Project Peekaboo

ITSP Summer 2019, ITC, IIT Bombay

Team Details 2

Discussion on Selected Problem 3

- Identification of problem 3
- Explain the selected problem statement 3
- Why is it a good problem statement? 3

Project details 4

Timeline 7

Budget 8

References and acknowledgments: 10

Team Details

TEAM ID: 76
TEAM NAME: Tech Chef

Name	Roll Number	Year & Program	Email ID	Contact No.
Harshit Shrivastava	18D070011	1 Dual Degree in Electrical Engineering	main.harshit@gmail.com	9971874599
Shubham Kar	180070058	1 B.Tech Electrical Engineering	karshubham257@gmail.com	6204734872
Shaan Ul Haque	180070053	1 B.Tech Electrical Engineering	shaanhaque2016@gmail.com	9386688100
Sumanyu Ghoshal	180070060	1 B.Tech Computer Science and Engineering	sumanyu.ghoshal@gmail.com	9820003259

Enlist subdivisions planned and essential skill-sets of the team members involved.

Name	Subdivision	Key Skill-set
Harshit Shrivastava	Software & Electronics	Face Recognition & Electrical Systems and their programming with Python
Shubham Kar	Electronics	Electrical systems and their programming with Python
Shaan Ul Haque	Mechanical	Construction of the chassis of the Bot.

Sumanyu Ghoshal	Software & Electronics	Face Recognition & Electrical Systems and their programming with Python

Discussion on Selected Problem :

Identification of problem

In most of the residential or industrial complexes, security is still handled manually with the help of security guards and other human personnel. Security guards, therefore, cannot be a fool-proof system: Human error creeping in, for complexes of a large scale is the reality. We cannot compromise on security. Automation is the only way forward to make a fool-proof and robust form of a security system.

Explain the selected problem statement

We aim to build an independent robot, which acts in place of a security guard at the entrance of a residential or industrial complex. The robot can do the following:

- Recognize whether the person entering the complex is already registered in the system or not.
- Send a video to a security manager if the person who is trying to enter a compound is not registered in the system so that the security manager can decide whether the person should be allowed into the complex or not.
- Take the guest to the place the security manager has allowed to enter.

Why is it a good problem statement?

Bottom line: We cannot compromise on security. A swarm of such robots, along with some defense mechanisms in robots will make a fool-proof security system. We are already seeing robots taking over so many parts of our lives; it is startling not to see them working in the field of security the way it should be.
Other implementations:

- https://www.socialtables.com/blog/hospitality-technology/hotel-brands-robot/
- https://www.theguardian.com/world/2015/jul/16/japans-robot-hotel-a-dinosaur-at-reception-a-machine-for-room-service

Project details

In the first two weeks, we have designed the final body as we would like to have, along with setting up a circuit and make the motors work as per the instructions from Raspberry Pi.
We have used the following equipment:

1. **Raspberry pi 3B+**: This microcontroller has the necessary hardware for Wifi connectivity for internet connectivity for maintaining the connection with the security, and Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz, which would ensure that the lag is minimal during video calls and face recognition.
2. 3 SG-90 servo motors: These motors are the lightest we found in the market. Pan and tilting of camera use two while rotating the ultrasonic sensor uses one motor.
3. 4 DC motors 300 RPM: To be used for movement of the robot. The torque of the motor=2kgcm. The purpose of using this motor is to be able to lift the load and at the same time move at a speed similar to any other person
4. CMOS Camera: For face recognition.

Along with this, we are using Haar Cascade for face detection. Although Haar Cascade is not extremely accurate, the fact that its execution is fast made us choose haar cascades. For recognizing the face, we have used our face trainer and recognizer based on the OpenCV library using a tutorial on hackster.io (link for the tutorial given in references).

For face recognition: We have implemented the LBPH face recognition algorithm, which bases itself on Local Binary Pattern Histograms.

Working of LBP (source: https://en.wikipedia.org/wiki/Local_binary_patterns)

The LBP feature vector, in its simplest form, is created in the following manner:

- Divide the examined window into cells (e.g., 16x16 pixels for each cell).
- For each pixel in a cell, compare the pixel to each of its eight neighbors (on it's left-top, left-middle, left-bottom, right-top, etc.). Follow the pixels along a circle, i.e., clockwise or counterclockwise.
- Where the center pixel's value is higher than the neighbor's value, write "0". Otherwise, write "1". This gives an eight-digit binary number (which is usually converted to decimal for convenience).
- Compute the histogram, over the cell, of the frequency of each "number" occurring (i.e., each combination of which pixels are smaller and which are higher than the center). This histogram can be seen as a 256-dimensional feature vector.

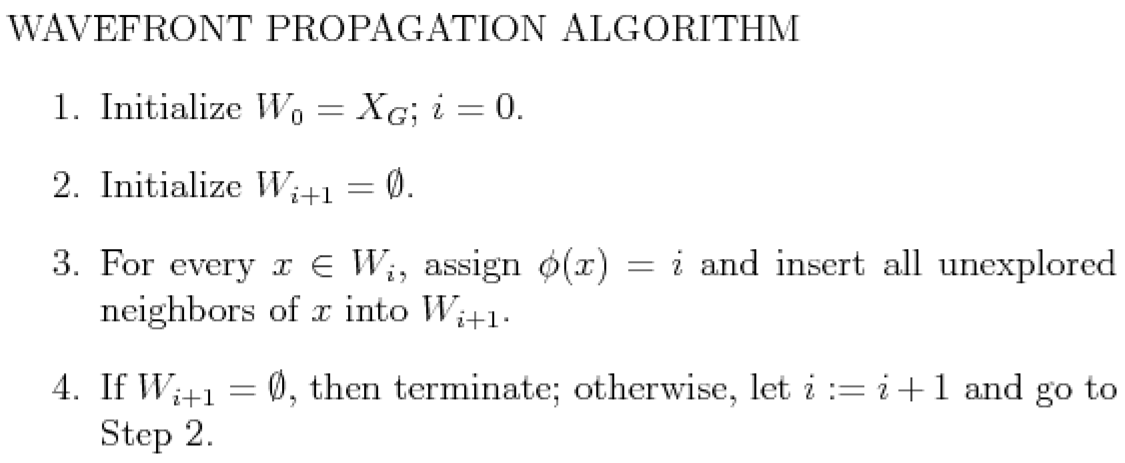
- Optionally normalize the histogram.
- Concatenate (normalized) histograms of all cells. This concatenation gives a feature vector for the entire window.

The face of a person once in the frame of the camera is being tracked by us using OpenCV and PID controls.

If a person cannot be identified, we then send an email to the security manager with a video clip of the person. The computer of the security manager will be connected to RPI via remote desktop. Once the security manager decides whether the person should be admitted in the society or not, (s)he can decide where the guest should be taken via a program running on Raspberry Pi, and on that basis will the raspberry pi take the person from one point to another.

If a person is identified, the robot will just welcome the person and then wait for the next person to arrive based on the wavefront algorithm.

.



WAVEFRONT PROPAGATION ALGORITHM

1. Initialize $W_0 = X_G$; $i = 0$.
2. Initialize $W_{i+1} = \emptyset$.
3. For every $x \in W_i$, assign $\phi(x) = i$ and insert all unexplored neighbors of x into W_{i+1}.
4. If $W_{i+1} = \emptyset$, then terminate; otherwise, let $i := i + 1$ and go to Step 2.

Source: http://planning.cs.uiuc.edu/node372.html

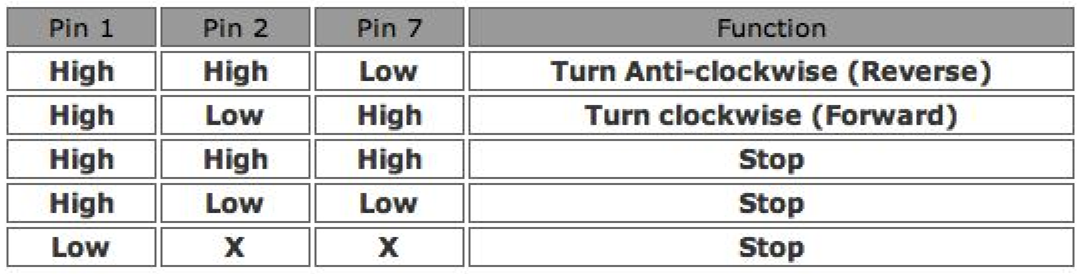
The wavefront propagation algorithm is a particular version of Dijkstra's algorithm that optimizes the number of stages to reach the goal. It computes an optimal navigation function. It can be considered as a special case of Dijkstra's algorithm that avoids explicit construction of the priority queue.

Hardware:

Functioning Of The Motors:

The DC motors are being run by the Raspberry Pi by giving appropriate signals to the L293D IC. The left wheels are rotating anti-clockwise and the right wheels clockwise for the bot to move forward. The left and the right wheels move

clockwise for the bot to rotate left, vice-versa to rotate right, and finally, the left wheels rotate clockwise and the right wheels anti-clockwise for the bot to move backward. The DC motors used are of 300 rpm and are being run on a 12V lead-acid battery.
The **DC_mtr_mov.py** program controls the movement of the bot and this is done by applying the truth-table for the L293D IC to perform the aforementioned tasks. The truth-table is mentioned below:



Pin 1	Pin 2	Pin 7	Function
High	**High**	**Low**	**Turn Anti-clockwise (Reverse)**
High	**Low**	**High**	**Turn clockwise (Forward)**
High	**High**	**High**	**Stop**
High	**Low**	**Low**	**Stop**
Low	**X**	**X**	**Stop**

High ~+5V, Low ~0V, X=Either high or low (don't care)

<u>Functioning Of The Servo Motors:</u>
Servo motor SG-90 used in our project runs on a 50 Hz pulse-modulated signal wherein a pulse of 5% duty-cycle rotates the servo to it's corresponding -90° and a duty cycle of 10% rotates the servo to its corresponding +90°. We can generate the required PWM signal through the **pigpio** module of the Raspberry Pi and rotate the servo motors to our respective needs. The **pigpio.set_servo_pulsewidth(<GPIO_PIN_NO>,X)** generates the respective PWM signal on the Raspberry Pi at the GPIO pin number mentioned. X refers to the amount of time (in microseconds) the pulse is high in a single clock cycle of the PWM signal. However, before running the **pigpio** module on the RPi, we have to first run the daemon file of the module by executing the command: **sudo pigpiod** in the terminal.
<u>Functioning Of The Sensor</u>:
We are using an HC-SR04 ultrasonic sensor in our project where it is used to measure the distance of the obstacle in front of it. It is used by first sending a Trigger pulse for a fixed duration via its **TRIG** pin and then calculating the time for which it is not detected on its **ECHO** pin which is basically an input pin. We then multiply that time with the speed of sound and we then get the distance of the obstacle in front of the sensor. The generation of a signal on the **TRIG** pin and the detection of the signal on the **ECHO** pin is carried out by using the **RPi.GPIO** module wherein we can just change the output of any GPIO pin by just configuring that pin as an output pin and then passing the command **GPIO.output(<GPIO_PIN_NO>, HIGH)**. Similarly goes the process for taking input from the pins. This sensor functioning is handled in the sensor.py program

wherein the ultrasonic sensor measures the distance of the obstacle 3 times and then calculates the average distance on the obstacle in front of it.

Github link: https://github.com/tech-chef/Main

Timeline

START OF PHASE 3

S.No.	Proposed task	Achieved task	Remark
1	Working of motors using Rpi	Y	
2	Face recognition	Y	
3	Circuit and Body	Y	

REVIEW MEET 1

S.No.	Proposed task	Achieved task	Remark
1	Taking a person to a particular place using a map (programming of motors)		
2	Mapping and moving in a map		
3	Avoiding obstacles		

REVIEW MEET 2

S.No.	Proposed task	Achieved task	Remark
1	Setting up the entire assembly		
2	Sending video via email		

REVIEW MEET 3

S.No.	Proposed task	Achieved task	Remark
1	Setting up the Entire Assembly		

FINAL PRESENTATION

S.No.	Proposed task	Achieved task	Remark

Budget

Item & Description	Justification	Purchase Link	Cost	Quantity
Raspberry pi 3B+	Microcontroller		3090	1
PiCam	Camera for face recognition		590	1
SG 90 Servo Motor	To pan/tilt camera and for ultrasonic sensors		549	3

300 RPM 12 V DC	To make the robot move		354	2
Wheels	Wheels to make the robot move		47.20	2
2 Axis Pan tilt brackets	For tilt/pan camera		180	1
Plastic Sheets	For the body of the robot		283.20	3
L Clamp			83	10
Battery holder			75.4	2
Nut Bolt			35.4	20
Duracell battery	To power servo motors		168	4
Connector			20	2
9V Battery			25	1
HDMI to VGA adapter	Use: Programming using a monitor		700	1
12V battery (rechargeable)	To power DC motor		566.4	1
Jumper wire			218.3	90
Micro SD Card 16GB - SanDisk	Used by Raspberry Pi for data storage		279	1
----------------------	--------------------	-------------------------	------------	------------
Servo Motor SG90	For the pan movement of Ultrasonic Sensor		150	1
12V DC Motor	For the movement of the bot		600	4

Double-Sided Tape	To stick the components		100	1
Borg Strip			10	1
IC Base			10	1
Clamp+PVC+Ply			200	1
Raspberry Pi 3B+ (MangalDeep)			3400	1
SD Card-32GB (SanDisk)			400	1
HC-SR04 Sensor			200	1
Heat Sinks			106	1
Raspberry Pi 3B+ (RoboKits)			3134	1

Total Claim: 15573.90 INR

References and acknowledgments:

We have used the following websites for our project:

1. https://www.raspberrypi.org -> for setting up of raspberry pi 3B+
2. https://www.pyimagesearch.com -> for installing OpenCV and for tracking the face
3. https://www.societyofrobots.com -> for developing the mapping program.
4. https://www.hackster.io/mjrobot/real-time-face-recognition-an-end-to-end-project-a10826 -> For Face training and recognition.

We would also like to thank Parth Patil and Lovekush Tak for helping us out, and Anirudh Mittal for heading this excellent program.