ROBOCON IITR

Github repo:- https://github.com/14-somil/Arrow-detection.git

Aim:- Arrow Detection

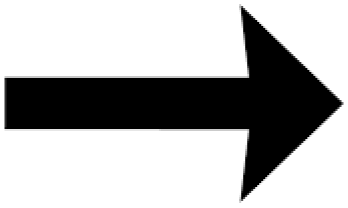
Problem:-

Purpose is to construct an algorithm using which an autonomous robot can capture images of an arrow and detect the direction that they are pointing so that it can move in the shown direction

Tech used:- Python programming language, OpenCV, numpy, pyfirmata and matplotlib libraries of python and Arduino.

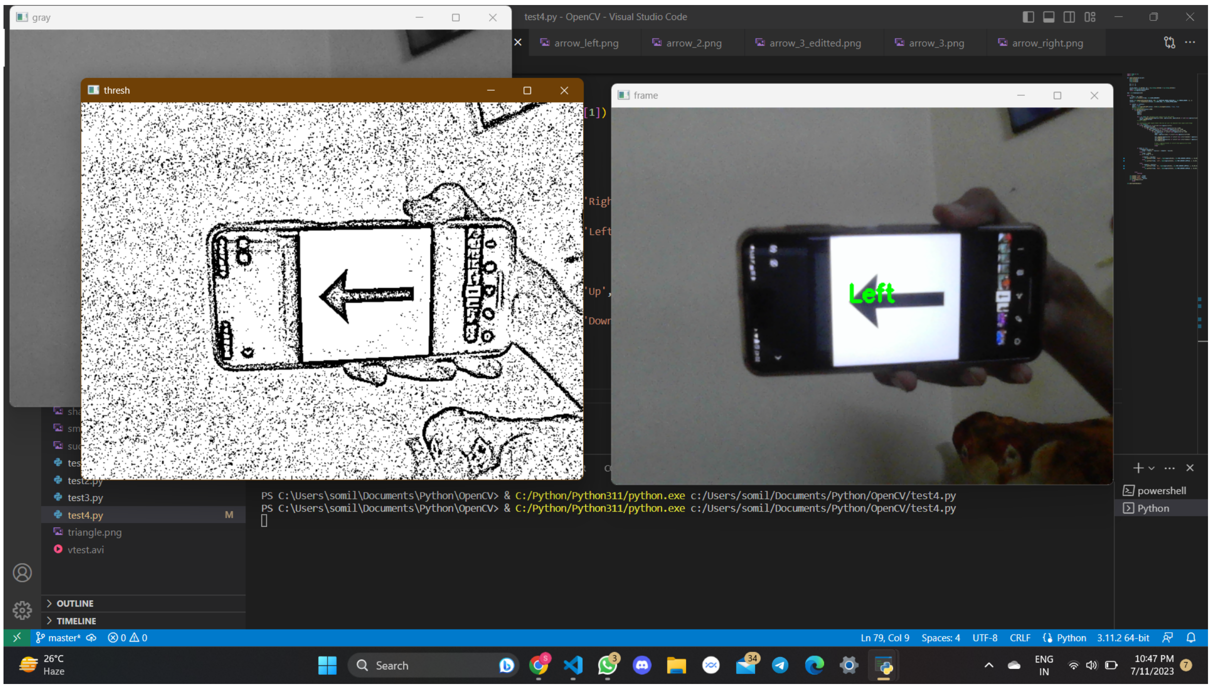
Our solution:-

- We used the geometric properties of the arrow shown below to distinguish it from different shapes.

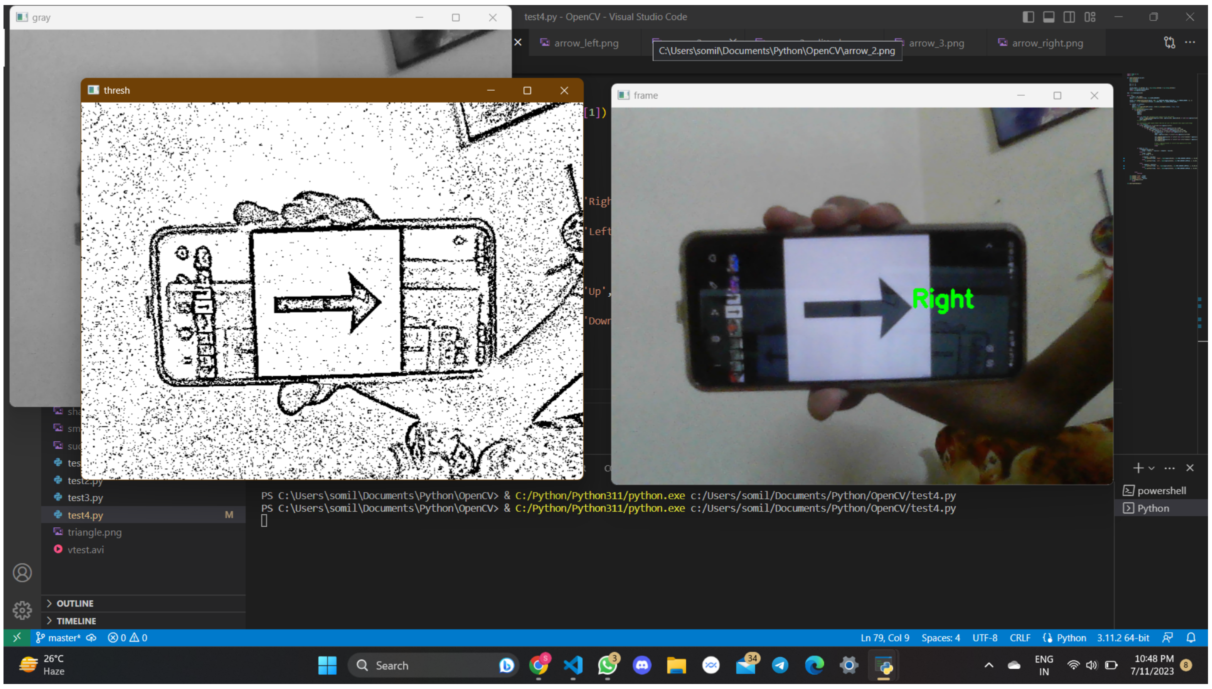


- Consider the head of the arrow as vertex 1, and the following vertices be 2, 3, …, 7 in clockwise order.
- For these types of arrows vertex 4 and 5 are nearly right angles, and vertex 2, 3, 6 and 7 can be anything between 0 to 90 degrees.
- To detect this each frame of the captured video is first converted into a black and white image.
- Then using the adaptive threshold method of OpenCV we converted it into a binary image to detect the edges in the frame.
- Then using findContours method we contours in the image were detected.
- Then for each set of contours we approximated it to a closed polygon using approxPolyDP method.
- The polygon was ignored if it either didn't have 7 sides or had an area less than 1000 or both.
- Then using the dot product method of the numpy library, the angle of each vertex was calculated.
- Then using the above written properties of the arrow, vertex 2,3,4,5,6 and 7 were determined and the remaining vertex was called the <u>Head</u> and the midpoint of vertex 4 and 5 was called <u>Tail</u>.

- Now if the Head-Tail slope was between -1 and 1, and Head is to the right of Tail it is concluded that it is a right pointing arrow. And if Head is to the left of Tail, then left pointing arrow.
- If the arrow pointed right then the built-in LED of Arduino lighted up and if it pointed left it blinked, accompanied with text written on screen for both cases.



```
PS C:\Users\somil\Documents\Python\OpenCV> & C:/Python/Python311/python.exe c:/Users/somil/Documents/Python/OpenCV/test4.py
PS C:\Users\somil\Documents\Python\OpenCV> & C:/Python/Python311/python.exe c:/Users/somil/Documents/Python/OpenCV/test4.py
```


gray
thresh
frame
Right
test4.py - OpenCV - Visual Studio Code
C:\Users\somil\Documents\Python\OpenCV\arrow_2.png
test3.py
test4.py
triangle.png
vtest.avi
OUTLINE
TIMELINE
powershell
Python

```
PS C:\Users\somil\Documents\Python\OpenCV> & C:/Python/Python311/python.exe c:/Users/somil/Documents/Python/OpenCV/test4.py
PS C:\Users\somil\Documents\Python\OpenCV> & C:/Python/Python311/python.exe c:/Users/somil/Documents/Python/OpenCV/test4.py
```

The left window is the output after applying the adaptive threshold and the right one is the final output.

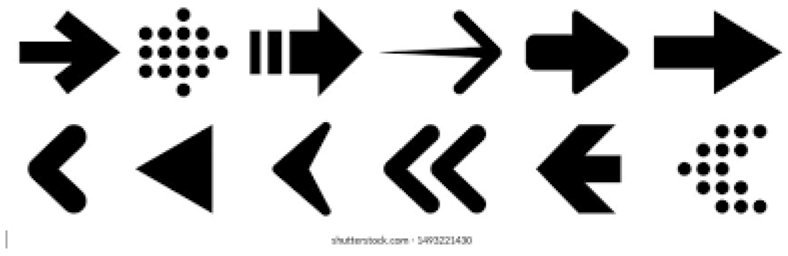
Problems we faced and their solution:-

- First we tried using absolute threshold or threshBinary method but it results in failure of code in difficult lighting conditions.

- Due to failure of the IMU unit it couldn't be applied to an actual bot which can be resolved if we had used an encoder to determine the orientation of the bot.

Scope of further improvement:-

- The Region of interest (ROI) can be reduced to focus only on the centre image.
- Noise reduction
- Using masking to determine arrows of certain colours only.
- Using neural network and machine learning models to determine arrows other than discussed above. Like:-


shutterstock.com · 1493221430