2018

Implementing face recognition on Raspberry Pi using Kinect camera

Kanaka Sunanda Vemulapalli

Follow this and additional works at: https://huskiecommons.lib.niu.edu/allgraduate-thesesdissertations

Recommended Citation
https://huskiecommons.lib.niu.edu/allgraduate-thesesdissertations/3498

This Dissertation/Thesis is brought to you for free and open access by the Graduate Research & Artistry at Huskie Commons. It has been accepted for inclusion in Graduate Research Theses & Dissertations by an authorized administrator of Huskie Commons. For more information, please contact jschumacher@niu.edu.
IMPLEMENTING FACE RECOGNITION ON RASPBERRY PI USING KINECT CAMERA

Kanaka Sunanda Vemulapalli, MS
Department of Electrical Engineering
Northern Illinois University, 2018
Dr. Reza Hashemian, Director

This work implements face recognition technique on Raspberry Pi using Kinect Xbox 360 camera and IRobot. The aim of this project is to create a cost-effective standalone face recognition system to use in search and rescue operations or in any hazardous areas. To achieve this, I am using Raspberry Pi as processor, Kinect camera to capture live feeds from the scene, and Roomba to move around the location. Anyone can access this program with appropriate setup.

We will ask Roomba to find out a specific person in a room. By moving in a predefined path, Roomba will identify every person and stops in front of the person to verify he/she is the person specified. It will continue this until it identifies the required person, and when it does, it will stop and say, “I found this person.” This can be done in two steps: face detection to locate the person and face recognition to recognize the person in a given frame.
IMPLEMENTING FACE RECOGNITION ON RASPBERRY PI USING KINECT CAMERA

BY

KANAKA SUNANDA VEMULAPALLI
©2018 Kanaka Sunanda Vemulapalli

A THESIS SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE
MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL ENGINEERING

Thesis Director:
Reza Hashemian
ACKNOWLEDGEMENTS

I would like to take this opportunity to thank my advisor and mentor, Dr. Reza Hashemian, for his continuous support, patience, motivation, enthusiasm, immense knowledge and generous attitude. The project idea and shaping were done by Dr. Hashemian.

Besides my advisor, I would like to acknowledge my thesis committee members: Dr. Mansour Tahernezhadi, Senior Associate Dean and Acting Department Chair, for encouraging me right from the beginning of my graduate studies, and Dr. Donald Zinger, Associate Professor, for his continuous support. I am highly indebted to my friends Bhanu Prakash, Karthik Reddy, Yashodeep Patil, Maruthi Sivakami and others who were part of my testing and demo work.

I must express my profound gratitude to my parents, Mr. V. Prabhakara Rao and Mrs. V. Vidyadhari, for giving me such a blessed life and their constant love and support, and my sisters Sravanthi and Prathyusha, who have been a constant source of encouragement in everything I chose to do. Last but not least, I would like to thank the love of my life, Kartheek Chintalapati, for being my pillar of support throughout. This thesis would have not been possible without any of these people.
DEDICATION

To Almighty! My parents, Mrs. and Mr. Prabhakara Rao, sister Sravanthi Vemulapalli, and my boyfriend, Kartheek Chintalapati
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>ix</td>
</tr>
<tr>
<td>CHAPTER-1: LITERATURE REVIEW</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Face Detection</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Face Recognition</td>
<td>1</td>
</tr>
<tr>
<td>1.3 OpenCV</td>
<td>2</td>
</tr>
<tr>
<td>1.3.1 Eigen Faces</td>
<td>2</td>
</tr>
<tr>
<td>1.3.2 Fisher Faces</td>
<td>4</td>
</tr>
<tr>
<td>1.3.3 Local Binary Patterns Histograms (LBPH) Face Recognizer</td>
<td>6</td>
</tr>
<tr>
<td>1.3.3.1 Training the Algorithm</td>
<td>7</td>
</tr>
<tr>
<td>1.3.3.2 Applying the LBP Operation</td>
<td>7</td>
</tr>
<tr>
<td>1.3.3.3 Extracting the Histograms</td>
<td>8</td>
</tr>
<tr>
<td>1.3.3.4 Performing the Face Recognition</td>
<td>8</td>
</tr>
<tr>
<td>1.4 Multiprocessing Technique</td>
<td>9</td>
</tr>
<tr>
<td>CHAPTER-2: HARDWARE</td>
<td>10</td>
</tr>
<tr>
<td>2.1 Raspberry Pi 3 Model B</td>
<td>10</td>
</tr>
<tr>
<td>2.1.1 Connectivity of Pi</td>
<td>12</td>
</tr>
<tr>
<td>2.1.1.1 External Display</td>
<td>12</td>
</tr>
<tr>
<td>2.1.1.2 Putty</td>
<td>12</td>
</tr>
<tr>
<td>2.1.1.3 Ethernet Cable</td>
<td>13</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.1.1.4 Virtual Network Computing (VNC)</td>
<td>13</td>
</tr>
<tr>
<td>2.2 Xbox 360 Kinect</td>
<td>14</td>
</tr>
<tr>
<td>2.2.1 Distance from camera to subject</td>
<td>17</td>
</tr>
<tr>
<td>2.3 iRobot Roomba Create2</td>
<td>18</td>
</tr>
<tr>
<td>CHAPTER-3: PROCEDURE</td>
<td>21</td>
</tr>
<tr>
<td>3.1 Block Diagram</td>
<td>21</td>
</tr>
<tr>
<td>3.2 Procedure</td>
<td>22</td>
</tr>
<tr>
<td>3.2.1 Principal Component Analysis</td>
<td>22</td>
</tr>
<tr>
<td>3.2.2 Cam capture</td>
<td>23</td>
</tr>
<tr>
<td>3.2.3 Face detection</td>
<td>24</td>
</tr>
<tr>
<td>3.2.3.1 AdaBoost Learning Algorithm</td>
<td>25</td>
</tr>
<tr>
<td>3.2.3.2 Cascade Classifier</td>
<td>25</td>
</tr>
<tr>
<td>3.2.3.3 Haar cascade classifier</td>
<td>27</td>
</tr>
<tr>
<td>3.2.3.4 LBP Detection Algorithm</td>
<td>28</td>
</tr>
<tr>
<td>3.2.4 Face recognition</td>
<td>29</td>
</tr>
<tr>
<td>3.3 Combined output – result deduction</td>
<td>31</td>
</tr>
<tr>
<td>3.4 Roomba</td>
<td>32</td>
</tr>
<tr>
<td>3.5 Making the project standalone</td>
<td>33</td>
</tr>
<tr>
<td>CHAPTER-4: RESULTS AND CONCLUSION</td>
<td>35</td>
</tr>
<tr>
<td>4.1 Results</td>
<td>35</td>
</tr>
</tbody>
</table>
4.2 Applications.............................................................................................................. 37

4.3 Improvements.......................................................................................................... 37

REFERENCES...................................................................................................................... 40
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Demonstration of the concept of sliding window</td>
<td>7</td>
</tr>
<tr>
<td>1.2</td>
<td>Extracting the Characteristics</td>
<td>8</td>
</tr>
<tr>
<td>2.1</td>
<td>Raspberry Pi</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>HDMI port</td>
<td>12</td>
</tr>
<tr>
<td>2.3</td>
<td>Putty interface</td>
<td>13</td>
</tr>
<tr>
<td>2.4</td>
<td>X11 VNC remote desktop</td>
<td>14</td>
</tr>
<tr>
<td>2.5</td>
<td>Xbox 360 Kinect</td>
<td>14</td>
</tr>
<tr>
<td>2.6</td>
<td>IR sensor array of Kinect</td>
<td>16</td>
</tr>
<tr>
<td>2.7</td>
<td>Distance information displayed on the top of the detected face</td>
<td>18</td>
</tr>
<tr>
<td>2.8</td>
<td>iRobot Create® 2 Programmable Robot</td>
<td>19</td>
</tr>
<tr>
<td>2.9</td>
<td>Mini-DIN connector pinout</td>
<td>20</td>
</tr>
<tr>
<td>3.1</td>
<td>Block diagram explaining the process</td>
<td>21</td>
</tr>
<tr>
<td>3.2</td>
<td>PCA calculation</td>
<td>22</td>
</tr>
<tr>
<td>3.3</td>
<td>Pi's GUI when program is running</td>
<td>23</td>
</tr>
<tr>
<td>3.4</td>
<td>General features of a human face</td>
<td>24</td>
</tr>
<tr>
<td>3.5</td>
<td>Cascade classifier</td>
<td>25</td>
</tr>
<tr>
<td>3.6</td>
<td>Depth information displayed on a detected image</td>
<td>26</td>
</tr>
<tr>
<td>3.7</td>
<td>Rectangular feature</td>
<td>27</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.8</td>
<td>LBP conversion to binary</td>
<td>28</td>
</tr>
<tr>
<td>3.9</td>
<td>Resized gray image</td>
<td>29</td>
</tr>
<tr>
<td>3.10</td>
<td>Dr. Reza Hashemian’s database</td>
<td>30</td>
</tr>
<tr>
<td>3.11</td>
<td>RGB recognition took 6.9 seconds with echo</td>
<td>31</td>
</tr>
<tr>
<td>3.12</td>
<td>RGB face recognition took 3.7 seconds without echo</td>
<td>31</td>
</tr>
<tr>
<td>3.13</td>
<td>Final standalone system</td>
<td>34</td>
</tr>
<tr>
<td>4.1</td>
<td>Testing environment</td>
<td>35</td>
</tr>
<tr>
<td>4.2</td>
<td>Roomba identified Dr. Zinger (Thesis Committee member)</td>
<td>36</td>
</tr>
<tr>
<td>4.3</td>
<td>Parallel processing</td>
<td>38</td>
</tr>
<tr>
<td>4.4</td>
<td>Comparison between pictures taken with Kinect v2 and v1</td>
<td>39</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 2.1: Features of Kinect................................................................. 16
Table 2.2: Mini-DIN Connector Pin Description...................................... 20
Table 3.1: Result Deduction Between RGB and IR.................................. 32
Table 3.2: Result Deduction Between Recognizer Algorithms.................... 32
CHAPTER-1

LITERATURE REVIEW

1.1 Face Detection:

Face detection determines if there are any faces in the image and where are they located. This step takes an image as input and produces patches containing all the faces in the image. In order to make face recognition system more robust and easy to design, face alignment is performed to justify the scales and orientations of these patches. Besides serving as the preprocessing for face recognition, face detection could be used for region-of-interest detection, retargeting, video and image classification, etc.

1.2 Face Recognition:

With the information vector in hand, the next and final step would be to recognize the faces. Face recognition step takes in the feature vectors as input and identifies the faces. To achieve this, a face database is built with multiple images of persons. These images are used to extract features of the person. The features extracted in the last step are compared to each face class in the database and a match is identified. Over the past two decades, face recognition has been studied extensively. Of the many algorithms developed, some of the most common ones are Eigen faces, Fisher faces and local binary patterns histograms.
1.3 OpenCV:

OpenCV (Open Source Computer Vision) is a popular computer vision library started by Intel in 1999. The cross-platform library sets its focus on real-time image processing and includes patent-free implementations of the latest computer vision algorithms [1]. The currently available algorithms are:

- Eigen faces
- Fisher faces
- Local binary patterns histograms

1.3.1. Eigen Faces:

The Eigen faces method described in [2] took a holistic approach to face recognition. A facial image is a point from a high-dimensional image space and a lower dimensional representation is found, where classification becomes easy. The lower dimensional subspace is found with principal component analysis, which identifies the axes with maximum variance. While this kind of transformation is optimal from a reconstruction standpoint, it does not take any class labels into account. The basic idea is to minimize the variance within a class while maximizing the variance between the classes at the same time.

In linear algebra the eigenvectors of a linear operator are non-zero vectors which, when operated by the operator, result in a scalar multiple of them. Scalar is then called eigenvalue (λ) associated with the eigenvector (X). Eigenvector is a vector that is scaled by linear transformation. It is a property of matrix. When a matrix acts on it, only the vector magnitude is changed, not the direction.

\[ AX = \lambda X, \]  

where A is a vector function
\[(A - \lambda I)X = 0\] where I is the identity matrix

This is a homogeneous system of equations and form fundamental linear algebra. We know a non-trivial solution exists if and only if \(\text{Det}(A - \lambda I) = 0\), where \(\text{det}\) denotes determinant. When evaluated, it becomes a polynomial of degree \(n\). This is called characteristic polynomial of \(A\). If \(A\) is \(N\) by \(N\) then there are \(n\) solutions or \(n\) roots of the characteristic polynomial. Thus, there are \(n\) eigenvalues of \(A\) satisfying the equation:

\[AX = \lambda X\]

If the eigenvalues are all distinct, there are \(n\) associated linearly independent eigenvectors, whose directions are unique, which span an \(n\) dimensional Euclidean space.

1. Compute the mean \(\mu\)

\[\mu = \frac{1}{n} \sum_{i=1}^{n} x_i\]

2. Compute the covariance matrix \(S\)

\[S = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^T\]

3. Compute the eigenvalues \(\lambda_i\) and eigenvectors \(v_i\) of \(S\)

\[Sv_i = \lambda_i v_i, i = 1,2,\ldots,n\]

4. Order the eigenvectors descending by their eigenvalue. The \(k\) principal components are the eigenvectors corresponding to the \(k\) largest eigenvalues.

The \(k\) principal components of the observed vector \(x\) are then given by:

\[y = W^T(x - \mu)\]

where \(W = (v_1, v_2, \ldots, v_k)\).
The reconstruction from the PCA basis is given by:

\[ x = W y + \mu \]

where \( W = (v_1, v_2, ..., v_k) \).

One of the best methods among appearance-based recognition process is Eigen face method. Though it was an initial successful method, it has many disadvantages, like illumination effect, pose, scale variation in face, facial changes, occlusion, etc. To overcome these cons we have used enhanced techniques in the next step [2].

1.3.2. Fisher Faces:

The previous algorithm takes advantage of the fact that, under admittedly idealized conditions, the variation within class lies in a linear subspace of the image space. Hence, the classes are convex and therefore linearly separable. One can perform dimensionality reduction using linear projection and still preserve linear separability. This is a strong argument in favor of using linear methods for dimensionality reduction in the face recognition problem, at least when one seeks insensitivity to lighting conditions. Since the learning set is labeled, it makes sense to use this information to build a more reliable method for reducing the dimensionality of the feature space.

Here we argue that using class-specific linear methods for dimensionality reduction and simple classifiers in the reduced feature space, one may get better recognition rates than with either the linear subspace method or the Eigen face method. Fisher’s linear discriminant (FLD) is an example of a class-specific method, in the sense that it tries to “shape” the scatter in order to make it more reliable for classification. This method calculates the ratio of the between-class scatter and the within-class scatter should be maximum value [3].
One thing to note here is that Fisher faces only prevents features of one person from becoming dominant, but it still considers illumination changes as a useful feature. We know that light variation is not a useful feature to extract, as it is not part of the actual face.

The variance among faces in the database may come from distortions such as illumination, facial expression, and pose variation. And sometimes these variations are larger than variations among standard faces! The images of a particular face, under varying illumination but fixed pose, lie in a 3D linear subspace of the high-dimensional image space (without shadowing).

We try to find a basis for projection that minimizes the intra-class variation but preserves the inter-class variation. Rather than explicitly modelling this deviation, we linearly project the image into a subspace in a manner which discounts those regions of the face with large deviation [4].

Let $X$ be a random vector with samples drawn from $c$ classes:

$$X = \{X_1, X_2, \ldots, X_c\}$$

$$X_i = \{x_1, x_2, \ldots, x_n\}$$

The scatter matrices $S_B$ and $S_W$ are calculated as:

$$S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

$$S_W = \sum_{i=1}^{c} \sum_{x_j \in X_i} (x_j - \mu_i)(x_j - \mu_i)^T$$

where $\mu$ is the total mean:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$
and $\mu_i$ is the mean of class $i \in \{1, ..., c\}$:

$$\mu_i = \frac{1}{|X_i|} \sum_{x_j \in X_i} x_j$$

Fisher’s classic algorithm now looks for a projection $W$ that maximizes the class separability criterion:

$$W_{opt} = \arg \max_W \frac{|W^T S_B W|}{|W^T S_W W|}$$

Following [5], a solution for this optimization problem is given by solving the General Eigenvalue Problem [6]:

$$S_B v_i = \lambda_i S_w v_i$$

$$S_W^{-1} S_B v_i = \lambda_i v_i$$

### 1.3.3. Local Binary Patterns Histograms (LBPH) Face Recognizer:

A common setback observed in both Eigen faces and Fisher faces is that both are affected by light and it is almost impossible to provide perfect light conditions everywhere. LBPH face recognizer overcomes this drawback by looking at the image at a pixel level rather than the whole image itself. LBPH construct a local structure from comparison of a pixel with neighboring pixels.

LBPH has been an important feature for texture classification since 1994. Studies have proven that a combination of LBP and HOG (histograms of oriented gradients) descriptor has shown considerable amount of performance improvement on some datasets as face images can be represented with a simple data vector. We can use LBP in face recognition as described below.
1.3.3.1. Training the Algorithm:

As the first step, we train the algorithm using facial images of our intended test subjects. We will construct an image dataset with images of subjects. All the images of a particular person will be given the same ID to differentiate between sets. We use this ID to identify a given image. Once tagging of images is done, the next step would be applying the LBP operation.

1.3.3.2. Applying the LBP Operation:

By using a concept of sliding window based on neighbors and radius, we create an intermediate image as shown in Figure 1.1 [7]. The explanation follows.

![Figure 1.1: Demonstration of the concept of sliding window.](image)

Let’s assume that we have a grayscale image consisting of a face. Take a 3X3 window from the image. This can also be represented as 3X3 matrix with the value of each element ranging between 0 and 255 (intensity value of each pixel). Now consider the center pixel’s value as the threshold. Based on the threshold, every pixel value that falls above is taken as 1 and below as 0. All these 0’s and 1’s form an 8-bit binary number. Convert the binary number into decimal number. Iterate the process all over the image and now we have a new image with all these values as intensity value of each pixel that better represents the original image.
1.3.3.3. Extracting the Histograms:

Using the Grid X and Grid Y parameters, we can divide the image from the last step into multiple grids as shown in Figure 1.2 [7]. The histogram of each region can be extracted as follows.

![Image](image_url)

**Figure 1.2: Extracting the characteristics.**

As we have a grayscale image, our histogram will have only 256 positions. Values of image from each grid will be plotted as a separate histogram and we will concatenate all the histograms to form a new and final histogram. This histogram represents the characteristics of the original image.

1.3.3.4. Performing the Face Recognition:

In this step, we will create histograms for every image in the training dataset using the method from previous step. When a new input image is given, we will form a histogram for that image too and compare it with the dataset. There are various approaches to do this, like Euclidean distance, absolute value, chi-square and so on. For example, consider Euclidean distance:

\[
D = \sqrt{\sum_{i=1}^{n} (hist1_i - hist2_i)^2}
\]

The ID of the closest image along with the calculated distance is considered. The distance is called confidence. We will determine the accuracy of algorithm with the confidence and threshold
values. If confidence is less than the threshold, then we will say that the face is correctly recognised [8].

1.4 Multiprocessing Technique:

Multiprocessing, similar to multithreading is a package that spawns processes by creating a process object and then calling its start method. This package offers concurrency and eliminates the global interpreter lock by using subprocesses instead of threads, thus letting the programmer make use of all the processors. If processes need some shared data, it is handled through processes like shared memory. Data can be stored in a shared memory map either as a value or an array [9].
CHAPTER-2

HARDWARE

The three main hardware components used in this project are Raspberry Pi 3 Model B, Kinect Xbox 360 camera, IRobot Roomba Create2. If we compare the whole project setup to a human body, Kinect Xbox 360 camera would be eyes to see and capture images, IRobot Roomba Create2 would be legs for movement and Raspberry Pi 3 Model B would be brain controlling these two.

Besides the above components, I have also used a JBL speaker to listen to the voice output, auto drive power bank to power Pi, TP-Link N150 USB WiFi adapter to connect Raspberry Pi to a WiFi network, Weifeng WT-3110A lightweight aluminium tripod to mount Kinect Xbox 360 camera. The camera is mounted on a Tripod so that it can detect a person’s face when the person is in a sitting posture.

2.1 Raspberry Pi 3 Model B:

Above face recognition technique will be implemented on Raspberry Pi 3 Model B (hereafter referred to as Pi). The Raspberry Pi is a single computer board developed to encourage and aid the teaching of programming and computing. The Pi is the third-generation Raspberry Pi (Figure 2.1). It replaced the Raspberry Pi 2 Model B in February 2016. Pi has a few advancements:

- 1.2GHz 64-bit quad-core ARMv8 CPU
- 802.11 b/g/n wireless LAN
- Bluetooth 4.1
Bluetooth low energy (BLE)

Like other Raspberry Pi models, Pi also has

- 1GB RAM, 4 USB ports, 40 GPIO pins
- Full HDMI port
- Ethernet port
- 1X10/100 LAN port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot (now push-pull rather than push-push)
- Video Core IV 3D graphics core
- 85.6 X 56 X 21mm size.

Figure 2.1: Raspberry Pi [10].

Raspbian is the recommended Linux-based operating system for Raspberry Pi. Raspbian comes with plenty of pre-installed software for education, programming and general use. It has
software like Python, Scratch, Sonic Pi, Java, Mathematica and more. I am using RASPBIAN JESSIE with pixel.

2.1.1 Connectivity of Pi:

There are several ways to access Pi.

2.1.1.1 External Display:

External display is the simplest of all. Connect the HDMI port of the Raspberry Pi to a HDMI (Figure 2.2) to VGA connector to connect it to your desktop or television display.

![HDMI port](image)

Figure 2.2: HDMI port.

2.1.1.2 Putty:

Putty is an open-source terminal emulator, SSH and telnet client which helps Raspberry Pi to get connected with another computer [11] (Figure 2.3).
2.1.1.3 Ethernet Cable:

To connect a Raspberry Pi to a laptop display, you can simply use an Ethernet cable. The Raspberry Pi’s desktop GUI (Graphical User Interface) can be viewed through the laptop display using a 100Mbps Ethernet connection between the two.

2.1.1.4 Virtual Network Computing (VNC):

There are many software programs available that can establish a connection between a Raspberry Pi and your laptop and one of them is VNC server (Figure 2.4). Installing a VNC server on Raspberry Pi allows you to see the Raspberry Pi’s desktop remotely, using the mouse and keyboard as if you were sitting right in front of your Pi. It also means that you can put your Pi anywhere else in your home and still control it. Also, the internet can be shared from your laptop’s Wi-Fi over Ethernet. This also lets you access the internet on the Pi and connect it to your laptop display [12]. This is the most used method for this project.
Figure 2.4: X11 VNC remote desktop.

2.2 Xbox 360 Kinect

Developed by Microsoft, an Xbox 360 Kinect is a bank of cameras connected within a single device having motion-sensing abilities (Figure 2.5).

Figure 2.5: Xbox 360 Kinect.
1. 3-D depth sensors consist of an infrared laser projector combined with a monochrome CMOS sensor with resolution 640 X 480 pixels with 11-bit depth.

2. RGB camera with resolution 640 X 480 pixels is capable of resolution up to 1280 X 1024 at lower frame rate.

3. The microphone array features four microphone capsules capable of voice recognition.

4. Motorized pivot is capable of tilting the sensor up to $27^\circ$ either up or down.

The Kinect is an amazing and intelligent piece of hardware. It has a RGB camera, an IR laser projector, an IR CMOS sensor, a servo to adjust the tilt of the device and a microphone array. Unlike other RGB cameras Kinect is well known for depth sensor which gives us 3D view. Depth stream supports 640 x 480 px, 320 x 240 px and 80 x 60 px resolutions.

An LED in the Kinect device is used to indicate the status that the Kinect device drivers have loaded properly. It shows green color when the Kinect is connected to the computer and tells that device is ready for use to create applications [13]. Table 2.1 outlines some of the features of Kinect.

The Kinect device contains two depth sensors: IR emitter and IR depth sensor. The IR emitter depth sensor is mounted as a camera on Kinect, actually it is an IR projector which emits the infrared light on the objects in a random dot pattern.

The infrared light is projected on the objects in the dot pattern which is captured by IR depth sensor (Figure 2.6). IR depth sensor captures depth information from the dotted light reflected off different objects. This invisible dot information is used to calculate the distance between the sensor and the object from where the IR dot was read and is transformed into depth data [13].
Table 2.1: Features of Kinect

<table>
<thead>
<tr>
<th>Feature</th>
<th>Kinect for Windows 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color Camera</td>
<td>640 X 480 @30 fps</td>
</tr>
<tr>
<td>Depth Camera</td>
<td>320 X 240</td>
</tr>
<tr>
<td>Max Depth Distance</td>
<td>~4.5 M</td>
</tr>
<tr>
<td>Min Depth Distance</td>
<td>40 cm in near mode</td>
</tr>
<tr>
<td>Horizontal Field of View</td>
<td>57 degrees</td>
</tr>
<tr>
<td>Vertical Field of View</td>
<td>43 degrees</td>
</tr>
<tr>
<td>Tilt Motor</td>
<td>Yes</td>
</tr>
<tr>
<td>Skeleton Joints Defined</td>
<td>20 joints</td>
</tr>
<tr>
<td>Full skeletons Tracked</td>
<td>2</td>
</tr>
<tr>
<td>USB Standard</td>
<td>2.0</td>
</tr>
<tr>
<td>Supported OS</td>
<td>Win7, Win8</td>
</tr>
<tr>
<td>Price</td>
<td>$299</td>
</tr>
</tbody>
</table>

Figure 2.6: IR sensor array of Kinect.
The PrimeSense chip sends a signal to the IR emitter to turn on the infrared light to capture the depth data. In addition, the chip also sends a signal to the IR depth sensor to initialize the depth sensor. The IR emitter starts emitting an electromagnetic radiation to the objects in front of the camera. The sensor's IR lights are invisible because, the wavelengths of the radiations are longer than the wavelength of the visible light. The IR depth sensor captures depth information and obtains the distance between the sensor and the object from where the IR dot was read. The depth sensor returns the coded depth light to the PrimeSense chip. The PrimeSense chip processes the depth stream and forms a frame-by-frame depth stream to create the output display data and form a depth image ready for the display. Raw depth values are transformed into meters [14]

\[
\text{Depth} = 1.0 / (\text{raw}_{\text{depth}} \times -0.0030711016 + 3.3309495161)
\]

Libfreenect is a userspace driver for the Microsoft Kinect. It runs on Linux, OSX, and Windows and supports:

- RGB and depth images
- Motors
- Accelerometer
- LED
- Audio

2.2.1 Distance from Camera to Subject:

OpenCV’s depth function freenect_depth_mm [14] didn’t yield the desired results. So, I did some study to find the transformation of raw depth values into meters. The formula used to measure distance in meters using depth parameters is:

\[
\text{Depth} = 1.0 / (\text{raw}_{\text{depth}} \times -0.0030711016 + 3.3309495161)
\]
This project is using a threshold of 0.5m. This will print out the distance of the person in meters onto the screen (Figure 2.7).

![Figure 2.7: Distance information displayed on the top of the detected face.](image)

### 2.3 iRobot Roomba Create2

The iRobot Roomba Create2 (hereafter referred to as Roomba) is an autonomous robot vacuum cleaner (Figure 2.8). We use Python to control Roomba via the serial port. Since Roomba communicates with 5V serial and the Raspberry Pi communicates at 3.3V, we need to convert levels so as not to damage any hardware. Roomba is a mobile robot platform designed for educational purposes and comes pre-assembled. Other Roomba features include:

- Serial cable sends commands from a computer or other microcontroller to the robot.
- Preprogrammed behaviors can be controlled via Open Interface Commands
- Built-in sensors allow the robot to react to its environment
- Rechargeable battery charges in three hours [13].
To use the open interface (OI), we need to connect a processor to the Mini-DIN connector on Roomba. This connector provides two-way, serial communication at TTL (0 – 5V) levels. The connector also provides direct supply to Roomba’s battery, which we can use to power the OI applications. The Mini-DIN connector is located on the top of Roomba [15]. Figure 2.9 gives a rough sketch of how the Mini-DIN connector looks like and Table 2.2 describes the pinouts.
Figure 2.9: Mini-DIN connector pinout.

Table 2.2: Mini-DIN Connector Pin Description

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>V_pwr</td>
<td>Roomba battery + (unregulated)</td>
</tr>
<tr>
<td>2</td>
<td>V_pwr</td>
<td>Roomba battery + (unregulated)</td>
</tr>
<tr>
<td>3</td>
<td>RXD</td>
<td>0 – 5V Serial input to Roomba</td>
</tr>
<tr>
<td>4</td>
<td>TXD</td>
<td>0 – 5V Serial output to Roomba</td>
</tr>
<tr>
<td>5</td>
<td>BRC</td>
<td>Baud Rate Change</td>
</tr>
<tr>
<td>6</td>
<td>GND</td>
<td>Roomba battery ground</td>
</tr>
<tr>
<td>7</td>
<td>GND</td>
<td>Roomba battery ground</td>
</tr>
</tbody>
</table>
CHAPTER-3

PROCEDURE

3.1 Block Diagram:

Figure 3.1 outlines the process as a block diagram. A name is given as input. Then the camera captures a frame/image followed by face detection. Once a face is detected, Roomba will stop and perform face recognition. If the person recognized is the desired person, Pi will echo the name. Otherwise, Roomba moves and repeats from camera capture.

Figure 3.1: Block diagram explaining the process.
3.2 Procedure:

3.2.1 Principal Component Analysis:

Initially, PCA is performed on training image set and then on test images. Then we will calculate the Euclidian distance between each test and training images and pick the one with the smallest distance [16].

This is executed on both single core and multicore and it is observed that it works very slow with delays. Figure 3.2 shows that PCA alone took 7.3 seconds. Also, it is finding the nearest match even if there is no entry of the person in the database by calculating the minimum Euclidian distance. In the final working model, this method is replaced with other algorithms that give a better output and these methods will be discussed later in the document.

![Figure 3.2: PCA calculation.](image)

A program that creates four processes is written using OpenCV libraries in Python to utilize all the cores of Pi. Libfreenect libraries [17] are installed in Pi to work with Kinect.
3.2.2 Cam Capture:

Some of the widely used libfreenect functions [14] on Kinect Xbox 360 are `sync_get_video`, `sync_get_depth`, `resolution_medium`, `depth_10bit`, and `video_ir_10bit`. In this step, the camera will capture an image using `sync_get_video()`. Then we convert RGB-formatted image into BGR format using `cv2.COLOR_RGB2BGR`; `sync_get_video`, `sync_get_depth` captures an IR image. The salt and pepper noise in the original IR image is removed using MedianBlur.

Right-side RGB frame and distance measured are displayed in green color and left side resized gray image with detection results (Figure 3.3).

![Figure 3.3: Pi's GUI when program is running.](image)
3.2.3 Face Detection:

The objective of this step is to find any faces in an image and extract them to be used by the face recognition algorithm. Detecting a face is always easier than recognizing it. So, to detect a face in any image, we need to construct a prototype or a general structure of a face. Consider Figure 3.4. Each of the panel in Figure 3.4 represents a human face feature. When we combine all these features, we will be able to construct a prototype [4].

Paul Viola and Michael Jones, in their paper, “Rapid Object Detection Using a Boosted Cascade of Simple Features,” proposed object detection using Haar feature-based cascade classifiers [18]. They came up with a method of rectangular Haar-like features with AdaBoost learning algorithm combined with a cascade of strong classifiers. I used this concept for face detection. The first step is to load the classifier which distinguishes between faces and other objects in a frame.
3.2.3.1 AdaBoost Learning Algorithm:

Paul Viola and Michael Jones [19] used AdaBoost learning algorithm in the previous mentioned paper to select a specific Haar-like feature as a threshold. AdaBoost is used to create strong classifier from combining a collection of weak classification functions [20]. The strongest classifier uses the strongest feature, which is the best Haar-like feature, that is, the feature that best separates the positive and negative samples.

3.2.3.2 Cascade Classifier

Cascade classifier [19] is a chain of weak classifiers for efficient classification of image regions. Its goal is to increase the performance of object detection and to reduce the computational time. As shown in Figure 3.5, each node in the chain is a weak classifier and filter for one Haar feature. AdaBoost gives weights to the nodes, and the highest weighted node comes first.

![Figure 3.5: Cascade classifier.](image-url)
When a filter fails to pass image regions, that specific subwindow of the image is eliminated for further processing. It is then considered as a nonobjective, meaning that the image regions processed do not contain the object to be detected. This is very crucial to the performance of the classifier, since all or nearly all negative image subwindows will be eliminated in the first stage. On the other hand, when image regions successfully passed the filter, they go to the following stage, which contains a more complex filter. Only regions that successfully pass all filters are considered to contain a match of the object. This means that regions of the image contain the object subject to detection (Figure 3.6).

The reason behind the multistage classifier is to reject efficiently and rapidly the nonobject subwindows. The next nodes in the chain in Figure 3.5 represent complex classifiers in the case of face detection. The classifier is used to reject more false positives (nonface regions) of the subwindows [19]. The number of false positives is radically reduced after several steps of processing [7].

![Figure 3.6: Depth information displayed on a detected image.](image.png)
According to [7], there are three types of Haar-like features. The first type is the edge feature, which is represented in Figure 3.7 by the two upper squares. The second type is the line feature (Figure 3.7 lower left square). The last type of features is the center-surround feature (Figure 3.7 lower right square).

As described in [4], the idea behind Haar-like feature selection algorithm is simple. It lies on the principle of computing the difference between the sum of white pixels and the sum of black pixels. The main advantage of this method is the fast sum computation using the integral image. It is called Haar-like because it is based on the same principle of Haar wavelets [21].

3.2.3.3 Haar Cascade Classifier

In Haar cascade classifier, instead of applying all the features on a window at once, we will apply them as groups in different steps. If a window doesn’t pass the first step, it will not be
considered for the subsequent steps. In this manner, we will keep discarding windows at every step until we have a window that passes all the steps. This is called a face region [19].

I have used a XML Haar cascade classifier in this project [13]. We feed each frame in a grayscale format with appropriate size from the first process to detect faces in the frame. This is done on both RGB and IR images. The output image will be 112 X 92px in size.

### 3.2.3.4 LBP Detection Algorithm

The algorithm used is a variant of the cascade algorithm introduced by Viola and Jones and uses LBP features instead of Haar-like features in order to have faster processing and boosted classifiers. The LBP algorithm slides its processing window over the object image for evaluating the successive stages of the cascade algorithm by scoring their features. Each feature is described by $3 \times 3$ neighboring rectangular areas [11].

As shown in Figure 3.8 [3], the original LBP operator uses the center pixel of $3 \times 3$ neighborhood to generate an 8-bit binary number. It labels any pixel greater than the center pixel as 1 and any pixel less than the center pixel as 0.

![Figure 3.8: LBP conversion to binary.](image)

Both LBP and the Haar-based approaches use an underlying Viola-Jones framework. The LBP approach uses a binary feature descriptor, represented as an unsigned integer. This results in:
– Faster training time and object detection performance, as binary feature representations (LBP) are faster to compute and process.

– Better tolerance to illumination variations compared to the Haar approach, as pixels are quantized to 0 or 1 depending on the "ratio" of pixel intensities and not the "actual" pixel intensities.

However, the LBP approach approximates features (pixel quantization) and this results in the Haar approach producing more accurate results [23]. The Viola-Jones approach offers real-time performance and scale/location invariance, but it still has a few disadvantages, such as intolerance to object rotations, sensitivity to illumination variations, etc [23]. Final face detection result is as shown in figure 3.9.

![Resized gray image](image)

Figure 3.9: Resized gray image.

### 3.2.4 Face Recognition

Face recognition technique has been implemented for both RGB and IR formats in two steps: training and recognition. In training, we assume that there is an existing face database for modeling. In my program, I am training my database with all the three methods, e.g., one of the Open CV’s LBPH face recognition algorithms. Figure 3.10 shows the thumbnails of Dr. Hashemian’s database.
The live-feed frame from camera will be sent to the trained model. We are using threshold method to get accurate results. If the predicted output is within threshold value, the result is from our database; otherwise, it is an unknown value. LBPH thresholds determined through trial and error method are 125 for RGB and 105 for IR.

The recognizer, when fed with an image, generates a histogram of it. It then compares that with the histograms it already has from the database. Finally, it finds the best match and returns the folder name associated with the match [24]. Now I called prediction function to check for closeness of face from my database. In my code, the perfect result comes out if prediction value is <110 or else the person will be unknown. Same functionality is used for the IR images as well. Hence, the result will be a combination of 70% RGB and 30% IR. This means that we will consider the prediction as is if RGB and IR produce same result. In any other case, we will rely more on RGB values. LBPH’s RGB recognition took close to 7 seconds with echo and 3.8 seconds without echo as shown in Figures 3.11 and 3.12. Similarly, I have implanted Eigen face recognition and Fisher’s face recognition. The combined output results are explained below.
3.3 Combined Output – Result Deduction:

We will finalize the end result from Eigen recognizer, Fisher’s recognizer and LBPH recognizer. Among the three, 40% for LBPH and 30% for the other two. Between RGB and IR, the resultant will be as tabulated in Tables 3.1 and 3.2. Let known value be C and unknown value be X.
Table 3.1: Result Deduction Between RGB and IR

<table>
<thead>
<tr>
<th>RGB</th>
<th>IR</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>C</td>
<td>X</td>
<td>C</td>
</tr>
<tr>
<td>X</td>
<td>C</td>
<td>C</td>
</tr>
</tbody>
</table>

Table 3.2: Result Deduction Between Recognizer Algorithms

<table>
<thead>
<tr>
<th>LBPH</th>
<th>Eigen</th>
<th>Fisher’s</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>X</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>X</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>X</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>X</td>
<td>C</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>C</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>C</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

3.4 Roomba

In my project, I added three functionalities for Roomba: initializing, moving and stopping the Roomba using OI opcode commands. Since the voltage levels of Mini-DIN connector and
PC serial ports are different, we need to use a converter for proper conversion. I have used iRobot Roomba serial cable as it has all the required functionalities.

The Roomba OI has four operating modes: Off, Passive, Safe, and Full. After a battery change or when power is first turned on, the OI is in “off” mode. When it is off, the OI listens at the default baud rate (115200 or 19200) for an OI start command. Once it receives the start command, we can switch between the modes with a corresponding opcode [15].

In the Roomba initialization, I have used opcodes 7 to reset, 128 to start and 131 to be in safe mode. I am using opcode 135 to run Roomba in clean mode and opcode 145 to stop. When the program starts Roomba, it starts initializing and goes to clean mode simultaneously and four processes explained above will take place. When a person is identified, Pi sends a stop command to Roomba via serial port to stop it for a second and says who the person is and then Roomba goes to clean mode again. This process goes continuously until we interrupt the program externally.

3.5 Making the Project Standalone

To make the project a standalone application, I came up with two ideas: to use external A23 battery, which is a dry-cell-type battery [18], or use Roomba as a power source. An A23 battery is an 8-cell device with voltage of 12V and capacity of 55mAh [16]. This is usually used in RF devices, but Kinect Xbox 360 needs 12V 1.08A.

Roomba can give 15V from its rechargeable battery. My professor suggested I get required values from Roomba itself. I tried various methods to reduce 15V to 12V and finally settled with LM317 DC-DC converter buck step down circuit board module. This module has a potentiometer to adjust voltages from 1.2V to 37V with current ratings 1.5A (min), 2.2A
(typical) [18]. To make Pi wireless, I used an auto drive power bank kit purchased from Walmart. The final setup is shown in Figure 3.13.
CHAPTER-4

RESULTS AND CONCLUSION

4.1 Results:

Figure 4.1: Testing environment.
Figure 4.2: Roomba identified Dr. Zinger (Thesis Committee member).
4.2 Applications:

- Search and rescue
- Biometrics – Authentication of different individuals
- Information security – Data security and saving personal identification numbers, passwords
- Access control (office access or computer logon)
- Law enforcement (eliminate duplicates in a nationwide voter registration system)

4.3 Improvements:

1. Multithreading: If we submit jobs to different threads, those jobs can be pictured as sub-tasks of a single process and those threads will usually have access to the same memory areas (i.e., shared memory). This approach can easily lead to conflicts in case of improper synchronization, for example, if processes are writing to the same memory location at the same time [17]. A safer approach (figure 4.3) is to submit multiple processes to completely separate memory locations (i.e., distributed memory). Every process will run completely independent of each other [17]. This is preferred despite the additional overhead due to the communication overhead between separate processes. Some of the processes are:
   - Frame capture and detection
   - RGB recognition
   - IR recognition
   - Combined recognition
2. **Timing issues with Roomba:** Roomba needs sufficient time to initialize, initializing another process (recognizer) before camera capturing and detection takes place.

3. **Database:** The initial data has only 10 images per person, but improving face variations with light, expression and other conditions increased the number to 20.

4. **Overclock:** We can make a device to run at a higher speed or clock time than it is designed to run. This is called overclocking. But the device may dissipate more heat in the process.

5. To avoid the thermal shutdown, we use heatsink.

6. Issues like Roomba not responding, connectivity, Kinect getting disconnected, Pi restarting abruptly and so on can be resolved if a personal local network is established. One such idea is using mobile hotspot instead of public access point.

7. Bad illumination can be eliminated by using multiple recognition methods.
8. The Kinect v2’s face recognition, motion tracking, and resolution are much more precise than the Kinect v1. The Kinect v2 has 1080 resolution (HD), and from Figure 4.4 you can see the difference between images (Amazon, $499).

![Comparison between pictures taken with Kinect v2 and v1.](image)

Figure 4.4: Comparison between pictures taken with Kinect v2 and v1.

9. Increasing the size dataset: More the data, less is the error. Taking 40 pictures per person instead of 20 may increase accuracy and address some of the above-mentioned issues like wrong recognition, etc.

10. Offloading Pi: Using Pi as a database and running all processes in a computer for fast processing.
REFERENCES


9) OpenCV: Face Recognition with OpenCV. (n.d.). Retrieved February 8, 2018, from https://docs.opencv.org/3.3.1/d8/d60/tutorial_face_main.html


21) R. Leinhart and J. Maydt, “An extended set of haar-like features for rapid object

22) Fisherfaces. (n.d.). Retrieved April 2, 2018, from
https://www.bytefish.de/blog/fisherfaces/#algorithmic-description

23) Algorithms: Why is viola-jones still considered as state-of- art in object detection