Hand Gesture Controlled Arm

Exploratory Project by-

GUJULLA LEEL SRINI ROHAN(20095041)

AKULA HEMANTH(20095005)

ROHIT KUMAR GAUTAM (20095093)

Table of contents

- Robot Arm path planning
- Simulation to Real
- Dice number detection

References

Conclusion

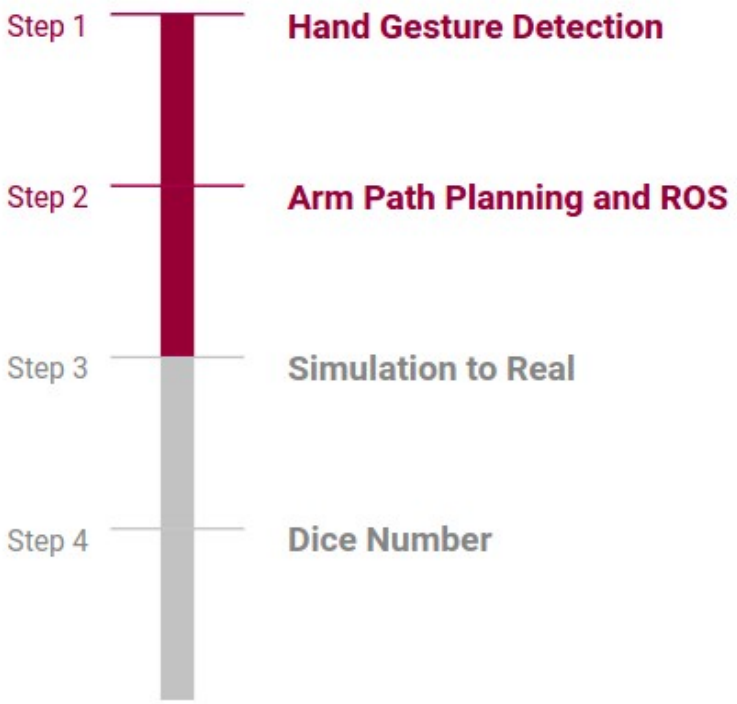
Introduction

The ability to control things by a hand seen in movies was like magic, I was so excited to know how it works and try it in real life. So I come out with a very nice project which is a robot arm that can be controlled by hand gestures .We used limited sensors to reduce the cost of hardware . Laptop camera was used to capture the image of hand and classify the gesture into Thumbs Up, Thumbs Down, and Fist.

Aim and Objectives

The main AIM of this project was to control robot arm with hand gestures. But we want arm to go to dynamic positions instead of a single position. Hence we have included dice in our project. So, the secondary AIM of this project is to identify the number on dice and make arm go to position according to the number on dice.

The path we followed to complete the project is as follows:



Theory and working

Hand Gesture Recognition

Gesture recognition is a topic in computer science and language technology with the goal of interpreting human gestures via mathematical algorithms. It is a sub discipline of computer vision. Gestures can originate from any bodily motion or state but commonly originate from the face or hand

Step1: Segmenting hand

We used laptop webcam to capture hand image. Background is subtracted from the original image to segment hand

The segmented image is converted into binary image using shareholding functions of OpenCV.

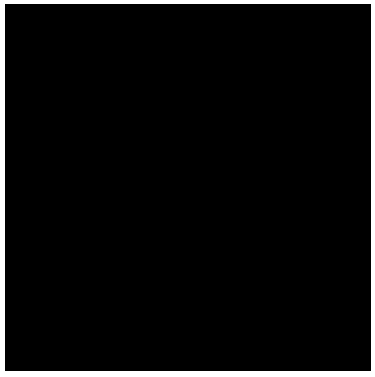
-

Step2: Dataset preparation and model training

We have manually saved 50 images from each class and done agumentation to make it 500 per each class. Divided the dataset into train and test data.

Five

Fist



Black

Super

Thumbs Down

Thumbs Up

We failed to classify super and fist hence we used it as same class. The reason we failed is mainly due to dataset as we have made dataset manually we couldnt make a big dataset.

We trained the dataset with the below model configuaration

```
model = keras.Sequential(
    [
        layers.Input((32, 32, 1)),
        layers.Conv2D(512, 3, padding="same",activation='relu'),
        layers.Conv2D(256, 3, padding="same",activation='relu'),
        layers.Conv2D(128, 3, padding="same",activation='relu'),
        layers.MaxPooling2D(),
        layers.Flatten(),
        layers.Dense(6,activation='softmax'),
    ]
)
```

With the above model we obtained a accuracy of 98% in train data and 92% on test data

Code:

```
import rospy

import cv2

import numpy as np

from segment import segment

import numpy as np


from keras.models import Sequential

from tensorflow import keras
```

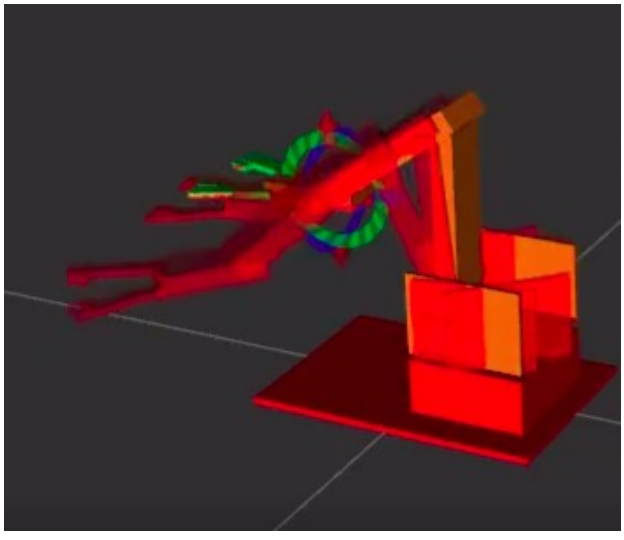
```
from tensorflow.keras import layers



cap = cv2.VideoCapture(0)


model = keras.Sequential(
    [
        layers.Input((32, 32, 1)),
        layers.Conv2D(512, 3, padding="same",activation='relu'),
        layers.Conv2D(256, 3, padding="same",activation='relu'),
        layers.Conv2D(128, 3, padding="same",activation='relu'),
        layers.MaxPooling2D(),
        layers.Flatten(),
        layers.Dense(6,activation='softmax'),
    ]
)


model.load_weights('my_model_weights.h5')
while cap.isOpened():
    ret, frame = cap.read()
    cropped_frame = frame[100:400, 100:300]
    cropped_frame=cv2.cvtColor(cropped_frame,cv2.COLOR_BGR2GRAY)
```

```
#cv2.imwrite('bg.jpg',cropped_frame)

thres,seg = segment(cropped_frame)

cv2.imshow('Captured Frame', thres)

thres=cv2.resize(thres,(32,32))

thres=np.reshape(thres,(1,32,32,1))

dic={}

dic[0]='blank'

dic[1]='fist'

dic[3]='super'

dic[4]='thumbsdown'

dic[5]='thumbsup'

dic[2]='five'

ans=np.argmax(model.predict(thres))

print(dic[ans])

file=open("gesture.txt",'w')

file.write(str(ans))

font = cv2.FONT_HERSHEY_SIMPLEX

org = (50, 50)

fontScale = 1

color = (255, 0, 0)

thickness = 2

image = cv2.putText(frame,dic[ans], org, font,
```

```
                fontScale, color, thickness, cv2.LINE_AA)

    cv2.imshow('frame',frame)#$

    cv2.imshow('cropped',cropped_frame)

    if cv2.waitKey(1) == ord('q'):

        break


    keypress = cv2.waitKey(1) & 0xFF


cap.release()

cv2.destroyAllWindows()
```

Robot Arm And Path Planning

We have used ROS MoveIT package to plan path from one pose to another instead of basic path of covering the difference.
ROS MoveIT package has a limitation that only 6 DOF should be used but we have sucessfully used it on 4 DOF but failed to move the robot gripper to a specific point. As our aim has was just to go to particular pose instead of particular point we proceeded further.



Code:

```
#! /usr/bin/env python
from __future__ import print_function
import sys
import rospy
import tf2_ros
import moveit_commander
import moveit_msgs.msg
import geometry_msgs.msg
import math
import numpy as np
rospy.set_param('gripper',0)
moveit_commander.roscpp_initialize(sys.argv)
rospy.init_node("final_node", anonymous=True)

robot = moveit_commander.RobotCommander()
scene = moveit_commander.PlanningSceneInterface()
group_name = "arm"
move_group = moveit_commander.MoveGroupCommander(group_name)
group1_name = "gripper"
gripper_group = moveit_commander.MoveGroupCommander(group1_name)
#display_trajectory_publisher =
rospy.Publisher("/move_group/display_planned_path",moveit_msgs.msg.DisplayTra
jectory,queue_size=20)
value=0
value1=0
value2=0
print('hello')
rate = rospy.Rate(10)
while not rospy.is_shutdown():

        file=open('/home/srinir/catkin_ws/src/control/src/gesture.txt','r')
        value=file.read()
```

```
    if value=='4' and value1=='4' and value2=='4':
        joint_goal = move_group.get_current_joint_values()
        joint_goal[0] = 0
        joint_goal[1] = 1.57
        joint_goal[2] = -70*3.14/180
        move_group.go(joint_goal, wait=True)
        move_group.stop()
    if value=='5' and value1=='5' and value2=='5':
        file=open('/home/srinir/catkin_ws/src/control/src/dice.txt','r')
        dice=int(file.read())
        joint_goal = move_group.get_current_joint_values()
        joint_goal[0] = dice*30*3.14/180
        joint_goal[1] = 20*3.14/180
        joint_goal[2] = 0
        move_group.go(joint_goal, wait=True)
        move_group.stop()
    if value=='1' and value1=='1' and value2=='1':
        rospy.set_param('gripper',1)
    value1=value
    value2=value1
    print(value,value1,value2)
    rate.sleep()
```

Robot Arm And Path Planning

We used Pyfirmata library which provides a easy method to mimic the pose of robot present in simulation to real.

We used 4 ports

1st port for robot base servo

2nd port for robot shoulder

3rd port for robot hand

4th port for gripper

The motor we used is a 180 degree motor hence we have limitation in the angle the robot arm can rotate. The same limitation was forced to happen in simulator so that no error will be there.

Code:

```
#!/usr/bin/env python
import rospy
from sensor_msgs.msg import JointState
from pyfirmata import Arduino,SERVO
from time import sleep
port = '/dev/ttyUSB0'
board = Arduino(port)
board.digital[2].mode = SERVO
board.digital[3].mode = SERVO
board.digital[4].mode = SERVO
board.digital[5].mode = SERVO
def rotateservo(pin,angle) :
    board.digital[pin].write(angle)
```

```
    sleep(0.015)


def callback(data):
    rotateservo(2,int(data.position[0]*180/3.14))
    rotateservo(3,int(data.position[1]*180/3.14))
    rotateservo(4,int(-data.position[2]*180/3.14))
    if rospy.get_param('gripper')==1:
        rotateservo(5,114)



def listener():
    rospy.init_node('subscriber', anonymous=True)
    rospy.Subscriber('/joint_states',JointState,callback)
    rospy.spin()


if __name__ == '__main__':
    listener()
```

Dice Number Detection

We used OpenCV to detect number of circles present in the image using Simple blob detection function.

We tuned the hyper parameters of blob detection function such that it detects only circles of specific area and curvature.

Code:

```
import cv2
import numpy as np


import cv2
import sys


vid_capture = cv2.VideoCapture("http://192.168.29.69:4747/video")
frame_width = int(vid_capture.get(3))
frame_height = int(vid_capture.get(4))


size = (frame_width, frame_height)
result = cv2.VideoWriter('bot_top_camera.mp4',
                         cv2.VideoWriter_fourcc(*'MJPG'),
                         25, size)
if (vid_capture.isOpened() == False):
        print("Error opening the video file")



params = cv2.SimpleBlobDetector_Params()
params.filterByInertia
params.minInertiaRatio = 0.6
```

```
detector = cv2.SimpleBlobDetector_create(params)


def get_blobs(frame):
    blur = cv2.medianBlur(frame, 7)
    grayscale = cv2.cvtColor(blur, cv2.COLOR_BGR2GRAY)


    blobs = detector.detect(grayscale)


    return blobs


def get_number(blobs):
    X = []
    for b in blobs:
        pos = b.pt
        if pos != None:
            X.append(pos)
    X = np.array(X)
    print(len(X))
    a=len(X)
    if len(X)>0:
        sum_x=0
        sum_y=0
        for i in range(len(X)):
            sum_x+=X[i][0]
            sum_y+=X[i][1]
```

```
        return [a, sum_x/a,sum_y/a]
    else:
        return [0,0,0]
def display_info(frame, data, blobs):
    for b in blobs:
        pos = b.pt
        r = b.size / 2
        cv2.circle(frame, (int(pos[0]), int(pos[1])),
                int(r), (255, 0, 0), 2)
    image = cv2.putText(frame,str(data[0]),(int(data[1]),int(data[2])),
cv2.FONT_HERSHEY_SIMPLEX,
                1, (0, 0, 0),2, cv2.LINE_AA)



while(vid_capture.isOpened()):


    ret, frame = vid_capture.read()


    blobs = get_blobs(frame)
    data = get_number(blobs)
    file=open("dice.txt",'w')
    file.write(str(data[0]))
    display_info(frame, data, blobs)
```

```
    cv2.imshow("frame", frame)
    res = cv2.waitKey(1)
    if res & 0xFF == ord('q'):
        break


cap.release()
cv2.destroyAllWindows()
```

References

https://www.kaggle.com/datasets/aryarishabh/hand-gesture-recognition-dataset

https://pyfirmata.readthedocs.io/en/latest/
https://golsteyn.com/writing/dice
https://moveit.ros.org/
https://www.youtube.com/watch?v=kUHmYKWwuWs

Conclusion

In this way we have successfully completed the project . The main challenging task we felt was to classify hand gestures and train the model. We successfully trained the model and classified the gesture, planned path in ROS, Used simulation to real techniques to mimic simulation into reality and finaly detected number on dice using basic functions of OpenCV.