BU

Senior Design
ENG EC 463

Memo

UPR

To: Professor Pisano

From: UPR Level Security: Juan Peralta, Eugenia Almandoz, Namir Fawaz, Brock Guzman, Brandon Webster

Team: 24

Date: 2/16/2018

Subject: Second Deliverable Test Report

1.0 Test Objective and Significance

1.1 Standing Person Detection/Tracking

For this section, the goal was to be able to detect and track a person. This task is for the security section of our project. One of our customer's requirements was to be able to detect an unauthorized person in a substation. The idea is that if a worker goes into the substation, there is no need for the robot to be performing its tasks, so they can turn it off through the web application when they get there. This means that if while on route it detects a person, we can assume they are unauthorized.

1.2 WebApp and AWS Hosting

The objective of the WebApp deliverable was to have an almost fully functioning WebApp running in an AWS environment so that it can be accessed from anywhere. This is a crucial component to our project, our client specifically mentioned that they want an application which will allow them to monitor the robot and receive alerts about any dangerous activity. Our goal was to have this section of the project completed early because it did not rely on any parts being ordered, so this deliverable was an important milestone to having this section completed, which would mean one large pillar of our was also completed.

1.3 Navigation

The objective of the navigation deliverable was to show how our team will be able to utilize a GNSS system to create a shortest path vector for the robot to travel to a goal location. We have started the autonomous navigation with a simple GNSS system that reads the location of the robot, the location of the goal area, and the orientation of the robot to direct the robot in the direction of the shortest path with the distance it needs to travel. This will be important in the future when we incorporate RTK error correction from online databases, making the navigation far more precise within an electrical

subsystem. This, along with the object avoidance module we're going to implement and the gauge detection will provide autonomous navigation for the robot.

2.0 Measurements and Data

2.1 Standing Person Detection/Tracking

During the testing we were able to show full functionality of the algorithm on a webcam of one of our laptops. We were in the hall and we have someone stand about 4 meters away from the webcam in order to be fully in the frame. The person then moved around left, right, closer, and farther from webcam. The algorithm was able to successfully detect the person and track their movements throughout. The script also boxed around the person, and the box moved with the person's movement. It also shows the amount of points it extracted as matching features of a standing person. The more points it displays, the more confident the algorithm is that it is a person.

2.2 WebApp and AWS Hosting

We were able to measure that the WebApp had a proper database by registering new accounts, and then using these to log in to the page. Additionally, we were able to add "dummy alerts" to the page and display them on both the homepage and the full records page. Both of these worked flawlessly on the AWS hosted site and in a localhost, demonstrating that the database was properly set up.

Additionally, we were able to demonstrate that logins were properly set up by showing that pages were restricted unless a user was logged in. Anytime a page such as the records or the remote control pages were requested, the user was immediately redirected to the login page. Additionally, the WebApp would store the page that was requested as the next page, so that the user could login and be directed there immediately.

Finally, we demonstrated that the "download video" option was working on a localhost, though it was not been debugged for the AWS hosted site. We demonstrated that by pressing the "Download Video" link, the user would immediately be redirected to a page of the video playing. We still need to incorporate this into the AWS site, but are struggling to add this component because of the different databases required to store the large video files. We have an idea of how we will solve this problem, and will finish it by the end of the weekend.

2.3 Navigation

The measurements that were taken for navigation included multiple reads from different satellites that included the latitude, longitude, number of satellites sending information, altitude above sea level, sea level average, the dilution of the different directions, and the velocity. The measurements that we used were the longitude, latitude, and direction of each from GPGGA satellites, and used them as variables for measuring direction and distance. Since we are going to install this robot in a relatively small area, we won't need to take into account the curvature of the earth when calculating distances and angles. Thus, we calculated the "flat earth" approximations of

distance between an inputed goal GPS location and the current reading from the GNSS receiver where $\Delta X = R\Delta\lambda cos\phi$ and $\Delta Y = R\Delta\phi$ where λ, ϕ are longitude and latitude respectively. Distance is a normal distance calculation and angle of shortest path is $\theta = tan^{-1}(\frac{\Delta Y}{\Delta X})$. We then calculated the angle needed to turn from the current orientation by either adding or subtracting the shortest path angle to or from the current orientation.

3.0 Equipment and Setup

3.1 Standing Person Detection/Tracking

This test was done on my laptop, and I ran the script on matlab which used the computer's webcam. We were in the hall and the person stood a few meters away to be in the frame. The matlab script checks for a camera and creates the object for it, then in feeds frame by frame to the detection and tracking algorithm. The detection algorithm uses a built in computer vision toolbox function called vision.PeopleDetector. This function extracts HOG features from a picture and matches them with their model. If there is enough matching points, the script will then locate the matching points and display a box around these points, which is the detected person. To track these points the script uses another matlab function called vision.PointTracker. Doing this the script is able to both detect and track a standing person inside the frame.

3.2 WebApp and AWS Hosting

The setup for testing our webapp was very similar to its production configuration. The main difference was that we used dummy values for our database to simulate an actual production database environment. Our alert values were procedurally generated, and we used a stock video for testing our video storage. Our setup used AWS Elastic Beanstalk for server hosting, AWS RDS for our database storage, and Amazon S3 for our file and video storage. We erased all user models in our database before the test, so that we could create a new one as part of the test. We had one stock video uploaded to our S3 server before the test, and a video model created to access that.

3.3 Navigation

The hardware required for the navigation included a SkyTraq GNSS RTK receiver, an antenna, and a Linux computer. The SkyTraq chip was connected in serial to the Linux computer via USB, and the antenna was connected to the chip with a simple converter. For this setup properly work, the antenna needs to have a clear view of the sky and should be outside, however we were able to test it inside to read 0' N, 0'E inside just to test the program. Once the program is executed, the user is asked to input the lat/long and direction of the goal object (ie. N/S, E/W), and as the SkyTraq reads in positions, the program calculates the distances and angles.

4.0 Conclusion

4.1 Standing Person Detection/Tracking

For this deliverable, we can conclude that the algorithm works well enough for our needs. Since the substation is large and spread apart, there shouldn't be a lot of objects in the way that could disturb the algorithm. The goal is to be able to get it running on the raspberry pi, which is going to be the processor on our robot, that also communicates with the web application. We have the code converted to C++ on the raspberry pi but are still working on getting it to compile.

Our goal is to be able to have it communicate with the web application and send an alert that a person was detected. So once it is working on the raspberry pi, we want to be able to stream the player straight to the website and also send an alert once a person is detected and be able to timestamp either where in the video we can find this occurrence, or be able to save a clip of it. This will then be stored in the database. Since the idea is to be able to track the person for security purposes, we will have the camera attached to a servo, this means we are going to have a control system that turns the camera so that the person stays tracked in the middle of the frame, in order to not lose track of the person. We have already been working with a servo, and are confident we can have it communicate with the algorithm in order to receive movement commands.

4.2 WebApp and AWS Hosting

Our objective was to finish the WebApp, besides the Remote Controlling portion. We are pleased that we were able to complete so much, and have a solid shell that is tied into the database. With this foundation, we can begin to work on the larger tasks left at hand. Our next objective will be to create a secure stream of video from the Raspberry Pi. With this in place, we will then begin creating the Remote Control aspect of the robot.

The remote control will be the first time we are able to get our robot moving. This will be integral to preparing the robot for autonomous navigation. We are confident in how we will accomplish this goal, using the WebApp to send direct signals to the Arduino which will be controlling the robot's motor. The robot is built in with an Arduino package to control it, meaning our main task will be finding a way to send signals to the Arduino. If we cannot send signals directly to the Arduino, we will send them to the Raspberry Pi which will in turn send them to the Arduino.

Having essentially completed the WebApp and successfully hosted it on AWS, we will be able to work on the next major tasks of the project. This is a big milestone, and we are confident we will be able to complete the rest of the project in a timely manner.

4.3 Navigation

While the deliverable would have worked better if we had clear view of the sky, the program at least demonstrated the success of its functionality. Professor Hirsch expressed his worries that without RTK the measurements of GNSS would be too inaccurate within a subsystem. After speaking with Sebastijan we learned that we will be able to use public records to implement RTK, found online at some states' department of transportation website, and this information can be implemented with our particular

GNSS receiver. The next steps following this deliverable include the actual navigation of the robot with online remote control, transmitting commands from our navigation program to the robot as directions to move, implementing the object avoidance technology, implementing RTK corrections to our GNSS system and finally incorporating the gauge detection system to navigate to the gauge once the robot is near.

For the remote control aspect of our robot we will be controlling it via the keyboard keys and we will track which direction it wants to go based on the key that has been pressed. This will then send a string to our server which will be communicating with the raspberry pi. The raspberry pi will be serially connected with the arduino and will send the string over to the arduino in order to control the direction of the robot accordingly. Each key will correlate to a certain wheel direction making it travel as desired. Depending on our time constraints the and whether we have time at the end the keyboard functionality may be changed to using a joystick to help better simulate the robots movement.

For the object avoidance technology we plan to use a dynamixel ax-12a in order to mount a ultrasonic sensor for object detection. The servo will be moving 180 degrees and will take 12 measurements throughout this 180 degree turn. The plan is to have the robot drive straight until it reaches an object in its path. It will then take these 12 measurements to detect the closest free space that will allow it to continue moving straight. We are essentially looking for the smallest angle from 90 degrees(which is straight ahead) to continue travelling and in turn get the robot to its destination in the smallest amount of time. Once it finds that direction it will continue driving in that direction but turn back to the position in which it was travelling before in order to get back on track to its destination.