PatrolBot

Progress Demo and Wrap-Up Document

University of Nevada, Reno
Department of Computer Science and Engineering

Team 08
Jesus Aguilera, Brandon Banuelos, Connor Callister, Max Orloff, Michael Stepzinski

Instructors: Devrin Lee, Dr. David Feil-Seifer, Vinh Le

Advisors: Dr. Hung La, Dr. Alireza Tavakkoli, Officer Matthew Stewart

March 14th, 2022

Table of Contents

Implemented Use Cases

1. CheckForOverlap - Brandon
2. DetectActivity - Brandon/Max
3. SendAlert - Brandon/Jesus
4. LoginUser - Jesus
5. LogoutUser - Jesus
6. SaveLogFile - Jesus
7. SaveAlertFile - Jesus
8. DisplayControls - Michael
9. DisplaySettings - Jesus/Connor
10. ScanEnvironment - Brandon/Max
11. MoveRobot - Michael

13. ViewStream - Brandon/Jesus

15. DetectObjects - Brandon/Max

17. MoveLocation - Michael

Implemented Requirements

ID	Level	Description
FR00	1	When malicious objects are detected, a medium threat alert will be made on the interface - Brandon/Max/Jesus
FR01	1	When malicious objects are near bikes, a high level threat alert will be made - Brandon/Max/Jesus
FR02	1	When suspicious activity is detected, a low threat alert will be made - Brandon/Max/Jesus
FR03	1	PatrolBot shall identify objects of interest through live video - Brandon/Max
FR08	**1**	PatrolBot shall allow the user to customize the dashboard to their individual liking. - Jesus
FR09	**1**	PatrolBot will stream video feed from the robot to the web application - Brandon/Jesus
FR12	**1**	PatrolBot shall operate using a robot and a camera setup - Brandon/Michael
FR13	**1**	PatrolBot shall have the option to be manually controlled through the web application - Michael/Connor
FR14	**1**	Multiple users shall be able to view the PatrolBot dashboard at once - Jesus
FR16	**2**	PatrolBot shall maintain a text log of objects detected and time stamp - Jesus
FR18	**2**	PatrolBot will utilize a camera that can pan around its environment. - Brandon/Michael

To be Implemented Use Cases

12. GetLocation - Connor/Michael

16. DisplayBoxes - Max

To be Implemented Requirements

FR04	1	PatrolBot shall allow users to activate and deactivate the object detection model - Max
FR05	1	PatroBot shall allow user to decide what objects are to be detected from set list of available objects - Max
FR10	1	PatrolBot will show estimated robot location - Connor/Michael
FR11	1	PatrolBot will use odometry readings to help with location tracking - Connor/Michael

General Items to Be Implemented

Increase camera and model performance - Brandon/Jesus

Finalize UI Design - Connor

Change log behavior - Jesus

Change robot controls - Michael

Current Project Status

Currently, we have each individual portion of the project connected through our website, now we're in the refining stage. The front end is usable, the back end accesses the correct functions, the AWS portion links the back end to all aspects of the project, robot control through AWS-IOT and ROS functions properly, and the models function properly when individually tested on a stronger computer. The only part of the system that doesn't work well is the camera and our deployment of the models on the server. We need this functionality to happen in near-real-time, and is our top priority. In general, everything needs to be refined and expanded upon from this point to make an easily usable application. Compared to our December 2021 CS 425 demo we had a small DIY robot with motors connected through a raspberry pi, accessed the interface on the raspberry pi using VNC (a remote access software), sent commands to the device keyboard controls over VNC, mounted a camera to the front of it all, and ran the models on the pi locally. This implementation barely worked, but it was our proof of concept. Now, we have a full-sized robot, a website, an improved object detection model, a usable threat prediction model, and AWS to connect everything.

Contributions of Team Members

Connor

Tasks: Frontend UI Website Design, Raspberry Pi local server communication, Raspberry Pi web server communication(helping/testing), Raspberry Pi hardware components integration.

Time: 40 hours

Brandon

Tasks: Action detection model implementation, Action detection algorithm implementation, PyQt5 UI design for local application, Raspberry Pi configuration and streaming to AWS Kinesis, retrieval of stream using AWS Python library, Application of YoloV5 and action detection model/algorithm on stream in backend through Django, triggers for logging activities in Django, experimenting with CUDA and models

Time: 60 hours

Max

Tasks: Object Detection Dataset expansion, Object Detection Model training, Improvement of Object Detection Model accuracy, Action Detection Dataset creation, Action Detection training.

Time: 40 hours

Jesus

Tasks: Frontend UI Website Design, Backend web server integration with Django, Web app publication with AWS Elastic Beanstalk, Domain name/hosted name server routing, Account database integration within the website, Video Feed integration within the web server (helping), Objects detected logging, Security threat logging, Integrated website settings communication for the video feed and model.

Time: 60 hours

Michael

Tasks: Current Project Status, updating raspberry pi firmware, fixing compatibility issues to get Ubuntu 18.04 64-bit working on a Raspberry Pi 4, understanding and using ROS Melodic to design robot controls on the raspberry pi, setting up AWS-IOT server to subscribe to and publish messages using MQTT, creating website backend and frontend code to publish to AWS-IOT server on button presses, creating raspberry pi MQTT communication with AWS-IOT to get commands, placing and affixing devices to robot, maintaining robot

Time: 120 hours

Repositories

https://github.com/banuelos-brandon/PatrolBot

https://github.com/jeaguil/patrolbot-webapp2