ECE2008 - ROBOTICS AND AUTOMATION

IRIS CONTROLLED BOT

J-Component Report
Winter Semester 2019-20

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

TEAM MEMBERS":

Vedant Tibrewal 17BEC0091
Arjun T 17BEC0191
Aman Pandey 17BEC0406
Godfrey Paul Joseph 17BEC0840

ACKNOWLEDGEMENT

The success and outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along with the completion of my project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

I would like to express my very great appreciation to Prof. Budhaditya Bhattacharyya for his valuable and constructive suggestions during the planning and development of this project work. His willingness to give his time so generously has been very much appreciated.

ABSTRACT

We suggest an economy efficient four-wheeled surveillance robot powered by a Raspberry Pi. Surveillance robots generally accommodate a video capturing peripheral, a GPS module, and GSM radios. Raspberry Pi accompanied by wonderful hardware satisfying the above requirements, (except the camera which is connected separately). This can be utilized and used as an advantage through API (Application Programming Interfaces), a provision inbuilt in the Raspberry Pi. Moreover, the price for building the robot with the help of a Raspberry Pi is covered back to a large extent. The robot may be controlled remotely from a computer using the Internet and a Raspberry Pi interface within the robot.

To capture and store the real-time visual input from the robot, the camera input is employed. The robot can be used via visual feedback from the same camera. Two motors facilitate a zero turning radius. The camera is connected to a stepper motor that makes it possible to capture the scene or object of interest. The captured video may be amplified to an intelligible video using image processing on the remote computer hence eliminating the requirement for further DSP hardware on the robot.

INTRODUCTION

Surveillance is the method of observing a state of affairs, a neighbourhood or an individual. Human surveillance is implemented by deploying personnel close to required areas to monitor for changes. However humans do have their limitations, and sending them to inaccessible places isn't possible. There are extra risks of losing personnel within the event of being caught by the enemy.

With the advancement in technology over the years, however, it's possible to monitor areas of importance, remotely by utilizing robots instead of humans. With the apparent advantage of not having to risk any personnel, terrestrial and aerial robots may detect details that don't seem to be obvious to humans.
Using high-resolution cameras and various sensors, it's easier to get information concerning the particular area under surveillance.

A standard theme is the use of a camera on the robot to receive live video feedback. A robot that performs image processing using the camera has been enforced. However, this technique is limited by the processing power of the Raspberry Pi, a drawback that we addressed by remotely carrying out image processing on a separate computer by taking the input from the camera. Our project is quite distinctive within the sense that, it is an inexpensive solution that provides the power to remotely control a robot while providing video feedback.

OBJECTIVE

Through our project, we propose an effective four-wheeled surveillance robot using raspberry pi, IRIS cameras and motors. Surveillance robots usually consist of GPS modules, various sensors, cameras for recording videos and Wifi modules. Raspberry pi comes with excellent hardware which satisfies all the above needs. All the necessary modules can be easily connected and controlled. Raspberry Pi acts as the brain of our robot, it receives commands and executes them efficiently. The robot can be controlled remotely from a PC by wifi. To capture and archive the real-time video from the robot, we are using IRIS camera input. The robot can be controlled based on feedback obtained. We are using two motors to obtain zero turning radius. The camera is mounted on top of our chassis which increases the range of view and makes it feasible to capture images more efficiently. Thus, we aim to build a fully-featured surveillance robot using an easily available Raspberry pi, which can be remotely controlled over the WIFI.

METHODOLOGY

We used two major concepts in this project. For the recognition of the iris movement and facial landmarks with the help of dlib, OpenCV and Python, and socket programming to communicate between the laptop (which sends the iris movement recognition data) and RaspberryPi (which takes that data of where the user is looking and decides the direction of the movement).

Our laptop acts as a client which captures video via web-cam and then using the aforementioned libraries it detects the movement of the iris and sends that data (left, right, blinking) to RaspberryPi (server) which with the help of a motor and chassis moves in the direction provided by the client.

SOCKET PROGRAMMING

Sockets (aka socket programming) is a program that enables two sockets to send and receive data, bi-directionally, at any given moment. Natively, Python provides a socket class so developers can easily implement socket objects in their source code.

```
from socket import *

host = "192.168.43.34"
port = 13000
addr = (host, port)
UDPSock = socket(AF_INET, SOCK_DGRAM)
cnt = 0
direction =b'center'
```

Creating a socket object at client
(*Entire code link added at end*)

One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.

HOW WE USED IT?

We used socket programming to communicate between RaspberryPi nad our laptop using UDP protocol. The Raspberry pi listens to a port of the IP address and the laptop reaches out to it.

The bind method at the server (Raspberry pi) binds it to the specific IP address and port so that it can listen to incoming data at this IP and port.

```
host = "192.168.43.34"
port = 13000
buf = 1024
addr = (host, port)
UDPSock = socket(AF_INET, SOCK_DGRAM)
UDPSock.bind(addr)
```

The data is then received by the server from the given port and address. Depending on the data received by the Rpi it calls the corresponding method which makes the motor run the chassis in the desired direction.

```
(data, addr) = UDPSock.recvfrom(buf)
if data =="center":
        print("forward")
        forward()
elif data == "right":
        print("right")
        right()
elif data == "blink":
        print("stop")
        stop()
elif data == "left":
        print("left")
        left()
```

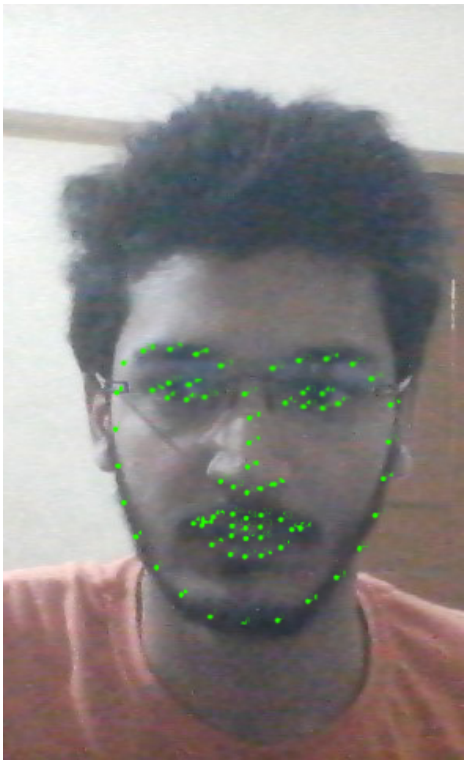
DLIBS

Dlib is open-source cross-platform software (as well as a library). Although dlib has a large variety of tools which has found its use in many fields such as machine learning, image processing, data mining etc. here we are using it along with image processing (library – OpenCV) to recognise the face of the user following by the recognition eye then iris and finally pupil of the eye.

HOW WE USED IT?

Dlib has a data file which consists of the facial landmarks as shown in the figure. Almost all the key feature of a face are marked in the data file. It helps us to map/manipulate/select the particular point of the face. For example- if the programmer has interest to work with only right eye of the face image he can direct select the points range from [42-47] and crop out the region of interest (in this case the right eye).

In our project, our final RoI was iris. So after the recognition of the face and iris with help of dlib and opencv. We are calculating the ratio of horizontal distance to vertical distance for blinking ratio. We divide eye into 3 parts vertically to calculate the gaze ratio.
For detecting whether the client is looking right/left/center, after binary thresholding we are calculating the white part of the eye(sclera) is in which region of the gaze ratio

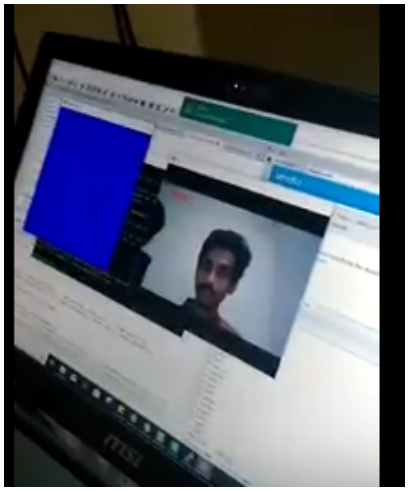


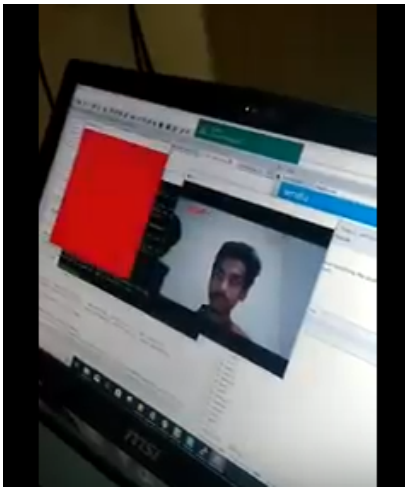
ALGORITHM FOR BLINKING
If (horizontal distance/vertical distance)>4:
Client has blinked;

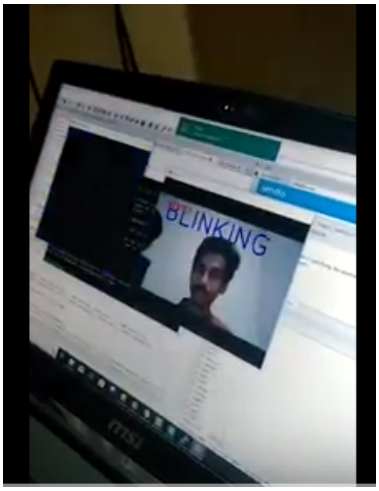
ALGORITHM FOR DIRECTION
If the white is more in left:
Client is looking right;
Else If the white is more in right:
Client is looking left;

NOTE: ALL THE THRESHOLD ARE CALCULATED ACCORDING TO USER IN THIS CASE.

WORKING







As seen in the image above, when the user looks right, the screen is red , blue the user look left and black if blinking. Upon this the client (laptop in this case) sends "right", "left", "center" , "blinking" (stop command for the Rpi) via the port and IP and the Rpi responds accordingly.

Link of the video :
https://drive.google.com/drive/u/1/folders/1gX3g2fCqowRSEEUMmMW7t7IrrhUSf42l

RESOURCES

https://github.com/Vedant-Tibrewal/Iris-controlled-bot/tree/master

Link for GitHub repository with the codes used in RaspberryPi and laptop. Files included:

- final_code_client : is code which is to be uploaded on the system, make sure your system has python (software), opencv, dlib and socket libraries installed

- final_code_server : is the code to be uploaded on the raspberry pi, it contains the command and command for making the robot mobile. Make sure raspberry pi has constant internet connection, socket library installed

- shape_predictor_68_face_landmarks.dat.bz2 : is the zipped file of the data file which contains the facial landmarks of the human. It is necessary along with client python file for your system to recognise your face and mark them with landmark which is used to extract the RoI.

https://drive.google.com/drive/u/1/folders/1gX3g2fCqowRSEEUMmMW7t7IrrhUSf42l

Video link of the movement of the bot which is controlled by the iris.

CONCLUSION

What we did?

1. Iris movement recognition using dlibs and OpenCv.
2. Communicating between our laptop and Rpi using socket programming
3. Capture and stream real time video from the robot
4. Movement of the bot on the basis of the iris movement.
5. Research on ROS simulation using softwares like simscape and Gazebo ROS

What we could have done?

1. Added an inbuilt obstacle avoidance system,
2. Actual simulation of the robotic movement. (*This we weren't able to add due to lack of time and software support / availability with the system*

REFERENCES

1) Mayank Vatsa, Richa Singh, P. Gupta "Comparison on Iris Recognition algorithms" presented at ICISIP, July 2004

2) IRIS recognition - Eduard Bakstein

3) Jatin Sharma, Anbarasu M, Chandan Chakraborty and Shanmugasundaram M, "Iris movement based wheelchair control using raspberry pi" published in IOP Conf. Series: Material Science and Engineering, pp 263, 2017

4) Priyanka Khelbude, Sushama Shelke "Real Time Iris Controlled Robot" In Proc. Online International Conference on Green Engineering and Technologies, 2016

5) Suzanna Kunik, Adam Bykowski, "Raspberry Pi based complete embedded system for Iris recognition", presented at Signal Processing Algorithms, Architectures, Arrangements, and Applications, Ponzan, Poland, September 2017

6) C.H. Mohammed Koya "Eye monitored wheelchair system using raspberry pi" published in International Journal of Innovative Research in Science, Engineering and Technology – Volume 6, special issue, March 2017

7) Shyam Narayan Patel, V. Prakash, "Autonomous Camera based Eye Controlled Wheelchair System using Raspberry pi", presented at IEEE Sponsored 2nd International Conference on Innovations in Information Embedded and Communication Systems , 2015

8) Poonam S. Gajwani and Sharda A Chhabria "Eye Motion Tracking for wheelchair control", published in International Journal of Information Technology and Management, Volume 2, issue 2, pp. 185-187, December 2010

9) Preethika Britto, Indumathi J, Sudesh Sivasaru, Lazar Mathew "Automation of wheelchair using ultrasonic and body Kinematics", presented at National Conference on Computational Instrumentation, Chandigarh, India, March 2010

10)Saranghi P. Parikh, Valdir Grassi Jr., R. Vijay Kumar, Jun Okamoto Jr. "Integrating Human Inputs with Autonomous Behaviors on an Intelligent Wheelchair Platform", published in University of Pennsylvania Scholarly Commons, Departmental Papers, April 2007