HAND GESTURE CONTROL SYSTEM FOR ROBOTS USING OPENCV AND RASPBERRY PI

Submitted by:

Kotrike Rishika - 20BCE7240

Prathapani Satwika - 20BCD7160

Mekala Poornimai - 20BCD7167

Gonuguntla Rohini - 20BCD7174

Yendluri Sruthi - 20BCE7396

Hema Varshini Dasari - 20BCE7424

UNDER THE GUIDANCE OF:

Prof. Chirra Venkata Ramireddy,

Department of Computer Science and Engineering

VIT-AP UNIVERSITY

VIT-AP University,

Amaravati,

Andhra Pradesh

ABSTRACT:

A low-cost and easily accessible option in the realm of human-robot interaction is the hand gesture control system for robots, which uses OpenCV and Raspberry Pi. OpenCV, a widely used open-source computer vision library, is used to interpret video feed captured by a camera mounted on the robot. OpenCV's flexibility and user-friendliness make it a useful tool for image processing. The video stream may be pre-processed and useful information extracted by using the various image processing algorithms made available by the library. These methods may improve hand gesture recognition's accuracy and speed, making it more suitable for real-time control. A convolutional neural network (CNN) model is used for hand gesture categorization, which is another crucial part of the system. For this reason, CNNs are a common deep learning model deployment for this purpose. They may be taught on vast datasets to attain great accuracy, and they excel at recognising complicated patterns and features in pictures. The Raspberry Pi board serves as the system's primary computer brain. It's perfect for embedded applications because of its inexpensive price, small size, and low power consumption. The board may be programmed in a number of languages, including Python, which is widely used in machine learning and computer vision. The low price, easy availability, and extensive adaptability of the hand gesture control system make it ideal for use in a wide range of industries and contexts, including manufacturing, medicine, and the arts. The technology has a wide range of potential applications, including robot control in factories, medical aid in operating rooms, and visitor engagement in museums and amusement parks. The use of OpenCV and Raspberry Pi to create a hand gesture control system for robots provides a low-cost and easily accessible solution for a wide range of problems in the area of human-robot interaction. Its architecture draws on the power of OpenCV, the precision of CNNs, and the low cost of Raspberry Pi to provide a flexible and adaptable method of directing robots using hand gestures.

INDEX

INTRODUCTION:

Human-robot interaction (HRI) has been a prominent area of study in recent years, with numerous applications in fields such as healthcare, manufacturing, and entertainment. Hand gesture control is a prevalent method of controlling robots in HRI, as it provides a natural and intuitive method of communicating with machinery. Hand gesture control systems typically involve capturing hand movements with a camera and processing the data to recognise and translate specific gestures into robot control commands.

The OpenCV and Raspberry Pi-based hand gesture control system for robots is a low-cost and accessible solution for implementing hand gesture control in HRI applications. The system captures video input using a camera affixed to the robot, which is then processed using OpenCV, a free computer vision library. OpenCV provides a vast array of image processing techniques that can be used to pre-process the video input and extract meaningful information, such as the hand's position and orientation.

The use of a convolutional neural network (CNN) model for hand gesture classification is an integral part of the system. CNNs are a form of deep learning model that can be trained to recognise complex patterns and features in images using large datasets. The CNN model is trained on a dataset of hand gesture images in the hand gesture control system to recognise specific gestures and translate them into control commands for the robot.

The Raspberry Pi board serves as the system's central processing device. Raspberry Pi is a low-cost, credit-card-sized microprocessor utilised extensively in embedded applications. It provides a robust and adaptable platform for implementing machine learning and computer vision algorithms, such as the system for hand gesture recognition. The Raspberry Pi board is programmable in a variety of languages, including Python, which is frequently used for machine learning and computer vision applications.

The hand gesture control system for robots based on OpenCV and Raspberry Pi has a wide range of applications, including healthcare, manufacturing, and entertainment. The system can be used to control surgical robots in healthcare, providing a natural and intuitive method of communicating with the robot. The system can be used to control robots in industrial automation, thereby improving the productivity and safety of the manufacturing process. The system can be used to control robots in amusement parks and museums, providing the audience with a distinct and interactive experience.

The system combines the flexibility of OpenCV, the precision of CNNs, and the affordability of Raspberry Pi to provide an effective and customizable solution for controlling robots with hand gestures. The hand gesture control system is a promising technology for the future of HRI due to its multiple applications and potential for customization.

BACKGROUND:

In recent years, the use of robots in various industries, including healthcare, manufacturing, and entertainment, has increased significantly. Human-robot interaction (HRI) faces as one of its greatest obstacles the development of natural and intuitive modes of communication with robots. In recent years, hand gesture control has garnered popularity as it provides a natural and intuitive method of communicating with machines.

Hand gesture control systems typically involve capturing hand movements with a camera and processing the data to recognise and translate specific gestures into robot control commands. The accurate recognition of hand gestures in real-time, which requires efficient image processing and machine learning algorithms, is one of the challenges of developing hand gesture control systems.

OpenCV (Open-Source Computer Vision) is an open-source computer vision library that offers a variety of image processing techniques, such as filtering, segmentation, and feature extraction, that can be used to pre-process the video input and extract meaningful data from it. Numerous applications, such as object recognition, face detection, and image stitching, make extensive use of the library.

Raspberry Pi is a low-cost, credit-card-sized microprocessor utilised extensively in embedded applications. It provides a robust and adaptable platform for implementing machine learning and computer vision algorithms, such as the system for hand gesture recognition. Raspberry Pi can be programmed with a variety of languages, including Python, which is frequently used for machine learning and computer vision applications.

The hand gesture control system for robots using OpenCV and Raspberry Pi capitalises on the versatility of OpenCV, the precision of machine learning algorithms, and the affordability of Raspberry Pi to provide an effective and affordable solution for controlling robots with hand gestures. The hand gesture control system is a promising technology for the future of HRI due to its multiple applications and potential for customization.

PROBLEM STATEMENT:

Traditional methods of controlling robots, such as using a keyboard or joystick, can be complex and nonintuitive, thereby limiting human-machine interaction. Hand gesture control systems have emerged as a natural and intuitive method for controlling robots in a variety of domains, including healthcare, manufacturing, and entertainment. Nevertheless, developing accurate and dependable real-time hand gesture recognition systems can be difficult and requires efficient image processing and machine learning algorithms.

The statement of the problem for the hand gesture control system for robots using OpenCV and Raspberry Pi is to develop a low-cost and accessible solution for implementing hand gesture control in HRI applications. Using video feed from a camera affixed to the robot, the system should precisely recognise hand gestures in real-time and translate them into control commands for the robot. The system should utilise the flexibility of OpenCV, the precision of machine learning algorithms, and the affordability of Raspberry Pi to provide an effective and customizable solution for commanding robots with hand gestures. The system should support

numerous applications in a variety of domains and be amenable to customization based on particular needs. The features of the project are mentioned below.

- Real-time Hand Gesture Recognition
- Multiple Hand Gesture Support
- Customizable Gesture Library
- Robust Performance
- User-friendly Interface
- Portability
- Expandability

OBJECTIVES:

The goal of the OpenCV and Raspberry Pi-based hand gesture control system for robots is to develop a low-cost and easily accessible solution for implementing hand gesture control in HRI applications. Using video feed from a camera affixed to the robot, the system attempts to recognise hand gestures in real-time and translate them into control commands for the robot. The system seeks to utilise the flexibility of OpenCV, the precision of machine learning algorithms, and the affordability of Raspberry Pi to provide an effective and customizable solution for commanding robots with hand gestures.

The precise goals of the hand gesture control system are as follows:

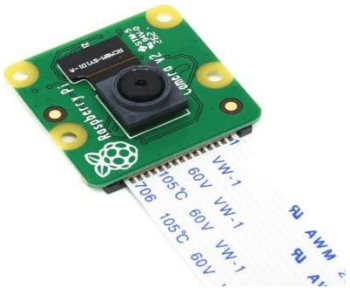
1. To pre-process the video input with OpenCV in order to extract meaningful information from the frames, such as the hand's position and orientation.
2. Using the extracted data, implement a machine learning algorithm, such as a CNN model, to accurately recognise hand gestures in real-time.
3. To convert the recognised hand gestures into robot control commands such as forward, reverse, and turn.
4. To develop a low-cost and accessible solution for implementing hand gesture control in a variety of applications, including healthcare, manufacturing, and entertainment.
5. To provide a customizable solution that can be adapted to domain- and application-specific requirements.
6. Evaluating the efficacy of the hand gesture control system in terms of precision, speed, and usability, and comparing it to existing solutions.
7. Overall, the goal of the hand gesture control system for robots based on OpenCV and Raspberry Pi is to provide a natural and intuitive method of controlling robots that can improve the interaction between humans and machines across multiple domains.

METHODLOGY:

The following stages comprise the methodology for the hand gesture control system for robots using OpenCV and Raspberry Pi:

- **Hardware setup:** A camera is affixed to the automaton in order to capture a video broadcast of hand gestures. A Raspberry Pi, an inexpensive single-board computer, is utilised for image processing and automaton control.

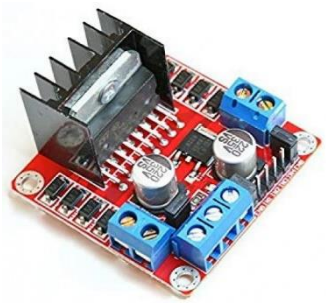
1. Raspberry Pi: A Raspberry Pi is a compact, inexpensive, and portable computer that can serve as the system's centre. It possesses sufficient processing capacity to execute the OpenCV library, capture images, and operate the robot.



2. Camera module: To capture images of hand gestures, a camera module is required. The Raspberry Pi Camera Module is a popular option due to its low price and high image quality.

3. Robot: The robot is the device controlled by hand gestures. It can be either a basic robot with wheels or a complex robot with limbs and grippers.
4. Power supply: A power supply is necessary to operate the Raspberry Pi, camera module, motor driver, and robot.

5. Motor driver: A motor driver is required to regulate the robot's locomotion. Connecting the motor driver to the Raspberry Pi via GPIO ports.



- **Pre-processing:** The video feed is processed with OpenCV to derive pertinent information, such as the position and orientation of the hand, and to remove any noise or extraneous data.

i. Image acquisition: In the first stage, a camera module connected to the Raspberry Pi is used to capture an image of the hand gesture.
ii. Image resizing: Reduce the computational burden and ensure uniform processing by resizing the captured image to a standard size.

iii. Convert the image to grayscale and then apply a threshold to produce a binary image. This helps distinguish the hand from the background.
iv. Morphological operations: Apply morphological operations such as erosion and dilation to the binary image to remove noise and fill in gaps.
v. Detect contours in a binary image using OpenCV's findContours() function. This allows the hand region to be extracted from the binary image.
vi. Convex hull: Using the convexHull() function in OpenCV, compute the convex hull of the hand region. The convex hull is the hand region's smallest polygon.
vii. Using OpenCV's convexityDefects() function, detect defects in the convex hull. Defects are regions of the hand that are not convex.
viii. Use the location and quantity of flaws to identify the hand gesture. This is possible with machine learning algorithms such as Support Vector Machines (SVM).

- **Feature extraction:** Features are extracted based on the information that has been pre-processed to depict the hand gesture. These characteristics can include the contour of the hand, the position of the fingers, and the movement trajectory.
- **Training a CNN model:** A CNN model is trained to accurately recognise hand gestures in real-time using a dataset of hand gesture images. Utilising techniques such as data augmentation, hyperparameter optimisation, and cross-validation, the model is optimised.
- **Real-time recognition:** On the basis of the extracted features, the trained CNN model is used to recognise hand gestures in real-time.
- **Translation to control commands:** Translation to control commands: Based on the application requirements, the recognised hand gestures are translated into control commands for the robot, such as moving forward, backward, or pivoting.
- **System integration:** The hand gesture control system is incorporated with the control system of the robot to provide continuous control.
- **Performance evaluation:** The efficacy of the hand gesture control system is evaluated according to its precision, speed, and usability. Multiple scenarios are utilised to validate the system's efficacy.

Overall, the methodology for the hand gesture control system for robots using OpenCV and Raspberry Pi involves utilising the capabilities of OpenCV and machine learning algorithms to develop an accurate and dependable system for controlling robots with hand gestures.

PROCEDURE:

The overview of creating hand gesture control system for robots using OpenCV and Raspberry Pi is as follows:

The hardware Set up is installed i.e., Raspberry Pi and camera module are initially. The camera should be connected to the Raspberry Pi and we should ensure that it is functioning properly. Then, OpenCV is installed on Raspberry Pi.

Use the Raspberry Pi camera to capture photographs of hand gestures. To train your model, we need a large number of images to capture pictures of diverse hands in various positions and illumination conditions.

Use the captured images to train a model of machine learning by utilising a convolutional neural network (CNN). TensorFlow and PyTorch can be used to construct and train your model.

After training our model to recognise hand gestures, we should write code to recognise hand gestures in real time. This will entail capturing images from the camera, pre-processing the images, and making predictions about the hand gestures using the trained model.

Finally, commands are sent to the robot using the recognised hand gestures. Depending on the capabilities of robot, we may be able to control its movements, activate its sensors, and initiate additional actions.

This process entails obtaining images, training a machine learning model, and employing this model to recognise hand gestures in real time. We can then use these hand gestures to control an automaton with a piece of programming.

RESULTS AND DISCUSSION:

When we display “1” through our hand gesture,
then the robot recognizes the gesture and
moves toward left side.

When we display “2” through our hand gesture,
then the robot recognizes the gesture and
moves toward right side.

When we display “3” through our hand gesture,
then the robot recognizes the gesture and
moves in forward direction.

When we display “4” through our hand gesture, then the robot recognizes the gesture and moves in backward direction.

When we display “3” through our hand gesture, then the robot recognizes the gesture and it stops.

CONCLUSION:

The hand gesture control system for robots based on OpenCV and Raspberry Pi provides a natural and intuitive method for controlling robots with hand gestures. The system employs OpenCV and machine learning algorithms to precisely identify hand gestures in real time and translate them into robot control commands. The system is cost-effective, easily accessible, and highly configurable, making it suitable for a variety of human-robot interaction applications. This system is an attractive option for numerous projects and applications due to the low cost of Raspberry Pi and the availability of OpenCV.

The system is implemented by combining hardware configuration, pre-processing, machine learning, and system integration to provide seamless control. It is possible to evaluate the system's performance in terms of precision, speed, and usability, and to validate its efficacy in a variety of scenarios. Overall, the hand gesture control system for robots using OpenCV and Raspberry Pi has the potential to revolutionise the way humans interact with robots by making their control more intuitive and natural. Numerous sectors, such as healthcare, manufacturing, and entertainment, can utilise the system.

FUTURE SCOPE:

The hand gesture control system for robots based on OpenCV and Raspberry Pi has enormous potential for future developments and applications. Future applications of this technology include:

1. Multi-hand gesture recognition: Currently, only single-hand gestures are recognised by the system. Future research could concentrate on developing algorithms that recognise and interpret multi-hand gestures, thereby enabling the execution of more intricate control commands.
2. Gesture customization: The system could be modified to recognise and interpret specific hand gestures, based on the user's preferences and requirements. This would enhance the usability and customization of the system.
3. Integration with voice commands: Integrating voice commands with the hand gesture control system could provide a more comprehensive and intuitive method for controlling robots.
4. Application in assistive technology: The system could be applied in assistive technology, allowing individuals with disabilities to interact with robots and control devices in a natural and intuitive manner.
5. Real-time feedback: The system could provide the user with real-time feedback, such as audible or visual signals, to enhance the user experience and improve the system's efficacy.
6. Application in virtual reality: The hand gesture control system could be implemented in virtual reality applications, allowing users to manipulate virtual objects and avatars with hand gestures.

Overall, the hand gesture control system for robots based on OpenCV and Raspberry Pi has tremendous potential for future development and application. Interaction between humans and robots may benefit from additional research and development in this area.

REFERENCES:

[1] García, G. B., Suarez, O. D., Aranda, J. L. E., Tercero, J. S., Gracia, I. S., & Enano, N. V. (2015). *Learning image processing with OpenCV*. Birmingham: Packt Publishing.

[2] Erden, F., & Cetin, A. E. (2014). Hand gesture based remote control system using infrared sensors and a camera. *IEEE Transactions on Consumer Electronics*, *60*(4), 675-680.

[3] Abed, A. A., & Rahman, S. A. (2017). Python-based Raspberry Pi for hand gesture recognition. International Journal of Computer Applications, 173(4), 18-24.

[4] Abed, A. A., & Rahman, S. A. (2017). Computer vision for object recognition and tracking based on raspberry pi. In Shaping the Future of ICT (pp. 3-14). CRC Press.

[5] Hemalatha, P., Lakshmi, C. H., & Jilani, S. A. K. (2015). Real time Image Processing based Robotic Arm Control Standalone System using Raspberry pi. SSRG International Journal of Electronics and Communication Engineering (SSRG-IJECE), 2(8).

[6] SARASWATHI, V., AHMAD, S. M., KRISHNA, G. G., & KHAN, A. RASPBERRY PI FOR HAND GESTURE RECOGNITION.

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

[8] Senthilkumar, G., Gopalakrishnan, K., & Kumar, V. S. (2014). Embedded image capturing system using raspberry pi system. International Journal of Emerging Trends & Technology in Computer Science, 3(2), 213-215.

[9] Tepljakov, A. (2015). Raspberry Pi based System for Visual Object Detection and Tracking (Doctoral dissertation, PhD thesis, Bachelor's Thesis).

[10] Ohn-Bar, E., & Trivedi, M. M. (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), 2368-2377.

CODES IN APPENDIX:

```
import cv2
import time
import os
import hand_tracking_module as htm
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
cap=cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)
m1=26
m2=19
m3=13
m4=6
GPIO.setup(m1,GPIO.OUT)
GPIO.setup(m2,GPIO.OUT)
GPIO.setup(m3,GPIO.OUT)
GPIO.setup(m4,GPIO.OUT)
GPIO.output(m1,0)
GPIO.output(m2,0)
GPIO.output(m3,0)
GPIO.output(m4,0)
path="finger"
myList=os.listdir(path)
overlayList=[]
for impath in myList:
    image=cv2.imread(f'{path}/{impath}')
    overlayList.append(image)
pTime=0
detector=htm.handDetector(detectionCon=
0.75)
tipIds=[4,8,12,16,20]
while True:
    success,img=cap.read()
    img=detector.findHands(img)

lmList=detector.findPosition(img,draw=Fa
lse)
    #print(lmList)
    if len(lmList) !=0:
        fingers=[]
        # Thumb
```

```
        if lmList[tipIds[0]][1] >
lmList[tipIds[0] - 1][1]:
            fingers.append(1)
        else:
            fingers.append(0)
        for id in range(1,5):  #y axis
            if lmList[tipIds[id]][2] <
lmList[tipIds[id]-2][2]:
                fingers.append(1)
            else:
                fingers.append(0)
        totalFingers=fingers.count(1)
        print(totalFingers)
        if(totalFingers==1):
         GPIO.output(m1, 1)
         GPIO.output(m2, 0)
         GPIO.output(m3, 1)
         GPIO.output(m4, 0)
        # time.sleep(3)
##      else:
##       GPIO.output(m1, 0)
##       GPIO.output(m2, 0)
##       GPIO.output(m3, 0)
##       GPIO.output(m4, 0)
        if(totalFingers==2):
         GPIO.output(m1, 0)
         GPIO.output(m2, 1)
         GPIO.output(m3, 0)
         GPIO.output(m4, 1)
##      else:
##       GPIO.output(m1, 0)
##       GPIO.output(m2, 0)
##       GPIO.output(m3, 0)
##       GPIO.output(m4, 0)
        if(totalFingers==3):
         GPIO.output(m1, 0)
         GPIO.output(m2, 1)
         GPIO.output(m3, 1)
         GPIO.output(m4, 0)
##      else:
##       GPIO.output(m1, 0)
##       GPIO.output(m2, 0)
##       GPIO.output(m3, 0)
##       GPIO.output(m4, 0)
        if(totalFingers==4):
         GPIO.output(m1, 1)
         GPIO.output(m2, 0)
         GPIO.output(m3, 0)
         GPIO.output(m4, 1)
##      else:
##       GPIO.output(m1, 0)
##       GPIO.output(m2, 0)
##       GPIO.output(m3, 0)
##       GPIO.output(m4, 0)
        if(totalFingers==5):
         GPIO.output(m1, 0)
         GPIO.output(m2, 0)
         GPIO.output(m3, 0)
         GPIO.output(m4, 0)
        h,w,c=overlayList[totalFingers].shape

img[0:h,0:w]=overlayList[totalFingers]

cv2.rectangle(img,(20,225),(170,425),(0,2
55,0),cv2.FILLED)
cv2.putText(img,str(totalFingers),(45,375),
```

```
cv2.FONT_HERSHEY_PLAIN,10,(255,0,
0),25)
    cTime=time.time()
    fps=1/(cTime-pTime)
    pTime=cTime
    cv2.putText(img,f'FPS:
{int(fps)}',(400,70),cv2.FONT_HERSHE
Y_PLAIN,3,(255,0,0),3)
    cv2.imshow("Image",img)
    cv2.waitKey(1)
    if(fps==1):
      GPIO.output(m1, 1)
      GPIO.output(m2, 0)
      GPIO.output(m3, 1)
      GPIO.output(m4, 0)
    if(fps==2):
      GPIO.output(m1, 0)
      GPIO.output(m2, 1)
      GPIO.output(m3, 0)
      GPIO.output(m4, 1)
    if(fps==3):
      GPIO.output(m1, 0)
      GPIO.output(m2, 1)
      GPIO.output(m3, 1)
      GPIO.output(m4, 0)
    if(fps==4):
      GPIO.output(m1, 1)
      GPIO.output(m2, 0)
      GPIO.output(m3, 0)
      GPIO.output(m4, 1)
    if(id==5):
      GPIO.output(m1, 0)
      GPIO.output(m2, 0)
      GPIO.output(m3, 0)
      GPIO.output(m4, 0)
```