Vikingbot

Travis Hermant
Abdullah Barghouti
Abdirahman Alasow
Jagir Laxmichand Charla

ECE 478-578 Intelligent Robotics I

Summary **3**

Software Design **4**

Intro 4

Overview 4

Challenges 7

Possible improvements 7

ROS Design **8**

Intro 8

Overview Diagrams 8

Coordinator 10

Motor Motion 10

Servo Motion 10

Recorder 10

Dialogflow 10

Polly 11

Talker 11

Challenges 11

Possible improvements 11

Hardware Design **12**

Intro 12

Overview 12

Challenges 12

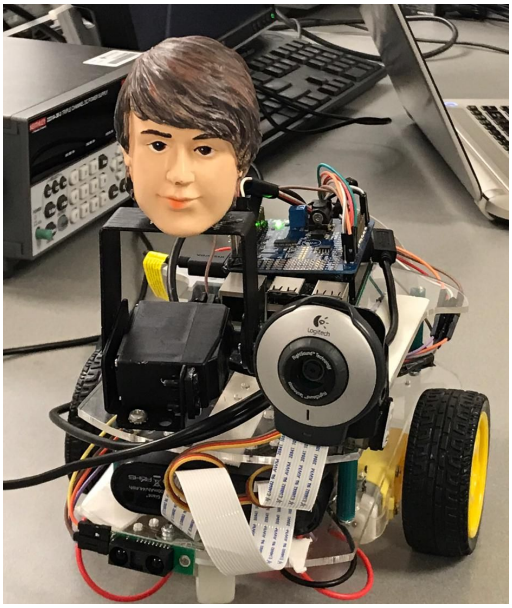
Possible improvements 13

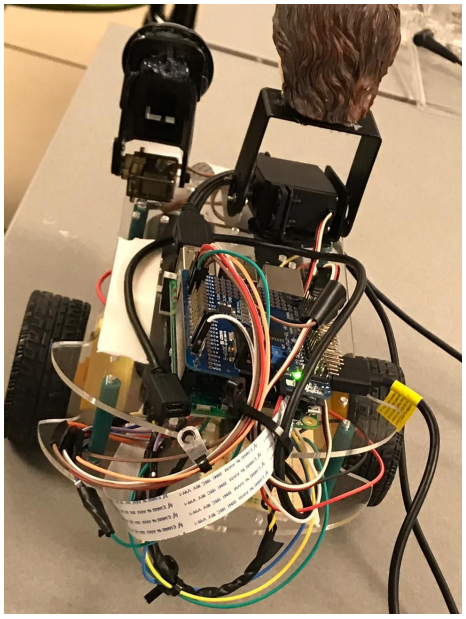
Conclusion **14**

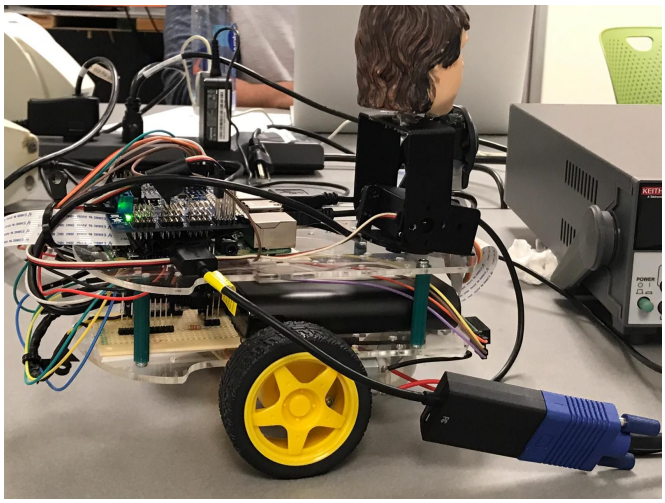
Resources **14**

Summary

The Vikingbot contains two DC motors for the wheels driven by an L293E H-Bridge making it capable of operating both wheels forward and backward in conjunction, or allows each to operate in separate directions to allow it to turn. Central processing is handled by a Raspberry Pi 3 B with an attached Adafruit Servo Hat for PWM control of the attached servo. Coding is done with Python; an attached Pi Camera is used in conjunction with OpenCV to handle applications such as color tracking with a ball, face tracking to steer the car, and smile recognition to make the servos respond to gestures. The Vikingbot utilizes ROS to facilitate using Dialogflow to interpret commands, Amazon Polly to communicate with users, and to drive the motors and servos.







Software Design

Intro

We have 4 different independent applications for the demo:

- Ball Tracking Bot
- Face Tracking Bot
- Remote Controlled Bot
- Smile Detector Bot

Which can be broadly divided into two parts like Vision and Motion.

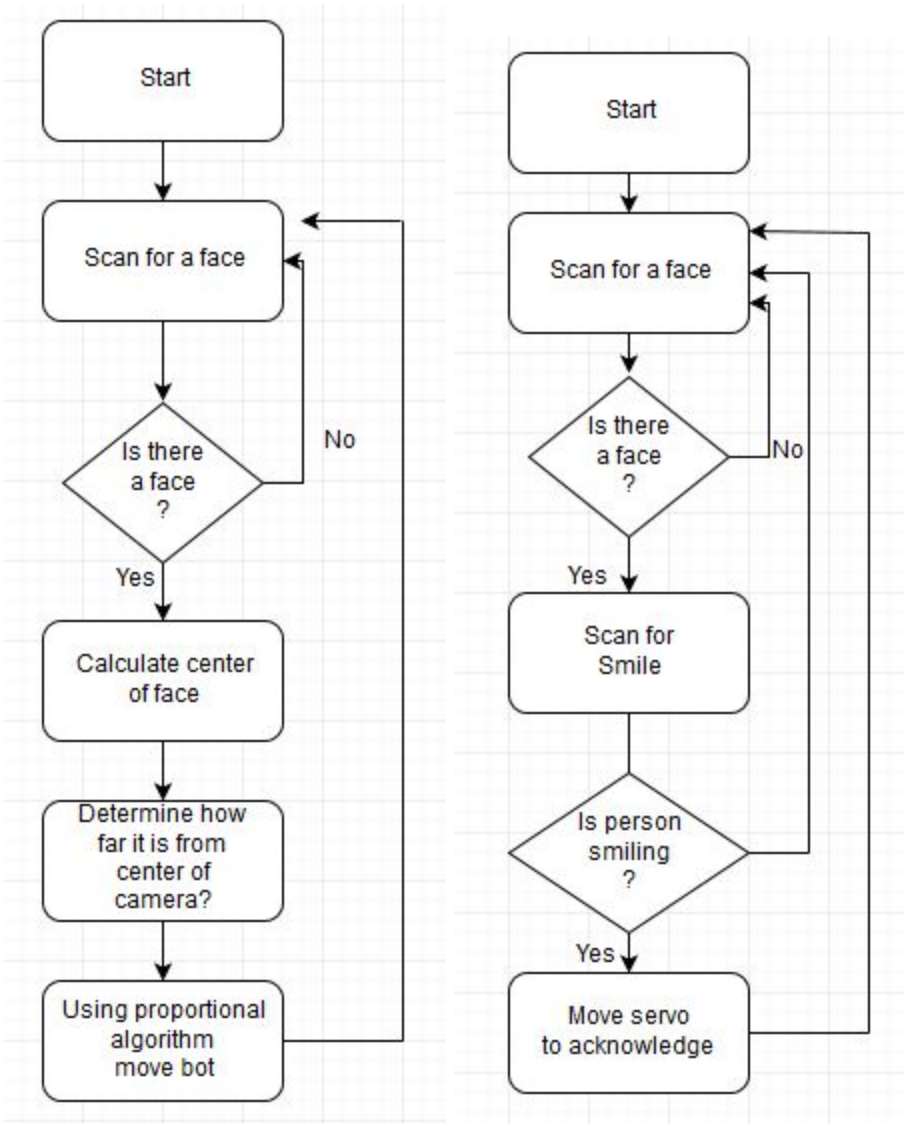
We have used OpenCV with python for the Vision and we have used RPI's GPIO module for the motion.

Overview

For Remote Controlled Bot we have made two applications talking over TCP socket. The one which actually executes the motion is TCP Server , whose task is listen to the command and execute it according to the request generated from TCP Client.

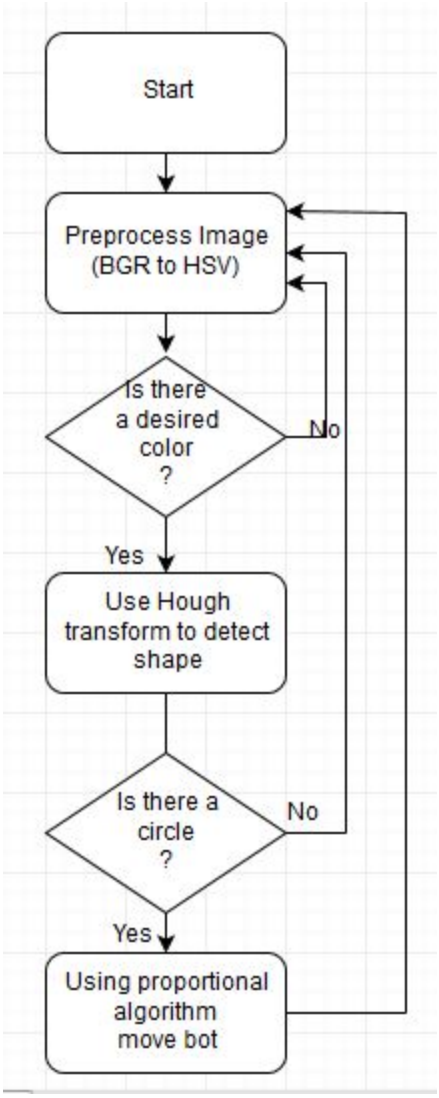
TCP Client takes data from a user in the form of letters 'w', 'a', 's', 'd' and 'b':
'w' for forward
'a' for left
's' for stop
'd' for right
And 'b' for backward

Server also monitors IR sensor to check for the obstacle in front. It stops the motion in case of detection of obstacle.



For smile detecting bot we have used Haar cascade classifier to detect a face in an image. After getting a face we are looking for smile in the region of interest using Haar cascade smile classifier. Depending on detection of smile we are moving the servo to make it look like a greeting gesture.

For face tracking bot we have used Haar cascade classifier to detect a face in an image. For tracking we are calculating the center of the face and determining how far it is from the center of camera. Depending on which we are rotating the bot till center of the face gets align to the center of the camera.



For ball tracking bot we have converted an image to HSV. HSV is good for color segmentation purpose. We are using hough transform to determine circles in an image. For tracking we are calculating the center of the circle and determining how far it is from the center of camera. Depending on which we are rotating the bot till center of the ball gets align to the center of the camera. After that we are comparing the area of the circle to determine how far the ball is from camera, and accordingly we are moving the bot forward if area is too small.

Challenges

Face detection and ball detection are prone to misbehave in case of poor lighting condition. Apart from that due to various morphological operations involved in pre processing frame rate reduces.

Possible improvements

Implementation of robust and fast methods for detection of face and ball, and integration of multiple Vision algorithms to make bot more interactive.

ROS Design

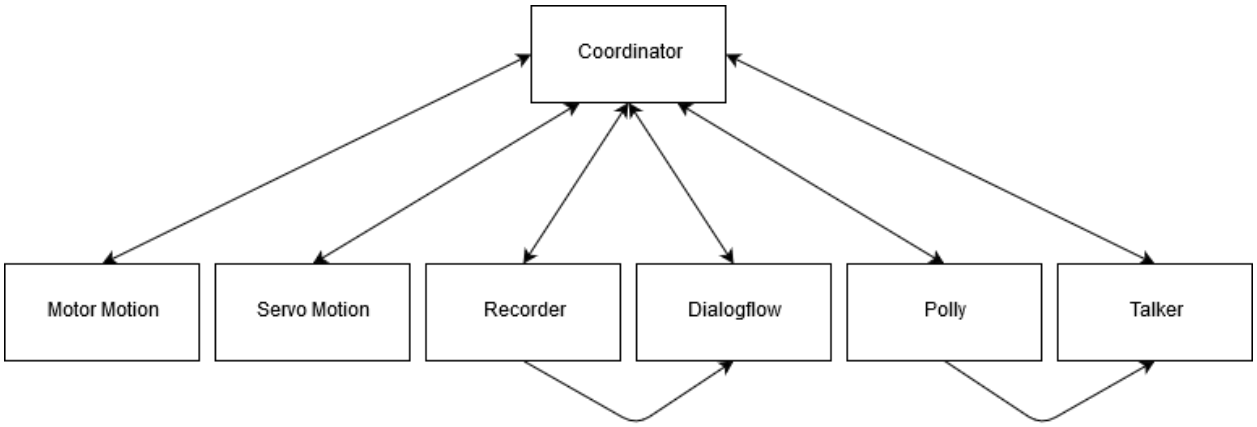
Intro

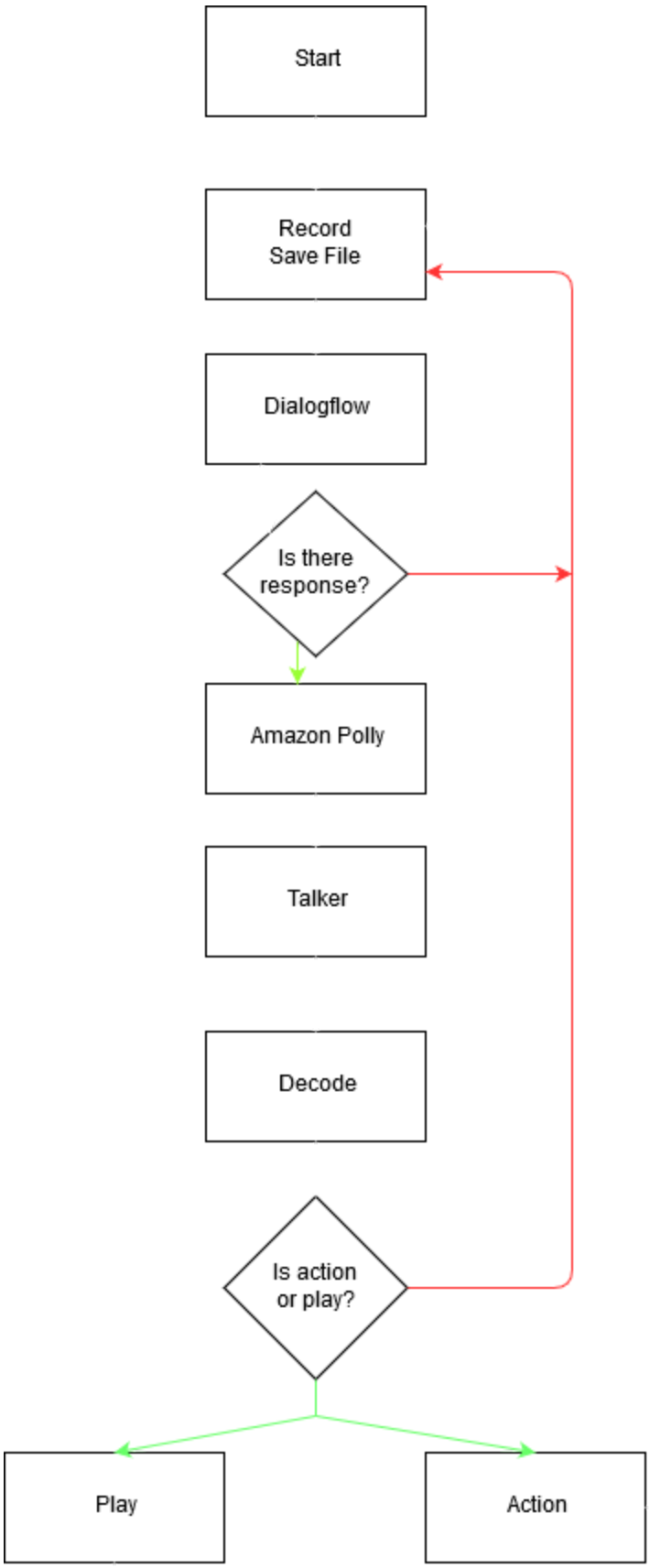
We have 7 different ROS nodes working in conjunction.

- Coordinator
- Motor Motion
- Servo Motion
- Recorder
- Dialogflow
- Polly
- Talker

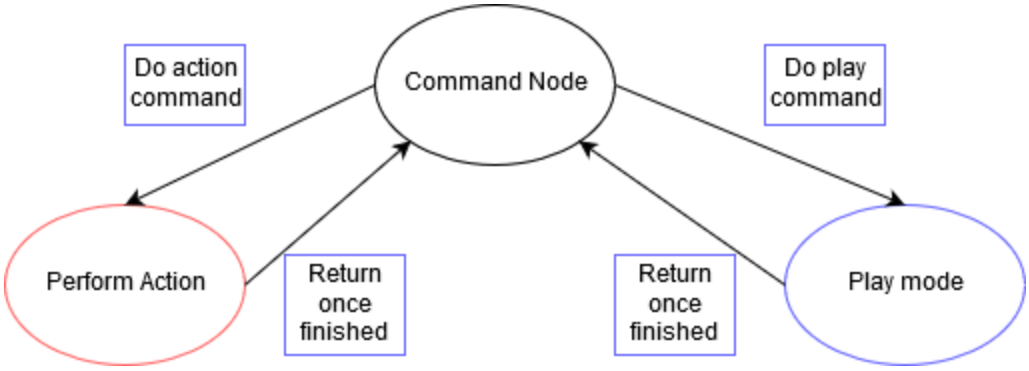
These nodes can be used in conjunction for speech to text, text to speech, and motion commands. Currently OpenCV and Color/Facial recognition are not implemented for use in ROS.

Overview Diagrams




Start
Record
Save File
Dialogflow
Is there
response?
Amazon Polly
Talker
Decode
Is action
or play?
Play
Action

Coordinator



The coordinator node is the “brain” of the robot, deciding what should be done and sending commands to all of the nodes.

Motor Motion

The motor motion node is a conversion of our previous driver so that they’re compatible with ROS. It is capable of moving forward, turning left and right, and going backwards.

Servo Motion

The servo motion node is also a conversion of our previous driver for controlling the servo that the bobble head sits on. Since there is only one servo, it is only capable of one degree of freedom and can move in a 180 degree arc. Currently its usage is limited to nodding as a form of expression. In the future it will be adapted to accompany additional servos for the arms.

Recorder

The recorder node is where the microphone will catch audio in 5 second loops and convert them into WAV files. The audio isn’t parsed here though, and will instead need to be sent out to the Dialogflow node

Dialogflow

The Dialogflow node will send the WAV files it receives from the Recorder node to the Google API servers, where it will be analyzed to return the intent so that the proper actions can be taken. Actions Dialogflow can detect are: entering ‘theater’ mode, turning ‘left’, turning ‘right’, going ‘forward’, and going ‘backward’.

Polly

The Amazon Polly node is used to convert text to speech. Lines of the play are sent in a string to the AWS servers, where the string is then converted to an MP3 files. Polly is configured to speak in a Male British accent, to coincide with our play script following Alan Turing.

Talker

The Talker node is used to drive the speakers with MP3s that are generated by Amazon Polly. The talker node also takes in the spoken speech and decodes it to identify what mode/ action the robot should enter.

Challenges

ROS is difficult to set up and having the capability of running nodes on different computers is nice, but also complicated. It is difficult to use Amazon Polly and Dialogflow because they both require internet and establishing a connection to the internet in conjunction with multiple robots/computers is also a complicated task.

Possible improvements

Currently, the Vikingbot is running solo and does not have any communication set up between it and another robot. A method of communication between the raspberry pi, a parent computer, and another robot is vital for further development.

Pulse Width Modulation should be added to the motor for the wheels, so that we can vary the speed at which the robot moves around. PWM would greatly increase our ability to emote and give greater control over what we want it to do.

We still have our temporary design in use due to complications with the 3D printers, so we’re limited in what we can do to increase functionality through hardware. Eventually when we have the new design, we can add new components such as an extra battery to separate the power draw from the servos, which would allow us to add more servos.

Hardware Design

Intro

During our initial stages of designing and assembling the robot's hardware, we figured it would be useful to list the parts we will make use of in our robot design. From our group discussions, we concluded that we would incorporate the DC motors that came with the robot base, raspberry pi 3b, raspberry pi hat, servo motors, portable battery pack, and an IR infrared obstacle avoidance sensor. We also used a voltage divider to step down the voltage from the power supply from 5v to 3.3v so that it can be wired to the raspberry pi pins safely.

We used the raspberry pi hat as a safety precaution to ensure that the servo motors we attach to the main board don't draw too much power and damage our raspberry pi. The Hardware Attached on Top (HAT) also allows us to connect up to 16 servo motors and has a 5-volt input to ease the power draw on our main board.

Overview

To use the raspberry pi pins, we had to use a voltage divider with a 1k ohm and 2k ohm resistors so that the power input on the pins is dropped from 5v to the rated 3.3 volts. To connect the servo motors we used the hardware attached on the top chip to protect the main board from any access power draw. This chip allows for protection and adds the capability to add up to 16 servos and another power input to suffice the needed current and voltage to power all the added components connected to it.

Our current design allows for the servo motor with the bobblehead to move in a single direction (upwards and downwards), our design, however, allows for up to 2 motors (right and left movement as well) but because of the power limitations, we had to omit the second servo.

Challenges

Because of our limited power supply, we had issues connecting more than one servo to our robot. Connecting the servos, raspberry pi, DC motors, and the camera drew just enough power that adding another servo motor caused the raspberry pi to shut off.

Setting up google dialog flow was tedious at first but the video uploaded by Melih Erdogan made things more manageable.

Due to the limited bandwidth, the connection between the raspberry pi and host computer was lagging and causing lots of delays. To overcome this challenge, we were forced to use a wired connection

Possible improvements

The power supply that we used was not powerful enough to support all the components we had connected to it. A future improvement would be to find a better alternative to supply the needed power to all our components while still keeping the robot relatively light and mobile.

Conclusion

This project incorporated all the movement and functionality that we had in the first project. For this project we used ROS to publish and subscribe to nodes, Google Dialogflow to create text from speech, and Amazon Polly to create speech from text. By the end of the project, the VikingBot supported motion, computer vision, speech to text, and text to speech.

Resources

Github:
https://github.com/travishermant/vikingbot2-ECE478
Youtube:
https://www.youtube.com/channel/UClnZui7ccUFAsCJwCExTw2A