DIGIT RECOGNITION WITH POINT TO POINT TRANSLATION

- R.S.Greeshwar
- 114119036

PROGRESS :-

I have completed all the phases (phase1 ,phase2 and phase3). But in case of phase 3 I need to improve my model because the model which I made includes only Proportional Controller for turning around a node and for translation in the straight (PWM voltage controller for speed) ,if I add derivative and integral to that it might work pretty well ,with least error .but this could complicate things because I might want do add and tune I and D terms for both the controller.

So I am working on implementing phase 3 with the help of pure odometry instead of involving the magnetometer for calculating the angle , because the magnetometer(QMC5883L) which used had some error in calculating angle of heading.

IMPLEMENTATION :-

- So 1st I started with phase 1 where I had to capture the coordinates which I write in the air using something to threshold with ,in this first I wrote a code using OpenCV and python to capture what I write in air with the marker and draw it on an image continuously as I move the marker(or any circular object of a specific colour ,in my code I did for green colour) in the air.
- Then I made a code to capture the coordinates x,y that I write using marker in the air and store the image of x and y coordinates separately by detecting the comma(,) in the image to pass it to a digit recognition model .

- After I completed phase 1 ,I started phase 2 and phase 3 parallelly .I started with a course on deep learning to learn the basic concepts required to build a digit recognition model , it took me 5- 6 days to learn till the concepts which require to build a digit recognition model like linear regression, neural networks , deep neural networks and CNN convolution neural networks.
- Then I started building my model .I used the LeNet architecture to train my model with the MNIST dataset ,I trained my model with google colab .i plotted a graph with training error and validation error for every epoch (the batch size of my training and validation dataset was 100).I didn’t obtain a proper graph ,I made adjustments with the no. of epoch and learning rate to generalize my model and to prevent overfitting error.
- It took me 3- 4 days to complete the model .parallelly I ordered the parts to build my bot online and for some parts I went to the respective shop and bought them .I extracted the weights of my trained model from colab as .pth file to attach it with my python code. Then I made a code to modify the image of x and image of y to pass it into the model for digit recognition.
- For recognizing minus sign I made a code with OpenCV itself.
- I then assembled the bot and started of building the code for phase 3 in Arduino ,I used Arduino UNO for this project.
- In phase 3 I partitioned the problem into 3 segments .1st was to build an algorithm to obtain the x,y coordinate from the user to calculate the angle to head(with respect to the True North at that location or wrt the coordinate system whose y axis makes some angle with the magnetic North in clockwise direction) and the shortest distance possible(point to point) to reach that point.2nd was to build a controller to make the bot rotate towards the destination coordinate .I made a proportional controller to both have a controlled rotation on both the wheels based on the feedback from the encoder and the magnetometer .3rd was to build a controller which makes the bot move in a straight line (towards destination point) with distance calculation to stop at that point based on tick counts from the encoder of both the wheels .after this I clustered the 3 segments of my phase 3 to a single Arduino code.

- After finishing the 3 phases separately I clustered all the phases together.so to communicate the recognized coordinates to the bot I used pyserial and Bluetooth module.

TIME DURATION FOR EACH PHASE :-

- Phase 1 and Phase 2 == 2 and a half weeks
- Phase 3 == 1 and a half weeks
- The above mentioned time limit is approximate and also not in order ,like I worked on all the 3 phases together for 4 weeks based on the error I obtained while clustering the 3 phases.

PROBLEMS FACED :-

1. In the digit recognition model even after tuning the learning rate I didn't obtain a proper curve (validation loss versus no. of epochs).like my model didn't properly fit the validation set or like my model overfitted the training data.

2. When I gave the x coordinate image (obtained from ROI based on the bounding rectangle function on the contour of x image). The model didn't predict properly even after it was properly generalized to fit new images.

3. In case encoder I used an optical encoder ,while using it with interrupts I had a lot of error(like it didn't count the described no. of ticks for 1 revolution ,like it counted 100 ticks for 40 ticks) .i also attached a rotary encoder to the wheel shaft even with that I had the same error .but the encoders worked properly with polling.

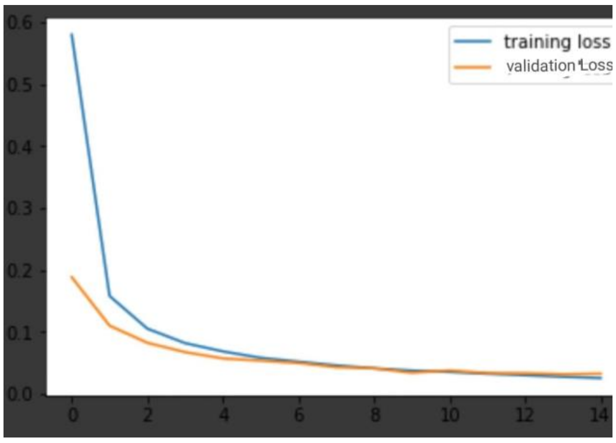
4. While using the magnetometer HMC5883L it didn't work properly with the code I made with wire.h to establish I2C communication with the Arduino UNO to find the heading angle via magnitude of magnetic field in that location.

5. While calculating the angle from magnetometer my readings changed constantly when it was placed between the centre of two motors(right and left) and near the battery.

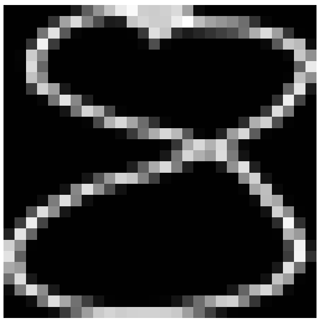
OBTAINED SOLUTIONS :-

1. So I added another layer called the dropout layer in between the fully connected layers .what it does is turns off a certain fraction nodes at random in the fully connected layers which improves the learning process of each nodes individually, to produce better predictions.

   This is how the training loss versus the no. of epochs graph(blue colour) and validation loss versus the no. of epochs graph(orange) Looks after usage of dropout layer.

2. The problem was the resized ROI image looked like this.



That is the features of the image on the corners where not retrieved or recognized by the convolution layer so I added some padding or added some extra black pixels around the image (like while taking ROI based on x,y,w,h values obtained from bounding rectangle function I took ROI as

```
ROI_x = x_img[(y - 40):(y + h + 40), (x - 40):(x + w + 40)]
```

Instead of

```
ROI_x = x_img[y :(y + h ), x:(x + w)]
```

)

3. The problem with the encoder was switch bouncing , I came over it by software debouncing .where the ticks are counted only when the ISR function is called after a certain milliseconds (found by making the motors to run at PWM of 255 and finding the average actual time interval between 2 ticks ,hence not considering the interrupt call when the time interval between the previous call of ISR and current call of ISR less than that average time interval).
   I didn't use hardware debouncing because I was in lack of capacitors and resistors .also when I used rotary encoder by attaching it to the shaft of the wheels after sometime of use the rotary encoder became loose and didn't work properly so I shifted to an optical encoder.

4. The magnetometer which I had was named HMC5883L but the one I actually had was QMC5883L so the address which I used in the code didn't match address to establish I2C communication

with Arduino and magnetometer. So I used a library from github to find the heading angle.

5. This was because of the electro magnetic interactions with battery and DC motors .so I placed the magnetometer in the front of the bot.

RESULTS :-

The digit recognition model has an accuracy of 98.5 – 99 %

The proportional controller for straight line movement along with distance calculation – when I made it to move 100cm along a particular angle made with true N it calculated the distance accurately(with encoders),but it stopped 1 – 2cm before the mark.

VIDEOS:-

In this the topics including FINAL are the final videos for the respective phases other videos are step by step process videos:-

PHASE 1:-

Phase1 step1:-

https://youtu.be/EwFA34ldsxA

Phase1 step2:-

https://youtu.be/gr_m3tF5p4M

PHASE 3 :-

Proportional controller for moving in straight line *(along a certain heading angle which I enter):-*

Description : -

I introduced some error to it to check whether it followed the angle properly

https://youtu.be/HVFEHlqr2Zk

Proportional controller for moving in straight line with distance calculation(optical encoders)

(along a certain heading angle which I enter):-

Description :-

In this video the bot moves a distance of 100cm

https://youtu.be/td_rFEItTsM

FINAL PHASE 1 AND PHASE 2:-

(Separating x,y and Digit Recognition)

For 1 digit and 2 digit nos. :-

https://youtu.be/QaYx-CPZhG0

For 3 digit no. :-

https://youtu.be/-JJiSkNpzAs

FINAL PHASE 3 :-

The bot is placed at 0,0 or initial point

https://youtu.be/wvqpsQEWj98

FINAL PHASE1,PHASE2 AND PHASE3 :-

While taking this video battery(of the bot) was low and also this video is just to show that serial communication between Arduino and python works properly ,kindly check

FINAL PHASE1 AND PHASE2 and FINAL PHASE3 before watching this .

https://youtu.be/panlfAg9dAQ