Gesture Detection And Replication

Project Report

EKLAVYA MENTORSHIP PROGRAMME

At

SOCIETY OF ROBOTICS AND AUTOMATION, VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE MUMBAI

October 2021

ACKNOWLEDGMENT

It was great working with fantastic seniors. We never imagined we could give something a shape of which we had almost no idea of. But with our collective effort and constant guidance from the seniors we could reach our milestone yet many more to go in the future. We are grateful towards you all and thank SRA for giving us this opportunity and igniting a lamp which could produce immense light.

Special Thanks to our Superb Mentors:

- ***Anushree Sabnis***
- ***Reshmika Nambiar***
- ***Saad Hashmi***

Team Members:

- ***Gaurav Kumar***
  gaurav.og.9920@gmail.com
  +91 9920522867
- ***Dhruv Kunjadiya***
  dhruvkunjadiya55@gmail.com
  +91 9321784815

2. TABLE OF CONTENTS:

1. Project Overview:

1.1 Description of Use Case of Project:-

Hand gesture recognition can be used in many applications. Some of them are described below:

1. **Hand gesture remote to control home appliances:**

It can be used to control objects using hand gestures. A gesture based remote can be made to control our home appliances like T.V , A.C , and other appliances. As you can see in image below we can also replace touch screen by gesture recognition. For ex: You can play an Mp3 player when a certain gesture is recognised.

2. **Gesture based games:**

Imagine you are playing a shooting game and now when you show a certain gesture bullet fires and you kill your opponent

how amazing that would be.You may have to focus on things like position and timing. But in future we might see this thing happening.

There are many other applications of gesture recognition like gesture controlled car driving , medical operations etc.

1.2 Technology Used:-

Technologies that made this project possible:

- Opencv library using python- OpenCV is a library of programming functions mainly aimed at real-time computer vision. It is an open-source computer vision and machine learning software library. This library will allow us to capture the video in front of the camera and will also allow us to detect the required person in front of the screen when used with various classifiers.

- Numpy- NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including

mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

- Python- Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

- Keras - Keras allows **users to productize deep models on smartphones** (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep-learning models on clusters of Graphics processing units (GPU) and tensor processing units (TPU).

- Google colab- Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to **machine learning, data analysis and education**.

- Coppeliasim- The robot simulator CoppeliaSim, with integrated development environment, is based on a distributed control architecture: each object/model can be individually controlled via an embedded script, a plugin, a ROS or BlueZero node, a remote API client, or a custom solution.

1.3 Brief Idea (Aim)

❖ Aim of the project is to detect and recognise basic hand gestures and imitate them using a simulation of a robotic hand. Recognition of gestures can be done using various approaches like CNN (Convolutional Neural Network), feature engineering (such as binary thresholding, circle detection, etc.), and others. The simulation will be done using CoppeliaSim.In this project will be focusing on two methods-

1. opencv(contours and convexity defects).
2. CNN

2. INTRODUCTION:

2.1 General

Our main idea of this project is to detect hand gestures like one, two,thumbs up, ok….etc using open cv (convex defect algorithm) and run corresponding simulations on coppeliasim using a model which will replicate this simulations.This is done through python api functions provided by coppeliasim.

2.2 Basic Project domains

- Open CV
- CNN (Convolutional Neural Network)
- Numpy
- Coppeliasim and python API functions.

2.3 Theory

If you are using CNN approach then you need to learn deep learning and convolutional neural networks , linear algebra. Links to all required theory has been provided on github repo.

3. METHODS AND STAGES OF PROGRESS:

3.1 Prerequisite

- Download any code editor. (click here to download)
- Install coppeliasim (click here to download)
- Requires python ver 3.6 or above
- Install Numpy
- Install Open cv-python
- To install keras (if using CNN approach...install)

3.2 Hand Gesture Recognition and It's code

OpenCV has been used for Hand gesture recognition in this project. The hand Hand gestures recognisable under this project are :-

Fist vertical , Best of luck , Thumbs down , 1 , 2 , L , 3 , Ok, 4 ,5

Here's the explanation of the code used in here for Hand gesture recognition -

First the region of interest has been defined

```
roi=frame[100:400, 100:400]

cv2.rectangle(frame,(100,100),(400,400),(0,255,0),0)

hsv = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
```

Now the input has been converted from bgr to hsv image.

Now anything as with the skin colour would be taken white or 1 and anything not as with skin colour is taken black or 0, then we have dilated and blurred the image to remove any amount of noise present.

```
lower_skin = np.array([0,20,70], dtype=np.uint8)

upper_skin = np.array([20,255,255], dtype=np.uint8)

mask = cv2.dilate(mask,kernel,iterations = 4)

mask = cv2.GaussianBlur(mask,(5,5),100)
```

After that the contours have been defined, Contours are generally any area that is being shown in the region of interest.

In the next step the max contours have been found which is gonna be the area of the hand.

```
contours,hierarchy=
cv2.findContours(mask,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

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

Now the convex hull has been defined, which is the outline(the green line) around the hand.

```
hull = cv2.convexHull(cnt)
```

The area of the convex hull and the area of the hand has been calculated. Then we have taken the area ratio i.e the = [(area of convex hull - area of the hand)/(area of hand)]*100

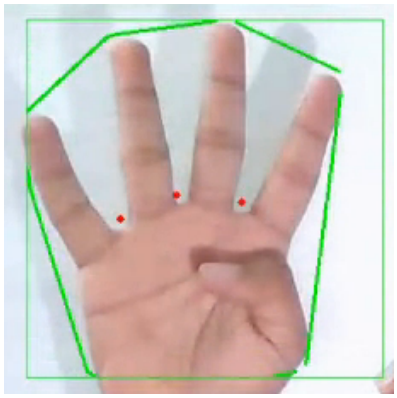
```
areahull = cv2.contourArea(hull)

areacnt = cv2.contourArea(cnt)

arearatio=((areahull-areacnt)/areacnt)*100
```

Now every gesture has a unique area ratio(percentage of area not covered by hand in a convex hull) and it’s the basis on which we are gonna define different hand gestures.

We are now gonna find the defects in the convex hull.



As shown in the picture above, the gaps haven't been underlined by the convex hull, these are the defects in the convex hull. But there are many defects in the convex hull but we only have to take the defects into consideration that matter to us, so we do that on the basis of angle.

If the angle between the fingers is less than 90 degrees and the distance between the convex hull and the red dot is more than a certain value it will count the dot(red).

```
hull = cv2.convexHull(approx, returnPoints=False)

defects = cv2.convexityDefects(approx, hull)
```

Now, the no. of dots(l) + 1 = no of fingers present in the frame.

Like in the image above the no of dots are equal to 3, hence the no of fingers get equal to 4.

We increment the value of l by one. i.e , l=+l

For l=1,

The following gestures are possible:-

I. Fist vertical
II. Best of luck
III. 1

Similarly for l=2, the following gestures are possible:-

I. L
II. 2

And so on……..

If the hand is not there in the box it says “put the hand in the box”.

Hence this is a brief explanation on how the hand gesture code works.

3.3 Coppeliasim And it’s Code:

In coppeliasim, a 7 dof arm has been used in this project. With the use of Inverse kinematics we are able to move the bot to a certain position. The gesture detection code has been linked to coppeliasim by using python API.

For us to proceed we first need to set up some files for API.

We first need to add the necessary files to our working directory that are needed for the API, those can be found in the coppeliasim installation folder.

1) Navigate to :- CoppeliaRobotics\CoppeliaSimEdu\programming\remoteApiBindings\python\python and copy all the .py files into the working directory.
2) Navigate to :- CoppeliaRobotics\CoppeliaSimEdu\programming\remoteApiBindings\lib\lib Depending on the System you are on, you can select the folder (Windows, Ubuntu 16/18, MacOS) copy the .dll, .so, or the .dylib file respectively to your working directory.

In the next step we need to establish communication with the bot. For that we run the following code :-

1) import the sim library in the code and add the below statements to the code

```
sim.simxFinish(-1)
```

```
clientID = sim.simxStart('127.0.0.1',19990,True,True,5000,5)
```

2) Now. for Testing the establishment of communication you have to run the following code after starting the simulation in CoppeliaSim : ```python import sim import sys

```python
if clientID!= -1:

   print("Connected to Remote API Server")

   sim.simxStartSimulation(clientID,sim.simx_opmode_blocking)

   _,handle =
sim.simxGetObjectHandle(clientID,'redundantRob_manipSphere',sim.simx
_opmode_blocking)

else:

   print("Connection failed")

   sys.exit('Could not reconnect')

   sim.simxFinish(clientID)
```

If the system fails to establish communication it will print “connection failed” and if the connection failed to establish. Otherwise if the connection is established it will print “connected to the remote API server”

The further moments of the bots have been made by remote API commands like...

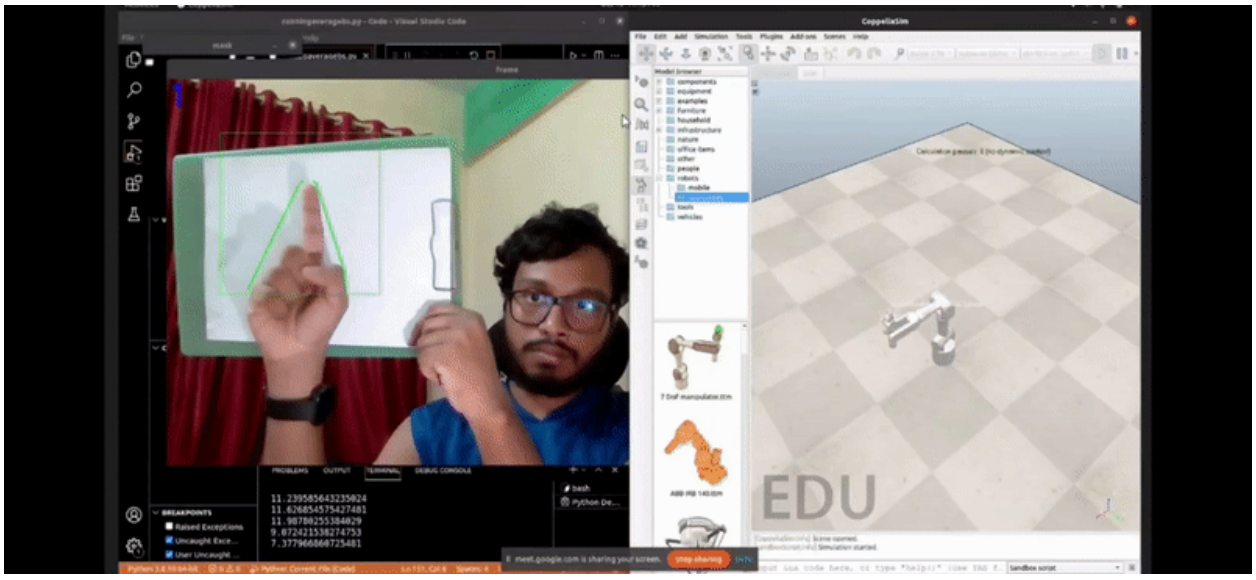
```python
sim.simxSetObjectPosition(clientID,handle,-1,(x,y,z),sim.simx_opmode_onesh
ot)
```

4. Conclusion And Future Work:

4.1 A Working Model.

With all the code and work it's now time to bring the codes and stimulations into life, as we can see the hand gesture recognition is taking place along with the corresponding simulations in coppeliasim.

Heres a demo of our project :-



[our git hub repository]

4.2 Future work

1) We would like to add more gestures and while replicating it on coppeliasim we can keep some particular target and try to reach that with help of gesture detection. Also we can add some text in coppeliasim to describe which gesture was detected at that movement.

2). We could make a gesture controlled game. Not yet thought but Could try in future.

5.Useful resources and links:

Some project resources and links:-

Deep learning specialization:

https://www.coursera.org/specializations/deep-learning

Linear algebra:

https://www.youtube.com/playlist?list=PL0-GT3co4r2y2YErbmuJw2L5tW4Ew2O5B

CNN theory:

https://www.ibm.com/cloud/learn/convolutional-neural-networks

CNN video:

https://youtu.be/FmpDIaiMIeA

Hand gesture model using CNN:

Hand gesture model using CNN's

Contours approach(using convexity defects)

contours approach (refer this doc for more clarity for contours approach)

Remote API functions coppeliasim:
Remote api functions coppeliasim

To make coppeliasim scene follow this:
To make coppelia scene follow this