Python-based Raspberry Pi for Hand Gesture Recognition

Ali A. Abed
(SMIEEE, MACM)
Computer Engineering Department
Basra University
Basra, Iraq

Sarah A. Rahman
Computer Engineering Department
Basra University
Basra, Iraq

ABSTRACT

In this paper, a real time vision based system is proposed to monitor objects (hand fingers). It is built based on the Raspberry Pi with camera module and programmed with Python programming Language supported by Open Source Computer Vision (OpenCV) library. It also contains a 5 inch 800*480 Resistive HDMI Touch screen for I/O data. The Raspberry Pi embeds with an image-processing algorithm called hand gesture, which monitors an object (hand fingers) with its extracted features. The essential aim of hand gesture recognition system is to establish a communication between human and computerized systems for the sake of control. The recognized gestures are used to control the motion of a mobile robot in real-time. The mobile robot is built and tested to prove the effectiveness of the proposed algorithm. The robot motion and navigation satisfied with different directions: Forward, Backward, Right, Left and Stop. The recognition rate of the robotic system reached about 98.

Keywords

Raspberry Pi; Mobile Robot; Hand Gesture; Feature Extraction, Python, OpenCV.

1. INTRODUCTION

Vision-based and image processing systems has various applications in pattern recognition and moving robots navigation. It is a processing of input images producing output that is features or parameters related to images [1]. Its application in robotics, surveillance, monitoring, tracking, and security systems makes it important and cover a wide range of applications worldwide. Object tracking is the main activity in computer vision and extracting its features is the basic principle. It has many applications in traffic control, human computer interaction, gesture recognition, augmented reality and surveillance [2]. An efficient tracking algorithm will lead to the best performance of higher-level image tasks. Persons over the universe has been used a monitoring system for assisting them in securing territories or specific areas [3]. It led to a system that has the ability of surveillance and applications in detecting and monitoring a known object. Raspberry pi is a small sized PC board [4] suitable for real-time projects. The main purpose of the work presented in this paper is to make a system capable of detecting and monitoring some features for objects that specified according to an image processing algorithms using Raspberry Pi and camera module. The feature extraction algorithm programmed with Python supported by OpenCV libraries, and executed with the Raspberry Pi attached with an external camera. This system is working well even in poor illumination conditions. Hand gesture algorithm that embeds in the Raspberry Pi is used to steer a mobile robot implemented to get a vision-based robotic system that depends on human machine interaction.

Many types of object recognition and monitoring algorithms proposed over the years. To overcome their limitations, the researchers have always improved algorithms. These algorithms applied to give the best results in object recognition and monitoring. In addition, several researches studied Raspberry Pi applications, but less of them use it for building object recognizers.

Ron Oommen Thomas [5] adopted the Raspberry Pi for the control of Robot Arm from a remote end. In addition, it used Python language to write the forward and backward motion. Tohari Ahmad et al [3] proposed a comparatively inexpensive monitoring system that is capable of discovering an object, as well as calculating its distance locating its coordinate and taking its picture. The user receives those data via email. Alaa [6] proposed a method for path planning of mobile robot from start point to the goal point while avoiding obstacles on robot's path using artificial potential field (APF) algorithm. R Dharmateja [7] showed an attempt to building a low cost stand-alone device called Pi-pad, which is helpful for educational purpose using the Raspberry Pi as its brain with Bluetooth for connecting peripherals and communicating with local devices like Wi-Fi, keyboard and mouse. Hamid A. Jalab [8] proposed an algorithm that recognized a set of four specific hand gestures: Play, Stop, Forward, and Reverse. Chuan Zhao [9] proposed a method for tracking objects with their size and shape which changing with time, based on a group of mean-shift and affine structure. The results showed the object has tracking capability in the existence of wide change and partial blockage. G.Senthilkumar et al. [10] proposed a technique of image capturing in an embedded system based on Raspberry Pi board. The results showed that the designed system is fast enough to run the image capturing, recognition algorithm, and the data stream can flow smoothly between the camera and the Raspberry Pi board, but there may be some problems with accuracy. Aleksei Tepljakov [11] find out a various methods for serving computers interpret the real world visually, as well as provide solutions to those methods offered by OpenCV, and implemented some of these in a Raspberry Pi based application for detecting and tracking of objects. Some of the useful information also transmits by the application, such as coordinates and size, to other computers on the network that send an appropriate query. It may not suitable for real-time applications since there may be a delay. Mirjana Maksimović et al [12] surveyed, defined and presented the abilities of using Raspberry Pi as well as the advantages and disadvantages of its usage in the development of the next generation of Internet of Things (IoT).

In this paper, it is introduced mobile robot using Raspberry Pi, where its movement is controlled via the camera connected with Raspberry Pi that forward commands directly to the driver of a two-wheel drive mobile rover. It used hand gesture algorithm to identify the object (hand) and control the movement of robot. In addition, it made this robot works with living situation poor illumination environment.

2. SYSTEM ARCHITECTURE

2.1 Frames Capture

The input data can be a frame or a sequence of video frames, taken by a Raspberry Pi camera module pointed toward user's hand. A 5MP camera module that capable of 1080p video and still image but also 720p60 and 640x480p60/90 captures the frame. The sensor of camera has 5-megapixel native resolution in still capture mode. In video mode, it supports capture resolutions up to 1080p at 30 frames per second. The Pi's camera module is capable of 80fps in later firmware. The camera module is lightweight and small making it an ideal choice for mobile projects. The Raspberry Pi could do 90 frames/second (fps) for high-speed photography using Raspberry Pi camera module. Some advanced techniques for Raspberry Pi Camera board:

- Uuencoded image capture (RGB format)
- Rapid capture and processing
- Rapid capture and streaming
- Capturing images whilst recording
- Recording at multiple resolutions
- Recording motion vector data

The frames captured with simple background and stable light. Region of Interest (ROI) is the hand region, so it captured the images of the hand and converts them to gray scale in order to find the ROI i.e. the portion of the image that is further interested for image processing as shown in "Figure 1".

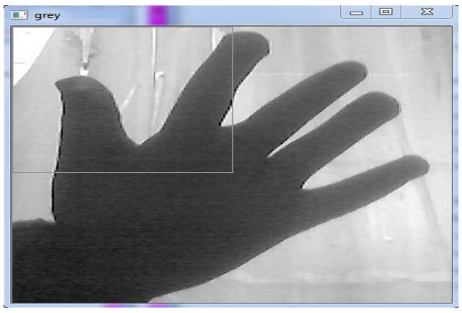

Fig 1: Gray scale Frame

2.2 Blur Frame

In image processing, a Gaussian blur (also known as Gaussian smoothing) is the result of blurring an image by a Gaussian function. It is a widely used effect in graphics software, typically to reduce image noise and reduce detail. The processing of blur frame shown in "Figure 2" starts with noise reduction using Gaussian Blurring on the original frame. Blur frame is necessary process for frame enhancement and for getting good results. Blurring is used for smoothing frames and reducing noise and details from the frame. With blurring, smooth transformation from one color to another and reduction of the edge contents are satisfied.

Mathematically, applying a Gaussian blur to an image is the same as convolving the image with a Gaussian function. This is also known as a two-dimensional Weierstrass transform. By contrast, convolving by a circle (i.e., a circular box blur) would more accurately reproduce the bokeh effect. Since the Fourier transform of a Gaussian is another Gaussian, applying a Gaussian blur has the effect of reducing the image's high-frequency components; a Gaussian blur is thus a low pass filter. The Gaussian blur is a type of image blurring filters that uses a Gaussian function (which also expresses the normal distribution in statistics) for calculating the transformation to apply to each pixel in the image. The equation of a Gaussian function in one dimension is:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

In two dimensions, it is the product of two such Gaussians, one in each dimension:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution. When applied in two dimensions, this formula produces a surface whose contours are concentric circles with a Gaussian distribution from the center point. Values from this distribution are used to build a convolution matrix, which is applied to the original image. Each pixel's new value is set to a weighted average of that pixel's neighborhood. The original pixel's value receives the heaviest weight (having the highest Gaussian value) and neighboring pixels receive smaller weights as their distance to the original pixel increases.

In addition, threshold process for frame segmentation is used to create binary images from gray scale images. It is not interested in the details of the image but in the shape of the object to track that by hand gesture.

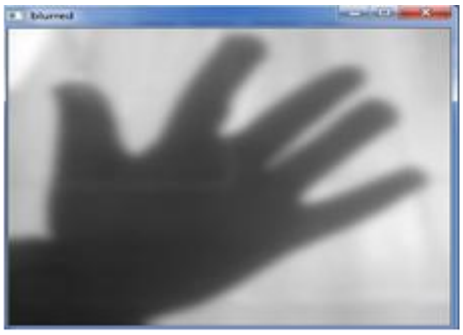

Fig 2: Blur Frame

2.3 Frame Segmentation

The task of segmenting and grouping pixels that are tracked is simplified by the high quality of the footage captured for most stop motion animations. Additionally, scenes shot with a moving camera tend to be the exception, so background-subtraction is a natural choice for segmenting the action. If a clean background plate is not available, median filtering in the time domain can usually generate one. It is observed a pixel location over the entire sequence, sorting the intensity values (as many as there are frames).

By choosing the median, the background can be reconstituted one pixel at a time. This highly parallelizable process results in choosing the colors, which were most frequently sampled by a given pixel, or at least colors that were closest to doing so. Given a reasonably good image of the background, I_b the pixels that are different in a given frame I_f are isolated.

An image I_m containing only pixels that is moving is obtained according to this criterion [11]:

$$I_m(x, y) = \begin{cases} I_f\ (x, y) & |I_f\ (x, y) - I_b(x, y)| > \text{threshold} \\ 0 & \text{otherwise} \end{cases}$$

Frame segmentation is the first step for any frame recognition process. The main purpose of hand segmentation is to separate the user's hand from the background in the frame [13]. In order to achieve this, different image segmentation algorithms has been used such as thresholding process, which leads to the result shown in "Figure 3".

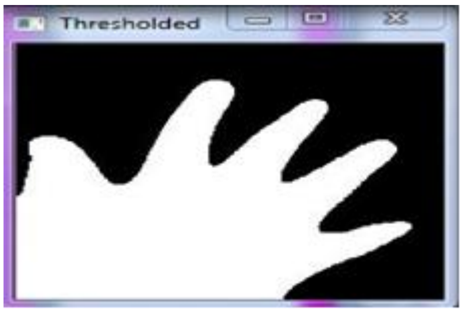

Fig 3: Thresholding process

2.4 Draw Contours

The basic idea in active contour models or snakes is to evolve a curve, subject to constraints from a given image in order to detect objects in that image. For instance, starting with a curve around the object to be detected, the curve moves toward its interior normal and has to stop on the boundary of the object.

Drawing contours in frames (as shown in "Figure 4") is the main problem in computer vision tasks. Contours featured from edges as follows [14]. Scan the frame from left to right and from top to bottom to find first contour pixel marked; then scan frame clockwise until the next pixel value is equal to 1.

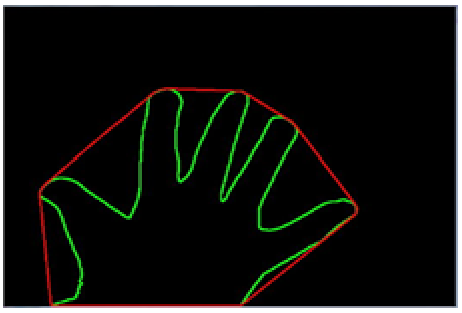

Fig 4: Contour Detection

2.5 Find Convex Hull and Convexity Defects

In this paper, the convex points are considered to be the tips of the fingers. Therefore, it founds convexity defects, which is the deepest point of deviation on the contour [15]. By this, it can find the number of fingers extended and then it can perform different functions according to the number of fingers extended. Convex Hull used to find the fingertips, is the convex set enclosing the hand region. The green line shown in "Figure 5" bounding the hand is a convex hull. The red dots are the defect points appeared in every valley. Depending upon the number of defect points, the number of fingers unfolded is calculated.

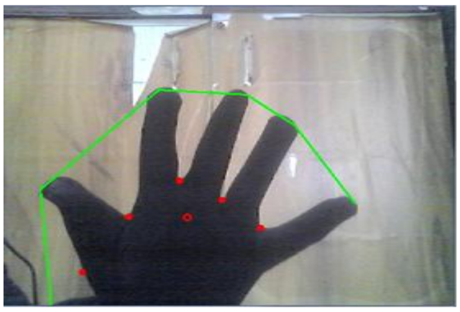

Fig 5: Convex Hull and Convexity Defects

3. SOFTWARE REQUIRED

Programs needed for the system are summarized as follows:

3.1 Raspbian OS (Operating System)

An operating system has been developed for Raspberry Pi. This system has over than 35000 packages, which are all set to backup Raspberry pi environment. It is free and can be downloaded from internet (known as NOOPS) and then copied into a 16GB (or more) RAM stick.

3.2 Python & OpenCV

Python is a well-known, high-level programming language used for general-purpose programming, created in 1991. It allows programmers to express concepts in fewer lines of code than possible in languages such as C or Java. Python features a dynamic system and automatic memory management and supports multi-programming style, including object-oriented, functional programming, and also procedural styles. Besides that, it has large and comprehensive standard libraries. Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems.

OpenCV is a free library includes hundreds of APIs for computer vision used in image processing to optimize a real time application. There are some features in OpenCV which support data processing, including: object detection, camera calibration, 3D reconstruction and interface to video processing. The primary interface of OpenCV written in C++ but it supports other interfaces also such as C, Python, Java and MATLAB. In this paper, python language is used as the programming language with the required libraries from OpenCV to build the hand gesture recognition system that controls the motion of a mobile robot.

3.3 System Implementation and Algorithm

- Import the necessary packages: define the necessary packages that needed in this algorithm such as:

```
import cv 2
import numpy as np
import time
from picamera.array import PiRGBArray
from picamera import PiCamera
import math
import string
import RPi.GPIO as GPIO
```

- Define the pins of GPIO of Raspberry Pi 3 to connect it with robot driver.

- Initialize the current frame of the video and define global variables as follows:

```
camera = PiCamera()

frame = None

camera.resolution = (640,480)

camera.framerate = 32
rawCapture = PiRGBArray(camera, size = (640, 480))
```

- Capturing is doing as follow:

```
for frame in camera.capture_continuous(rawCapture,

 format="bgr", use_video_port=True):

   # grab the raw NumPy array representing the image,

  # then initialize the timestamp

   # and occupied/unoccupied text
  image = frame.array
```

- Convert frame to grey-scale, after that Blur and threshold it as follows:

```
grey = cv2.cvtColor(crop_image, cv2.COLOR_BGR2GRAY)

  value = (35, 35)

  blurred = cv2.GaussianBlur(grey, value, 0)

  _, thresh1 = cv2.threshold(blurred,

                          127, 255, cv2.THRESH_BINARY_INV +

                        cv2.THRESH_OTSU)
```

- Then find out the contours as follows:

```
image, contours, hierarchy = cv2.findContours(thresh1.copy(), \
        cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
```

- Find Convex Hull and points of defects as follows:

```
cnt = max(contours, key = lambda x : cv2.contourArea(x))

  x,y,w,h = cv2.boundingRect(cnt)

  cv2.rectangle(crop_image,(x,y),(x+w,y+h),(0,0,255),0)

  hull = cv2.convexHull(cnt)

  drawing = np.zeros(crop_image.shape,np.uint8)

  cv2.drawContours(drawing,[cnt],0,(0,255,0),0)

  cv2.drawContours(drawing,[hull],0,(0,0,255),0)

  hull = cv2.convexHull(cnt,returnPoints = False)

  defects = cv2.convexityDefects(cnt,hull)

  count_defects = 0
  cv2.drawContours(thresh1, contours, -1, (0, 255, 0), 3)
```

- Draw Contours as the following piece of code:

```
for i in range(defects.shape[0]) :

   s,e,f,d = defects[i,0]

   start = tuple(cnt[s][0])

   end = tuple(cnt[e][0])

   far = tuple(cnt[f][0])

   a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)

   b = math.sqrt((far[0] - start[0])**2 + (far[1] - start[1])**2)

   c = math.sqrt((end[0] - far[0])**2 + (end[1] - far[1])**2)
   angle = math.acos((b**2 + c**2 - a**2) / (2*b*c)) * 57
```

4. HARDWARE DESCRIPTION

The hardware components of the system consists of Raspberry Pi 3 model B, Camera Board, 5 inch 800*480 Resistive HD Touch Screen, L298 H-bridge driver, Rover 5 two-wheel drive platform, and rechargeable batteries as a power supply.

4.1 The platform

The robot considered for practical test is a low-cost (about 65$), differential drive robot with two motors as shown in "Figure 6". It is a high torque tank suitable for on-road and off-road environments. Its voltage of operation is 9VDC.

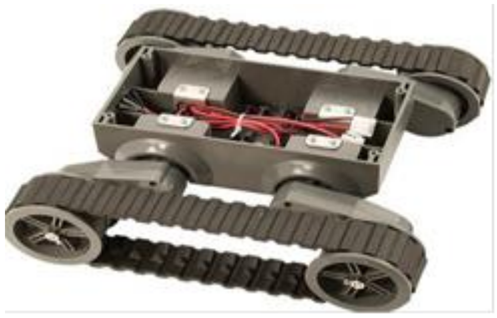

Fig 6: The Robot platform

4.2 Motor driver (H-bridge)

Motor driver is available as an integrated board, which is a high voltage, high current dual full-bridge L298 driver designed to accept standard TTL logic levels and drives inductive loads such as relays, solenoids, DC and stepping motors. It is a low-cost (5$), small size, and very light-weight. In this paper, it is used to drive the two DC motors of the Rover controlling the speed and direction of each one independently. Its real image is shown in "Figure 7".

4.3 Power Supply

There are several ways for powering hardware components (Raspberry Pi, driver, LCD, and the two DC motors of robot) such as: smart power and rechargeable battery as shown in "Figure 8". Both of these power supplies are used to provide sufficient power for the overall system.

4.4 Raspberry Pi 3 Model B

The Raspberry Pi 3 shown in "Figure 9" is the third generation of Raspberry Pi appeared in February 2016. The most important specifications are: A 1.2GHz 64-bit quad-core ARMv8 CPU, 1GB RAM, 40 GPIO pins, Full HDMI port, Micro SD card slot, etc. The rest of its strong specifications are available at [16]. It has the following advantages to be used for the proposed work of this paper: small credit card sized suitable for embedded systems with the full capability of a PC computer, open source Linux-based operating system, low cost (about 40$), low power (5VDC power supply), high processing speed (1.2 GHZ, quad-core), simple 40 I/O pins for external interface, high performance operation with Python and OpenCV, and provided with all the required accessories needed for computer vision.

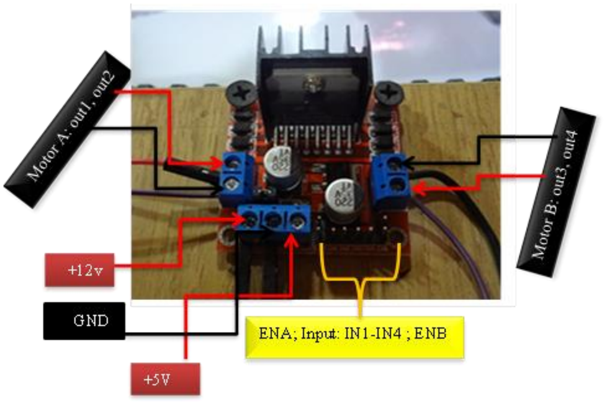

Fig 7: L298N motor driver board

Fig 8: Power Supplies

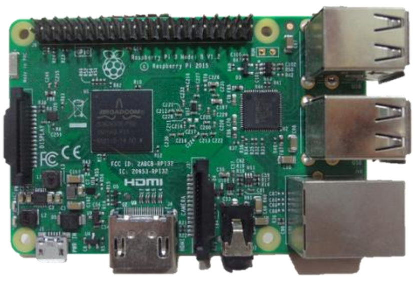

Fig 9: Raspberry Pi 3 Model B

4.5 Camera Model

The camera module for Raspberry Pi is shown in "Figure 10". It is used for taking high-video clarity, in addition to stills photographs. It is provided with a ribbon cable that plugs in easily to the Raspberry Pi board with simple settings. It is low-cost (15$), small size, and light-weight (20g). It designed to contact to Camera Serial Interface (CSI) of the Raspberry Pi. It fits with all models of Raspberry Pi. It is used efficiently with our vision-based robotic system.

4.6 Touch Screen HDMI interface

The touch screen adopted in this work is a 5 inch 800*480 Resistive touch LCD ("Figure 11") compatible with Raspberry Pi 2 and 3. It has small-size and light-weight (about 100g) suitable for embedded system design. It is plugged in with the Raspberry Pi as a one embedded unit. All the required software, settings, and inputs are provided to the Raspberry Pi through this very slim LCD. It is a low-cost (about 50$) and low-power (5VDC power).

Fig 10: Raspberry Pi Camera Modules

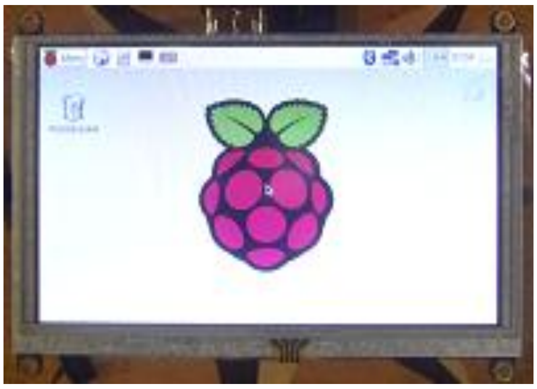

Fig 11: Five-inch Touch Screen HDMI interface

5. RESULTS and DISCUSSION

In order to evaluate the performance of the presented gesture recognizer, recognized samples are separated into training and testing sets. The recognition rate (RR) of each finger is calculated by the following equation [17]:

$$RR = \frac{No.of\ correctly\ recognized\ gestures}{No.of\ tested\ gestures} \times 100\%$$

Each gesture of the five direction gestures is trained fifteen times to provide sufficient number of template vectors for four different cases of experimental work. Hence, the number of stored templates is 75. In the test phase, each of the five gestures is tested at run time of the mobile robot that operates 30 times. Mobile robot motion with respect to the corresponding finger command is monitored and recognition accuracy is computed. Table 1 shows the RR for a sample of tested gestures in the database which have been already done for four different cases use 3, 5, 10, or 15 template vectors for each gesture. From Table 1, three template feature vectors per target gesture give poor recognition rate. RR is significantly increased with 5 template vectors. Using 10 template vectors give very good RR. Increasing the number of template vectors to 15 led to little increasing in the recognition rate. It is possible with a series of gesture (fingers) commands to steer the robot along a desired trajectory to its target. Also, controlling the number of fingers in front of camera leads to control motion and avoid obstacles in the way of the robot. A recognition rate of about 98% is reached.

Table 1. RR for various gesture sets

No. of fingers	Motion	RR			
		3	5	10	15
1	Forward	47	66	98	97
2	Backward	44	66	90	88
3	Right	45	70	90	89
4	Left	50	82	95	92
5	Stop	48	74	89	88

The schematic diagram of the hardware components constituting the robotic system is shown in "Figure 12". To control the two DC motors of mobile robot, first it connects each motor to A (Out 1 & Out 2) and B (Out 3 & Out 4) connections on the L298N module. Next, it is powered by 12V battery. The Raspberry Pi is powered from the smart supply with 5V. Six GPIO pins are needed on Raspberry Pi, GPIO10 to enable motor A, GPIO09 to enable motor B, and the input pins (IN1, IN2, IN3 and IN4) of L298N driver are connected with (GPIO18,GPIO15, GPIO14 and GPIO04) of Raspberry Pi respectively. The direction of motors is controlled by the hand gesture recognition system through the distinguishing of each finger separately as shown in "Figure 13". The final setup of the whole system is displayed in "Figure 14".

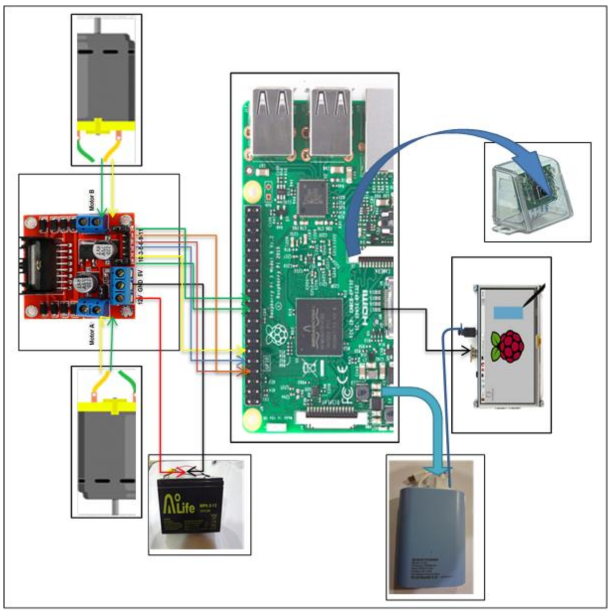

Fig 12: Schematic diagram for the whole system

6. CONCLUSIONS

The performance of the presented algorithm is evaluated based on the recognition of hand gestures. The hand gesture algorithm did not used previously with Raspberry Pi for recognition and robot motion control. The database which used for human hand gesture recognition is supported with five types of gestures for five movements controlled with hand. The experimental results showed that the designed system can be used for tracking and has a robust recognition level in detecting and recognition of hand of human by a low-cost computer interaction technique. The real-time vision-based system is implemented efficiently with Python programming language, OpenCV libraries, Raspberry Pi computer device, camera module, and Linux-based LCD touch screen besides the robotic car with its H-bridge driver. The implemented embedded system proved accurate and easy implementation for mobile robot navigation and direction control. It introduced a new and modern method for hand gesture recognition that does not depend on classical sensors (flex, IR, or Ultrasonic). Hand gesture commands obtained from the proposed algorithm routed to the GPIO pins of the Raspberry Pi in order to use it for driving two wheels robotic car in four directions: Forward, Backward, Left, Right movements in addition to Stop. The RR of the designed system is reached to 98%. The overall system costs about 200$ and proved a good operation when navigated in a clear environment.

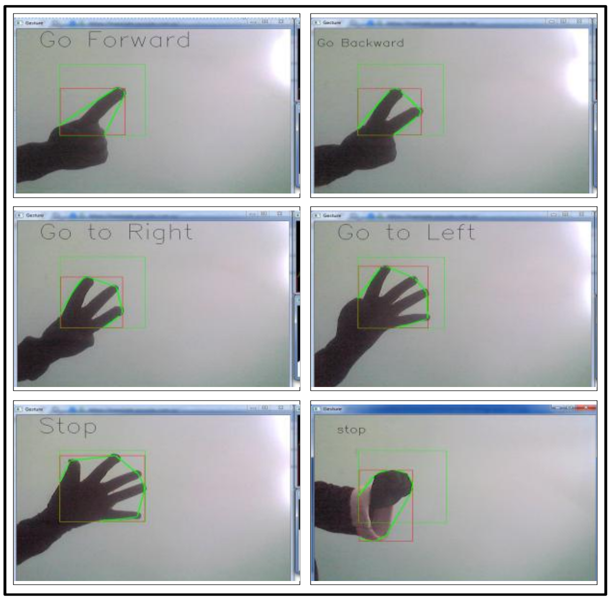

Fig 13: Samples of hand gestures

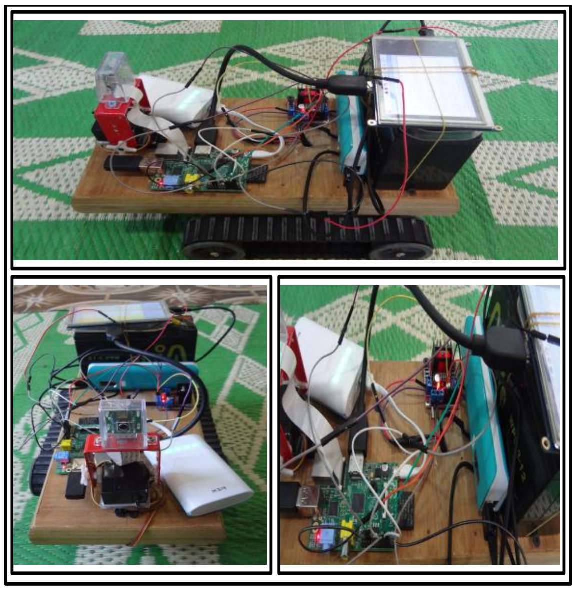

Fig 14: Overall system setup

7. ACKNOWLEDGMENTS

First praise is to Allah, for his uncountable blessing, helping and supporting all the time. No one can thanks God, as he deserves because of his limitless blessing on us. I would like to express my deep sense of gratitude to my supervisors Dr. Ali A. Abed for their valuable guidance and encouragement during the development of this work. Finally, my special thanks and appreciation go to my family especially my husband for their help.

8. REFERENCES

[1] Ali A. Abed, Sarah A. Rahman, October 11, 12 2016. Computer Vision for Object Recognition and Tracking Based on Raspberry Pi. International Conference on Change, Innovation, Informatics and Disruptive Technology ICCIIDT’16, London- U.K.

[2] John G. Allen, Richard Y. D. Xu, Jesse S. Jin, 2004. Object tracking using camshift algorithm and multiple quantized feature spaces. Proceedings of the Pan-Sydney area workshop on Visual information processing. Australian Computer Society, Inc., Vol. 36, pp. 3-7.

[3] Ahmad, Tohari, Hudan Studiawan, and T. Ramadhan. 2014. Developing a Raspberry Pi-based Monitoring System for Detecting and Securing an Object. International Electronics Symposium (IES), pp. 125-129.

[4] Hemalatha, P., CK Hemantha Lakshmi, and S. A. K. Jilani, August 2015. Real time Image Processing based Robotic Arm Control Standalone System using Raspberry Pi. SSRG International Journal of Electronics and Communication Engineering (SSRG-IJECE), Volume 2, Issue 8, pp. 18-21.

[5] Thomas, Ron Oommen, and K. Rajasekaran, April 2014. Remote control of robotic arm using raspberry pi. International Journal of Emerging Technology in Computer Science &Electronics (IJETCSE), Volume 8, Issue1, pp.186-189.

[6] Ahmed, Alaa A., Turki Y. Abdalla, and Ali A. Abed. March 2015. Path planning of mobile robot by using modified optimized potential field method. International Journal of Computer Applications 113.4, Volume 113 – No. 4.

[7] Dharmateja, R., and A. Ruhan Bevi, February 2015. Raspberry Pi Touchscreen Tablet (Pi-Pad). International Journal on Recent and Innovation Trends in Computing and Communication, Volume: 3, Issue: 2, pp.600 – 605.

[8] Jalab, Hamid A., and Herman K. Omer. 2015. Human computer interface using hand gesture recognition based on neural network. Information Technology: Towards New Smart World (NSITNSW), 2015 5th National Symposium on. IEEE, pp.1-6.

[9] Zhao, Chuan, Andrew Knight, and Ian Reid., 2008. Target tracking using mean-shift and affine structure. Pattern Recognition, 2008. ICPR 2008. 19th International Conference on. IEEE, pp. 1-5.

[10] Senthilkumar, G., K. Gopalakrishnan, and V. Sathish Kuma, March – April 2014. Embedded image capturing system using raspberry pi system. International Journal of Emerging Trends & Technology in Computer Science 3.2, Volume 3, Issue 2, pp.213-215.

[11] Tepljakov, Aleksei, 2015. Raspberry Pi based System for Visual Object Detection and Tracking. Bachelor’s Thesis.

[12] Maksimović, M., Vujović, V., Davidović, N., Milošević, V., & Perišić, B, JUNE 2014. Raspberry Pi as Internet of things hardware: performances and constraints. design issues 3.

[13] Ohn-Bar, Eshed, and Mohan Manubhai Trivedi. DECEMBER 2014. Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations. IEEE Transactions on Intelligent Transportation Systems 15.6, Vol. 15, No. 6, pp.2368-2377.

[14] Chan, Tony F., and Luminita A. Vese, FEBRUARY 2001. Active contours without edges. IEEE Transactions on image processing 10.2, Vol. 10, No. 2, pp.266-277.

[15] Panwar, Meenakshi. 2012. Hand gesture recognition based on shape parameters. 2012 International Conference on Computing, Communication and Applications. IEEE, pp.1-6.

[16] Raspberry Pi 3 Model B datasheet, RS Company, www.rs-components.com/raspberrypi.

[17] Ali A. Abed and Abbas A. J. 2016. Design and implementation of wireless voice controlled mobile robot. Al-Qadisiyah Journal for Engineering Science, Vol.9, No.2.