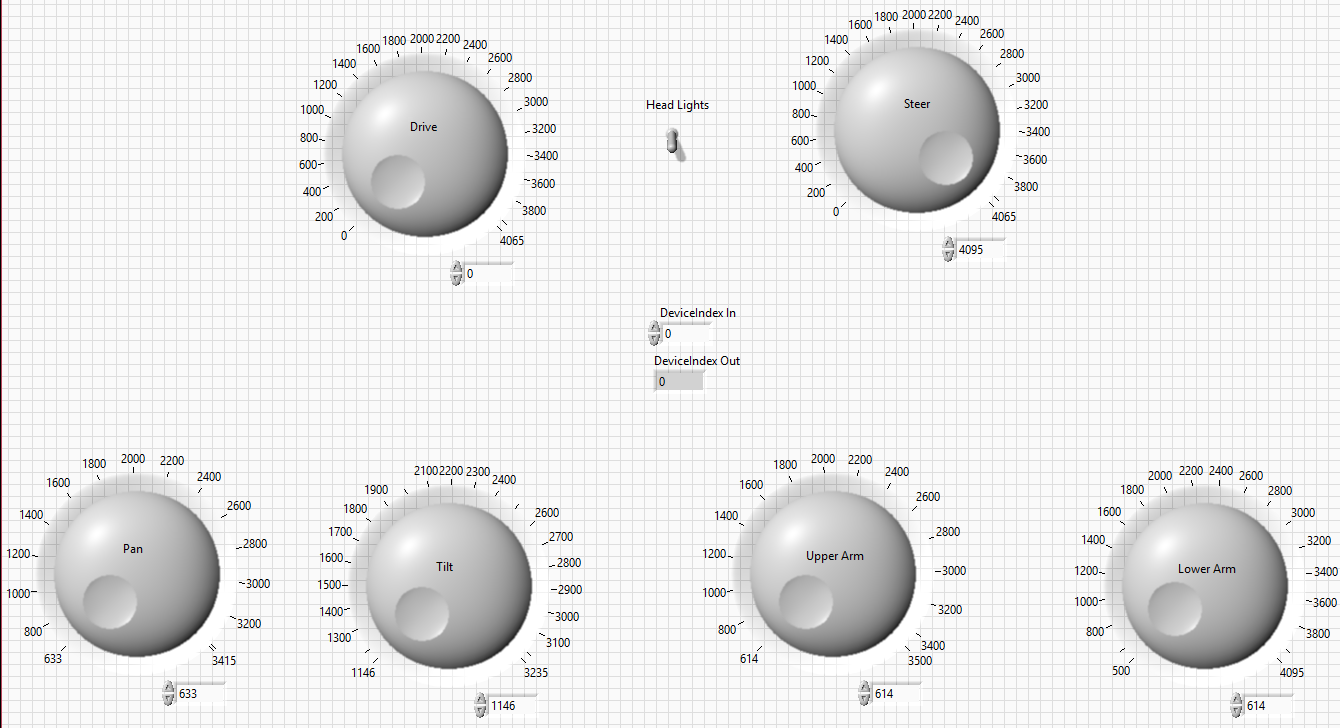


*Figure 13-KLT Flowchart*

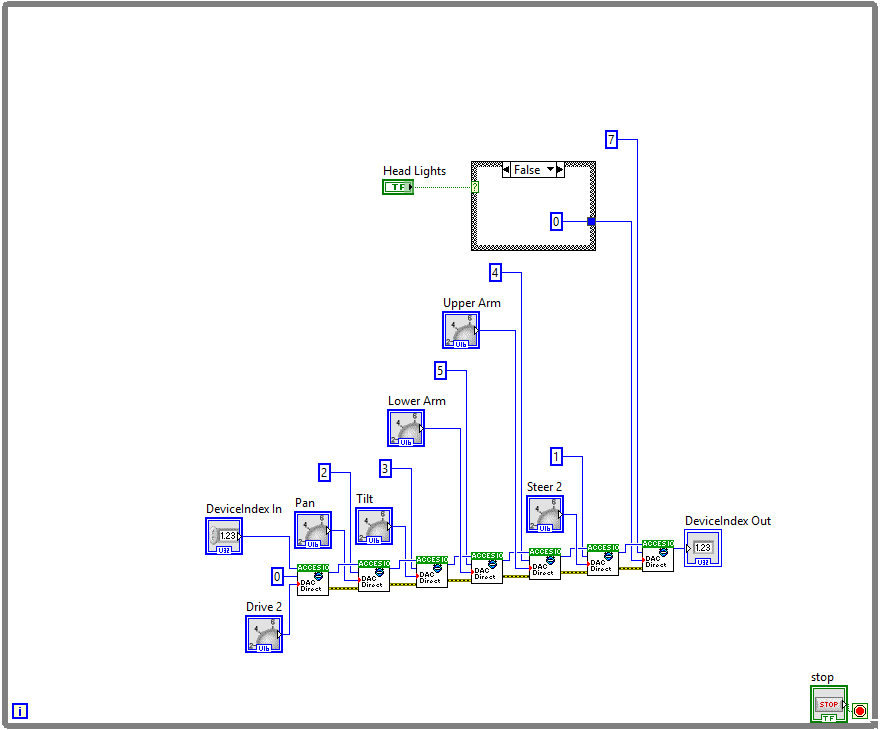
The MATLAB implementation was very similar to the OpenCV implementation. MATLAB’s CascadeObjectDetector was used to detect the faces, like OpenCV’s CascadeClassifier, but the ObjectDetector was already initialized to detect faces, so the FrontalFace xml wasn’t needed. The main difference in the algorithm was the recognition. In order to use MATLAB’s ImageCategoryClassifier, the databases were broken down into “bagOfFeatures” objects, which highlighted the most common features in each category of image (in this case, the different faces the classifier was trying to label). The classifier is then trained using these features, and the training function ignores characteristics that are common across all categories. This allows for more accurate recognition. Finally, the tracking is accomplished using Matlab’s pointTracker object. This object uses the EigenFeatures of the object it is trying to track, and then passes those features to the point tracker. In each iteration, the algorithm checks which points it can still track, and then updates the point tracker. This loop repeats until a minimum number of points is reached, then the program loops back to detection to find another face to track.

### **c.) LabView Modification**

The original LabVIEW program transferred from the previous project contained the values needed to operate each servo. Those values to fully control each servo was crucial to moving forward with the LabVIEW manual control. The overall design for manual control of the robot using LabVIEW was minimal, only requiring the remapping of the upper arm, lower arm, pan, and tilt servos. The updated manual control of the robot is shown below in figure 14 and 15.

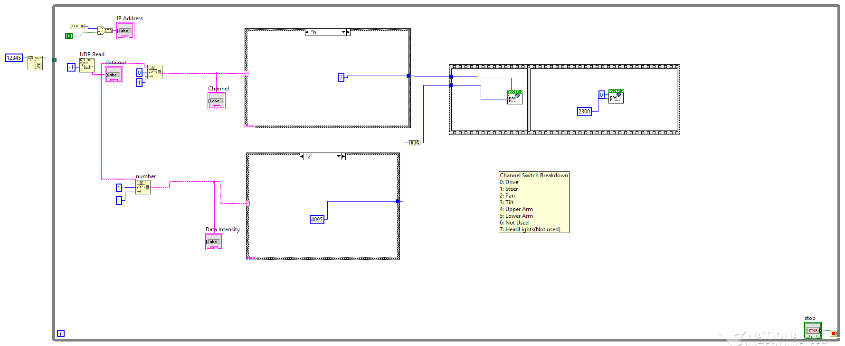


*Figure 14-LabVIEW Manual Control Front Panel*



*Figure 15-LabVIEW Manual Control Block Diagram*

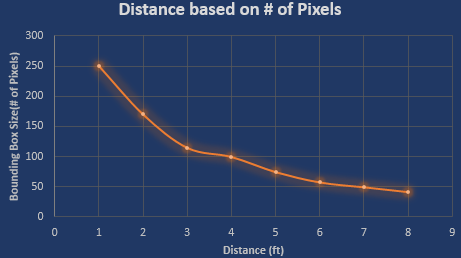
To move forward with autonomous movement, the LabVIEW modification required the addition of a UDP (User Datagram Protocol) support.  The UDP update allows the end user to operate the MARCbot remotely and use a single host computer as the controller. Updating the commands allows the Driving ToolBox to have more control over the MARCbot.  The updated commands allow not only the MARCbot to reposition itself but to control the camera that is attached to the arm. Each movement servo is designated to a specific channel, with an intensity selected 0 to 12. The eight channels include: drive (Ch. 0), steer (Ch. 1), pan (Ch. 2), tilt (Ch. 3), Upper Arm (Ch. 4), Lower Arm (Ch. 5), Blank (Ch. 6), and headlights (Ch. 7), as shown in figure 16. The use of a single servo for the drive and steer requires the use of a midpoint intensity to ensure no movement and testing the various intensities on either side for controllable intensities for future use.



*Figure 16-LabVIEW using UDP Input*

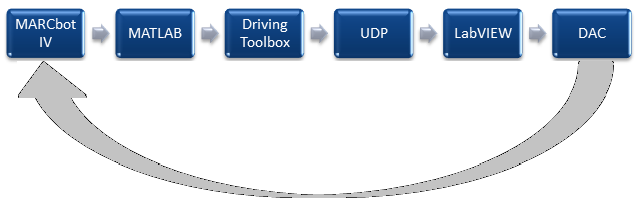
### **d.) Driving Toolbox Design**

The Driving Toolbox, consists of analyzing the image and getting the X and Y coordinates of the rectangle that is around the person of interest face. MATLAB returns the upper left x and y coordinate of the face as-well-as the height and width of the box. This information is processed to find the distance in pixels in the x and y space. This is then is used to send the command via UDP with an intensity value to LabVIEW. Besides measuring the distance from the center of the face is away from the center of the frame, the Toolbox also measures the diagonal distance of the rectangle.  This is used to determine the rough distance that the MARCbot is from the person of interest. The graph below in figure 17, shows the pixel box size for images in the frame from 1ft to 8ft.



*Figure 17-Distance Based on Pixilation*

Before passing the value to the LabVIEW program, it gets added into a memory of the last 10 values.  This is used to make sure the MARCbot isn't getting the same values and can reposition itself for new commands.  The Toolbox prioritizes local movements with the arm before the MARCbot tries to reposition itself.



*Figure 18-Driving Toolbox Flowchart*

If the face is recognized as one of the three group members, then we can proceed tracking the face using the driving toolbox. Otherwise, we will loop back to taking a snapshot of the webcam feed, detecting the face, and attempting to recognize again. The MARCbot will only move if it recognizes the object/face. Otherwise it will remain stationary.

## **Realistic Constraints**

### ***a.)******Electrical Constraints***

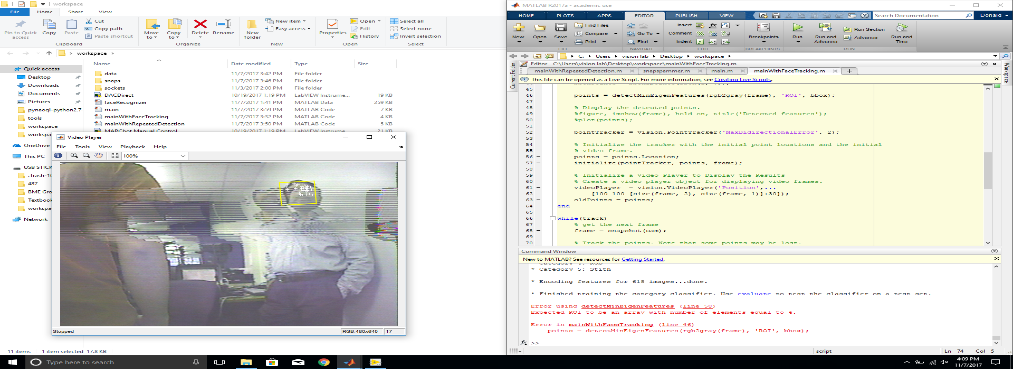
The MARCbot IV has been upgraded and subsequently discontinued for further use by the U.S. Army. Initial MARCbot IV design and construction was contracted to Exponent Inc [1]. In 2009, NASA modified the original MARCbot IV design for use as relay points for a wireless communications network on the Moon. The new modification was contracted to Applied Geo Technologies Inc. and was reclassified as MARCbot IV-N [8]. When looking for electrical drawings for upgrading electrical components, numerous contractors were reluctant to assist due to the classification of the robot by the U.S. Army. Due to the expiration of the government contracts held by Exponent, they no longer maintain drawings for the MARCbot IV. Exponent was only able to assist in providing the wiring diagram for the camera mounted on the movable arm. Applied Geo Technologies Inc. has since gone out of business, and no available drawings can be located from the modified MARCbot IV-N model. The technologies used by Exponent are antiquated due to the development of this robot in 2004, and upgrading of the robot required visual verification of individual components.

### ***b.) Computer Programming Constraints***

The first constraint of the computer program software used in manipulating the MARCbot IV is the lack of training on these systems by the project members. Initial operations on the MARCbot IV were performed with the linked application of LabVIEW for the robot controls, and MATLAB for the video feed. Both computer engineering students were required to learn how to use LabVIEW and MATLAB for the toolboxes used on this robot. The students were familiar with C++ programming, but had minimal experience with OpenCV libraries. Assistance from the TA Alexander Glandon was required to move forward with the new programs. This is the reason also the reason for the switch from MATLAB to C++ with OpenCV libraries initially.

Another restraint is the use of 3 separate operating systems to program the robot. The original files were programmed in Windows, the facial recognition algorithm was programmed in Linux, and the driving toolbox was programmed on Mac IOS. Transferring the files between computers present problems with drivers for operating the various peripherals. Currently the driver for displaying the dazzle converter on Linux is causing issues with connection to the facial recognition database. To correct this issue, all operations were moved to the lab computer using windows. Once this problem was solved, we increased the facial database to include various lighting and distances. The use of an analog camera was incompatible with OpenCV, and conversion back to MATLAB was required.

Lastly, the use of an analog camera creates some issues with communications. The signal from the robot is sent at 1.2GHz, and received by the controller. The controller then converts the signal from a 3.5mm output to RCA video feed, and finally converted to digital by the use of a Dazzle. The multiple conversions and the use of analog communications creates aliasing of the signal. The overall result of this distortion creates lines in the video feed creating difficulty for MATLAB to continuously track a face in the frame, as shown in figure 19 and 20. Another problem is the latency timing between MATLAB and LabVIEW. Pausing the program within MATLAB creates loss of the face recognition algorithm, and pauses in LabVIEW created inability of the program to process the commands. The latency was not able to be overcome.



*Figure 19-Example of Video Feed Distortion*

**

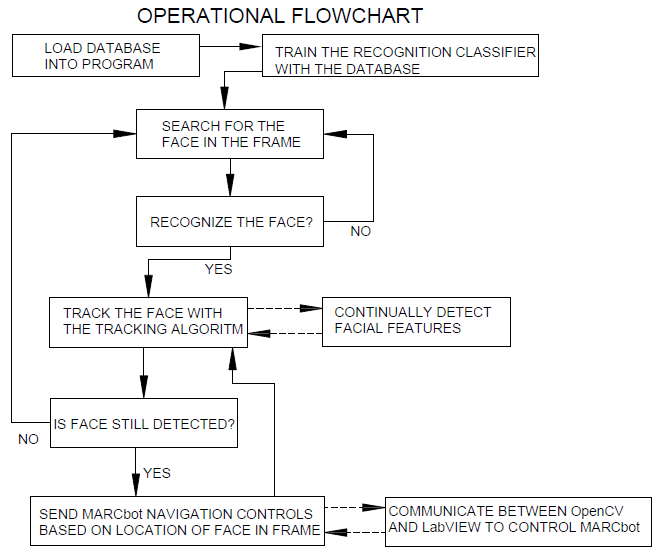
*Figure 20-Use of Facial Recognition with Analog Camera*

## **Overall Summary of Current Operation**

On Jan 24th, the group was introduced to the MARCbot IV robot, and the associated corrupt programs used by previous projects. The previous face recognition software would not compile and was scraped immediately. The previous LabVIEW program was incomplete, and would only control the driving of the robot. Initial startup of the LabVIEW code caused the robot to violently move forward, almost crashing into one of the members of the group and the wall. Immediate operator action was required to stop the robot from damage. After initial evaluation, the group looked into competing facial recognition algorithms using MATLAB, and C++ with OpenCV libraries. The first task was to look into the advantages and disadvantages of Eigenface, LBPH, and Fisher. After much consideration, LBPH was selected along with C++ using OpenCV libraries to control the facial recognition software. At the same time, the LabVIEW program was modified to include the upper arm, lower arm, tilt and pan of the camera for manual control.

The facial recognition algorithm was satisfactorily tested on a separate laptop, and was ready for implementation with the robot. The robot was still awaiting parts to upgrade the camera and other hardware components. Upon receiving the camera, the robot hardware was upgraded, and communication with the robot was lost. During troubleshooting, it was discovered that poor workmanship on the MARCbot IV controller led to failure of the communication between the robot and the controller, as shown in figure 6 and 7. Also during troubleshooting of the robot, it was discovered that one of the piezoelectric crystals that assists with the control transmission at 92GHz had failed. A replacement crystal was ordered and installed, and the control to the robot was restored.

The facial recognition algorithm was next moved to be implemented with the MARCbot IV robot, and build a database with the newly installed camera. This is when it was realized that the OpenCV libraries used, were not compatible with the analog video feed. Immediately, the code was transferred to MATLAB, and modified to KLT, as required for use in MATLAB. The largest change to the algorithm was the output x and y coordinates for the driving toolbox. The driving toolbox was written during the testing of the facial algorithm in C++, and was quickly re-written in MATLAB. The x and y coordinates will vary continuously with the distance and movement of the face in the frame. To overcome this, the distance of the face was based on the pixilation of the face, and split the frame into quadrants for input to the turning mechanism. The overall operation flowchart in figure 21, shows how the overall operation of the autonomous robot will work.



*Figure 21-Overall Operational Flowchart*

The final obstacle to overcome is the latency issue between MATLAB and LabVIEW. Initial design to the driving toolbox placed pauses in the MATLAB code to ensure enough time to send signals via LabVIEW to the robot for operation. The pauses would then cause the facial recognition algorithm to lose the face in the frame and the robot would continue operating on the last signal sent. Pauses were then moved from MATLAB and incorporated into the LabVIEW matrix, but now the robot will take the last signal, and continually loop the signal without correction. Continued troubleshooting is being performed, but overall operation of the robot may lack automation by the end of the semester. The view of the program within MATLAB shows correct operation, but the robot is not responding to MATLAB, so the current belief is an issue with the UDP or LabVIEW matrix.

## **Alternative Design**

Various methods of controlling the MARCbot through Qt and LabVIEW with a backend of MATLAB or OpenCV were researched. It was determined that the combination of LabVIEW and OpenCV would help give the team the expected results in an object detection and automated surveillance. It was only the discovery of program incapability of OpenCV and analog video feed that led to moving the project into MATLAB.

Another design aspect for the MARCbot IV robot, is the upgrade performed by NASA. As discussed previously, the MARCbot IV-N was created in 2009 and implemented by the U.S. Army for operations in Iraq and Afghanistan. Currently, the MARCbot IV is remotely operated via a remote control unit with a digital analog converter interface to the computer. The improved MARCbot IV-N removes the control unit, and allows for direct communication with the robot from the computer with controlled operation of the robot via a game controller attached to the computer. However, the upgraded version is only available to law enforcement and government agencies. The replacement of the control transmitter/receiver was considered for operating directly from the computer, but compatible replacements were not found. The drawings for the MARCbot IV-N were sought, but found to be classified by the military.

Alternative face recognition models were considered, such as Eigenface and Fisherface. These models require all training images to be the same size. LBPH analyzes each image in the database separately, and compares new images to the database. Due to the independent comparisons, LBPH worked better in different lighting environments than Eigenface or Fisherface. KLT was only used due to the implementation of MATLAB and the facial recognition algorithm being default.

# **Project Considerations**

## **Ethical Implications**

In 2007, the US was deploying over 5,000 robots in Iraq and Afghanistan [9]. This was a direct result of 40% of US armed forces casualties due to improvised explosive devices (IEDs) [9]. By the end of 2007, it was estimated that over 10,000 IEDs were defused or neutralized with robot assistance [9]. Robots offer numerous benefits to the military in providing remote ground reconnaissance, bomb defusing, remote bomb detonation, aerial reconnaissance, and combatant confrontation [10]. The use of robots with object recognition capabilities could lead to the use of robots identifying potential improvised explosive devices without operator aid. The result of this technology could lead to more military lives saved, and the improved ability of recognizing IEDs due to the accuracy of programs used by robots. Another benefit from the project would be the ability of the robot to use facial recognition from a predetermined facial database to identify and track known enemy combatants. This would allow soldiers the ability to remotely enter known enemy territory to determine if key figures are present without risking lives. The use of armed robots using facial recognition software brings about complicated ethical issues.

## **Lifelong Learning**

**Robert Prestridge –** As a student Navy veteran seeking a second degree in Electrical Engineering, any accomplishment I can make toward the advancement in safety to our armed forces is exciting. Through previous projects in my education at Old Dominion University, I have learned how to work well with my current partners and hope to learn from their lead. I am learning the benefits of robots in every aspect of electrical engineering fields. This project has help me gain experience working with peers on a specific engineering based design outside my area of expertise.

**Donald Stith –** This project has given me experience with the automation of robotics. I feel that knowledge of automation can be applied to many different projects and will help in the future. This type of vision-based movement was difficult to learn, and provided some very challenging experiences. This project has also introduced us to OpenCV libraries, and extensive MATLAB operation. The experience from using these libraries will also play an important part in future projects, especially if they involve vision-based input.

**Daniel Sciortino –** What I have gained from this project is a better understanding of the OpenCV framework, and an understanding of Deep Learning. This understanding will help with any future projects that involve Computer Vision.  As well as being able to implement Deep Learning with any project will be able to help with more accurate prediction of objects. This better understanding will help me with fields outside of computer engineering that I am less familiar with.

## **Engineering Standards**

### ***a.) IEEE.1872.2015 Standard Ontologies for Robotics and Automation***

This standard establishes the base ontology for design vocabulary and communication of robotics with automation [11]. This standard sets the base vocabulary utilized for manufacturing, end use, and robot integrators [11]. The base vocabulary is determined by the operating system utilized to produce robot motion and automation [11]. This includes the robot processing device, interface, communication, robot commands, sensors, and robotic environments [11]. This standard establishes the substructure of various operating systems such as ALFUS (Autonomy Levels for Unmanned Systems), CORA (Core Ontology for Robotics and Automation), ORA (Ontology for Robotics and Automation), POS (Position and Orientation Systems), and SUMO (Suggested Upper Merged Ontology) [11].

### ***b.) IEEE.15288.1-2014 Standard for Application of Systems Engineering on Defense Programs***

This standard introduces key concepts and application of engineering projects with the Department of Defense [12]. This standard includes the planning, acquisition, modification, and sustainment of defense systems across the life cycle of the defense system [12]. The basis of this standard is to supplement the acquirer-supplier agreement mode of defense contracts, but can assist with other organizations or projects dealing with defense systems [12]. This standard sets the baseline agreement of defense contracts, while allowing for engineering advancement processes [12]. This standard includes processes used by system suppliers, contractors, government system developers, integrators, maintainers, and sustainers [12].

### ***c.) ISO/IEC.14882.2011 Standard for the C++ Programming Language***

This standard specifies the requirements for implementing C++ programming language using standard libraries [13]. The standard introduces key differences between C and C++ programming language through introduction to C++ program behaviors [13]. This standard provides structured programming and object-oriented programming in C++ language [13]. This standard only provides the guidelines for writing the structure and basic programming, and will not provide all-inclusive assistance when writing C++ programming language.

# ***d.) ISO/IEC.8802.11 IEEE-STD 802.11 Information technology***

This standard defines the physical characteristics for wireless local area networks [14]. The standard contains five physical layer units consisting of: four radio units, operating in the 2400-2500 MHz band and bands composed of 5.15-5.25 GHz, 5.25-5.35 GHz, and 5.725-5.825 GHz, and baseband infrared (IR) units [14]. This standard defines the protocol and interconnection of data through communication equipment connected through air, radio or infrared medium [14].

# **Broader Impact**

## **Global Impact**

The use of robots feeds into public fears of Hollywood based action movies. Public perception of robots stems from various movies such as “Terminator”, "iRobot”, “Wall-E”, and “The Matrix” to name a few [9]. The thought of fully autonomous robots brings public resentment to questions of the ability of robots to revolt against their designers [9]. These fears are rooted in difficult ethical questions that are strongly debated as the technology catches up with fiction [9]. First and foremost, who is to be held responsible when an autonomous robot makes uncalculated errors resulting in damage to other equipment, and injury to personnel [9]? In military operations, can robots make ethical considerations to distinguish civilian and enemy combatants? Can a system be 100% tested for accuracy for autonomous operation, when human counterparts cannot be trusted to perform as accurately [9]?

## **Economic Impact**

The economic impact coincides with the ethical impact of using robots in wartime. The remote observation of potential IEDs by inexpensive robots saves military lives, along with civilian lives. The ability to mobilize remote observation robots can prevent human exposure to nuclear, biological, or explosive environments [9]. These environments would leave minimal effect on the robots, or replacement of the robots would result in lower costs than human life.

## **Environmental Impact**

The MARCbot uses three BB-2590/U 7.5 Ah Rechargeable Lithium-Ion batteries. Lithium batteries can easily catch on fire if exposed to water, and become hazardous when the casing has become damaged [15]. Lithium batteries also contain additional toxic materials, such as lead, copper, and nickel alloy [16]. Exposures to these materials are closely regulated during manufacturing of batteries, but there is limited regulation in the disposal of these batteries [16]. Metals recovered from lithium-ion batteries can be recycled; this process can be complicated by the modifying of the battery chemistry during the manufacturing process [15]. Also, many manufacturers refuse to incorporate recycled materials, due to the possibility of reduced output. While some components of the batteries can be removed and recycled, most of the time they are no longer battery-grade and instead used for other products [15]. Most lithium-ion batteries will be slated for hazardous waste disposal, resulting in incineration or storage in landfills [16].

## **Societal Impact**

The use of robots in wartime brings multiple ethical considerations. The unit cohesion within a group of soldiers working alongside robots deteriorates [9]. The use of militarized robots to combat enemy combatants leads to even more ethical questions. Can a robot make the distinction between civilian, friendly, and foe? In April 2008, it was reported that several TALON SWORD units (semi-autonomous robots armed with machine guns) operating inside Iraq were suspended after anonymous reports of a friendly fire incident [17]. Immediate operator action prevented loss of human life, but trust of robot counterparts was lost [17]. In 2007, a South African National Defense Force was using a computerized air defense system, when a computer malfunction caused the gun to swivel uncontrollably and kill 9 soldiers and wounding 14 others [18]. The air defense system uses radar and laser detection to acquire enemy aircraft, automatically fire and reload when the magazine empties [18]. Lastly, the use of robots in combat removes the ability of soldiers to gain the psychological advantage over noncombatants, allowing for the increased intelligence gathering required to gain technical advantage over the enemy [9].

# **Project Contributions**

The group consists of 3 members who are currently studying computer and electrical engineering. Daniel Sciortino is a computer engineer and worked on the driving toolbox for the MARCbot IV robotic platform, using MATLAB and LabVIEW. Donald Stith is a computer engineer that developed the facial recognition programming using LBPH in OpenCV and then converted it to MATLAB. Robert Prestridge is an electrical engineer that led in the documentation of the project, and upgrading hardware for the MARCbot IV.

# **Conclusion**

In conclusion, the vision guided mobile robot for automated surveillance project team has made progress toward its final goal. The decision for use of LBPH in C++ OpenCV Libraries to develop facial recognition provided acceptable results. Multiple conflicts led the team to move the program into MATLAB, and set the team back. Other hardware issues led to trouble with the overall schedule. The team came together and worked through each obstacle. A lack of software training hindered initial progress, but determination drove the team to work throughout the summer toward our goal. The driving toolbox has brought new obstacles to the final project, and as of writing this proposal, the driving toolbox is producing erroneous results. The team will continue troubleshooting until the final presentation. The ability to adapt and collaborate on each software package has assisted the team in accomplishing its current goals.

**References:**

|  |  |
| --- | --- |
| [1] | Exponent, "MARCbot," Exponent Inc., [Online]. Available: http://www.exponent.com/experience/marcbot/?pageSize=NaN&pageNum=0&loadAllByPageSize=true. [Accessed 18 Mar 2017]. |
| [2] | W. L. Miller, "USMC Ground Robotics Current and Desired Future Capabilities," Defense Technical Information Center. |
| [3] | National Intruments Corporation, "LabVIEW," National Instruments, 2014. [Online]. Available: http://www.ni.com/labview/. [Accessed 18 Mar 2017]. |
| [4] | K. Pulli, A. Baksheev, K. Kornyakov and V. Eruhimov, "Realtime Computer Vision with OpenCV," Queue, Seattle, WA, 2012. |
| [5] | G. Bradski and A. Kaehler, Learning OpenCV: Computer vision with the OpenCV library, O'Reilly Media, Inc, 2008. |
| [6] | C. Moler, "The Origins of MATLAB," MathWorks, 2004. [Online]. Available: https://www.mathworks.com/company/newsletters/articles/the-origins-of-matlab.html. [Accessed 18 Mar 2017]. |
| [7] | MathWorks, "Face Recognition," MathWorks, [Online]. Available: https://www.mathworks.com/discovery/face-recognition.html. [Accessed 18 Mar 2017]. |
| [8] | D. Lockney, "Robots Save Soldiers' Lives Overseas," NASA Spinoff, 2009. [Online]. Available: https://spinoff.nasa.gov/Spinoff2009/ps\_2.html. [Accessed 19 Mar 2017]. |
| [9] | P. Lin, G. Bekey and K. Abney, "Robots in War: Issues of Risk and Ethics," California Polytechnic State University, San Luis Obispo, CA, 2009. |
| [10] | R. C. Arkin, "Ethical Robots in Warfare," *IEEE Technology Society Magazine,* vol. 28, no. 1, pp. 30-33, 2009. |
| [11] | Standing Committee for Standards Activities, "IEEE.P1872-2015 Standard Ontologies for Robotics and Automation," IEEE Robotics and Automation Society, New York, NY, 2015. |
| [12] | Software & Systems Engineering Standards Committee, "IEEE.P15288.1-2014 Standard for Application of Systems Engineering on Defense Programs," IEEE Computer Society, New York, NY, 2014. |
| [13] | ISO International Standard, "Programming Languages C++," ISO/IEC 14882:2011(E), 2011. |
| [14] | LAN/MAN Standards Committee , "Information technology-Telecommunications and information exchange between systems-Local and Metropolitan area networks," IEEE Computer Society, New York, NY, 2011. |
| [15] | C. Hand, "Wolters Kluwer," 02 Oct 2013. [Online]. Available: https://app.croneri.co.uk/feature-articles/dealing-waste-lithium-batteries-0. [Accessed 19 Mar 2017]. |
| [16] | D. H. P. Kang, M. Chen and O. A. Ogunseitan, "Potential Environmental and Human Health Impacts of Rechargeable Lithium Batteries in Electronic Waste," *Environmental Science & Technology,* 04 Feb 2013. |
| [17] | L. Page, "US war robots in Iraq 'turned guns' on fleshy comrades," *The Register,* p. 6, 11 Apr 2008. |
| [18] | N. Shachtman, "Wired," 18 Oct 2007. [Online]. Available: https://www.wired.com/2007/10/robot-cannon-ki/. [Accessed 19 Mar 2017]. |
| [19] | Design Automation Standards Committee, "IEEE.P1666-2011 Standard for Standard SystemC Language Reference Manual," IEEE Computer Society, New York, NY, 2011. |

**Executive Summary**

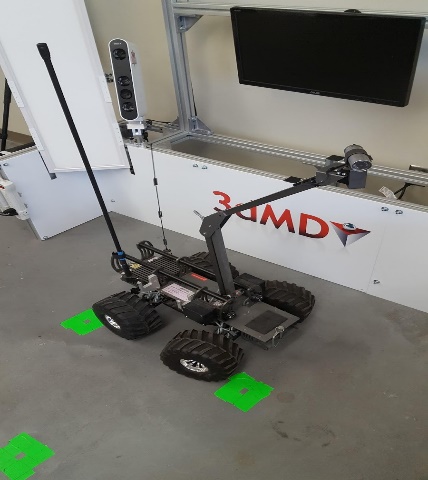
**Vision Guided Mobile Robot for Automated Surveillance**

**Old Dominion University**

**Electrical and Computer Engineering Department**

Robert Prestridge, Donald Stith, Daniel Sciortino

Advisor:  Dr. Khan Iftekharuddin



*Figure 22-MARCbot IV Robotic Platform and Controller*

**Goal of Project:** The goal of this project is to design and implement real-time automated object detection and tracking system for the robotic platform MARCbot IV. The capabilities of the robotic platform include remote navigation and observation through real-time video feedback from a pan-tilt camera mounted on an extendable arm. Currently, the robotic platform has been used to implement limited scale object detection and tracking capabilities using its pan-tilt camera. The planned project includes: the development of a vision-based navigation system for the MARCbot IV platform, the installation of an algorithm that detects and recognizes human face, or objects of special interest, and identification of the object location in the visual frame and usage of the location information to send navigation control signals using the aforementioned navigation system.

**Summary of Progress:** Foremost, hardware upgrades were required to facilitate continued operation of the 2004 produced robot. The wheels were deteriorating, and required replacement. Batteries were showing signs of fatigue, and required backup replacement. The digital to analog converter used for remote control of the robot movements by the computer showed signs of misuse. Lastly, the Camera installed on the robot was upgraded from 0.3 megapixels to 450 TVL. Due to transmission rate of 1.2 GHz, the camera was limited to below 525 TVL. Next, we commenced on determining the facial recognition algorithm. Eigenface, LBPH, and Fisher were considered for the facial recognition algorithm. Due to the ability to detect a face moving within frame, LBPH was chosen for its ability to adjust to the face within the frame. The facial recognition algorithm is currently in testing phase with 70% accuracy. The use of LabVIEW to control the robot was already present from previous projects, so the group will interface using a driving toolbox to interface via UDP and MATLAB to control the robot based on the facial recognition algorithm.

**Impact:** Currently, the United States military uses multiple ground robotic platforms for improvised explosive device detection and remote reconnaissance. The use of cheap robotic platforms removes human counterparts from the dangers associated with the mission critical roles performed by the robots. The use of fully automated robots to identify objects or facial recognition from a pre-determined database will be critical toward the continued combat environment. Fully automated robots can potentially increase the accuracy of object and facial recognition without putting human lives at risk.

**Budget**

Table 1-Budget for MARCbot IV Upgrades

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SENIOR DESIGN BUDGET | | | | | |
| DATE | DESCRIPTION | AMOUNT | MERCHANT NAME | SHIPPING & HANDLING | BALANCE |
| 3/28/2017 | Beginning Balance | - | Old Dominion University | - | $2,600.00 |
| 4/10/2017 | Sony Mini 550 Res Bullet Camera | -$179.00 | StuntCams.com | $0.00 | $2,421.00 |
| 4/10/2017 | USB-DA12-8E (DAC) | -$349.00 | Accesio.com | $0.00 | $2,072.00 |
| 4/10/2017 | BT-70791CK Batteries (x3) | -$1,095.00 | Bren-Tronics.com | -$7.59 | $969.41 |
| 5/1/2017 | Jumbo Kong Tires for T-Maxx (x2) | -$300.00 | HobbyChimp.com | $0.00 | $669.41 |
| Total | - | -$1923.00 | - | -$7.59 | - |

**Face Recognition and Driving Toolbox Code**

%% Clear the screen before the start of the program

%% Define the global constants for the x and y distances

clc;

%% This is the face detecting object

faceDetector = vision.CascadeObjectDetector();

%% Open the webcam stream

if ~exist('cam','var')

cam = webcam(2);

end

%% Set up an internal Connection to the LabVIEW Layer

%{

The commands will be sent accross in the following format

fprintf(u,'x1'), Where u is the connection, and 'x1' is the char array

%}

u = udp('127.0.0.1',12345);

u.EnablePortSharing = 'on';

fopen(u);

% Inital State

disp('--------------------------------------------------');

disp("\* Starting Intial Setup");

fprintf(u,'hz');

disp("\* Lights On");

fprintf(u,'dc');

disp("\* Drive Starting Value");

pause(5);

fprintf(u,'u8');

disp("\* Upper Arm Raised");

pause(10);

disp("\* Lower Arm Raised");

fprintf(u,'l5');

pause(10);

fprintf(u,'pf');

disp("\* Pan Centered");

pause(5);

fprintf(u,'tb');

disp("\* Tilt Level");

pause(5);

disp("\* Wheels Centered");

fprintf(u,'sf');

pause(5);

disp("\* Lights Off");

fprintf(u,'h0');

fclose(u);

disp("\* Intial State Set up Complete");

disp('--------------------------------------------------');

disp("\* Starting Classifer and training images")

%% Load Images & train Classifier

imds = imageDatastore('./data',...

'IncludeSubfolders',true,'LabelSource','foldernames');

%Next four lines are to test the faceRecognizer given the current database.

%[trainingSet,testSet] = splitEachLabel(imds,0.8,'randomize');

%bag = bagOfFeatures(trainingSet);

%faceRecognizer = trainImageCategoryClassifier(trainingSet,bag);

%evaluate(faceRecognizer,testSet);

%Identify key features of each subset of images

bag = bagOfFeatures(imds);

%Train the classifier using the database.

faceRecognizer = trainImageCategoryClassifier(imds,bag);

fopen(u);

%% Main loop

while(1)

%% Detection: Capture a frame from the webcam and detect the face.

frame = snapshot(cam);

bbox = step(faceDetector, frame);

while(isempty(bbox))

frame = snapshot(cam);

bbox = step(faceDetector, frame);

end

% This crops the frame for just the face to be used for recognition.

face = imcrop(frame,bbox(1,:));

%figure; imshow(face); title('Detected face');

%% Recognition: Attempt recogniton. If not recognized, skip tracking.

prediction = predict(faceRecognizer,face);

%% Tracking: Setup for face tracking

if(prediction == 4)

track = false;

else

track = true;

% Track the face and feed data to labview.

% If the face is lost, loop to detection

bboxPoints = bbox2points(bbox(1, :));

points = detectMinEigenFeatures(rgb2gray(frame), 'ROI', bbox(1,:));

% Display the detected points.

%figure, imshow(frame), hold on, title('Detected features');

%plot(points);

pointTracker = vision.PointTracker('MaxBidirectionalError', 2);

% Initialize the tracker with the initial point locations and the initial

% video frame.

points = points.Location;

initialize(pointTracker, points, frame);

% Initialize a Video Player to Display the Results

% Create a video player object for displaying video frames.

videoPlayer = vision.VideoPlayer('Position',...

[100 100 [size(frame, 2), size(frame, 1)]+30]);

oldPoints = points;

end

while(track)

% get the next frame

frame = snapshot(cam);

% Track the points. Note that some points may be lost.

[points, isFound] = step(pointTracker, frame);

visiblePoints = points(isFound, :);

oldInliers = oldPoints(isFound, :);

if size(visiblePoints, 1) >= 2 % need at least 2 points

% Estimate the geometric transformation between the old points

% and the new points and eliminate outliers

[xform, oldInliers, visiblePoints] = estimateGeometricTransform(...

oldInliers, visiblePoints, 'similarity', 'MaxDistance', 4);

% Apply the transformation to the bounding box points

bboxPoints = transformPointsForward(xform, bboxPoints);

% Insert a bounding box around the object being tracked

bboxPolygon = reshape(bboxPoints', 1, []);

%disp(['X1= ',bboxPoints(1,1),'Y1= ', bboxPoints(1,2)]);

disp('--------------------------------------------------');

fprintf('x1 = %f : y1 = %f\n', bboxPoints(1,1), bboxPoints(1,2));

fprintf('x2 = %f : y2 = %f\n', bboxPoints(2,1), bboxPoints(2,2));

fprintf('x3 = %f : y3 = %f\n', bboxPoints(3,1), bboxPoints(3,2));

fprintf('x4 = %f : y4 = %f\n', bboxPoints(4,1), bboxPoints(4,2));

disp('--------------------------------------------------');

%% Start of calculations

a = bboxPoints(1,1) - bboxPoints(2,1);

b = bboxPoints(2,2) - bboxPoints(4,2);

diagnonalDistance = sqrt(a.^2+b.^2);

fprintf('Digonal Size: %f\n',diagnonalDistance);

[ geom, iner, cpmo ] = polygeom(bboxPoints(:,1),bboxPoints(:,2));

x\_center = geom(2);

y\_center = geom(3);

fprintf("Center At: %f,%f\n", x\_center, y\_center);

if diagnonalDistance < 100

fprintf('\* Moving Closer\n');

fprintf(u,'d5');

fprintf(u,'db');

elseif diagnonalDistance > 170

fprintf('\* Moving Away\n');

fprintf(u,'d3');

fprintf(u,'db');

else

fprintf('\* I am Good here');

fprintf(u,'dc');

end

if x\_center < 210

fprintf('\* Moving Left\n');

fprintf(u,'sc');

fprintf(u,'d5');

elseif x\_center > 420

fprintf('\* Moving Right\n');

fprintf(u,'s3');

fprintf(u,'d5');

else

fprintf("Center\n");

fprintf(u,'sf');

fprintf(u,'d5');

end

frame = insertShape(frame, 'Polygon', bboxPolygon, ...

'LineWidth', 2);

% Display tracked points

frame = insertMarker(frame, visiblePoints, '+', ...

'Color', 'white');

% Reset the points

oldPoints = visiblePoints;

setPoints(pointTracker, oldPoints);

% Display the annotated video frame using the video player object

step(videoPlayer, frame);

else

track = false;

% Clean up

release(videoPlayer);

release(pointTracker);

end

end

end