# Autonomous Navigation of a Quadrotor in an Indoor Environment using Deep Learning

Shanmuga Perumal Harikumar,Master of Science in Robotics Engineering,WPI

***Abstract*—The purpose of this project is to develop an autonomous quadrotor system that could navigate an indoor environment without running into obstacles. This documents outlines the project in which the author uses a deep neural network to generate motion commands based on the input image. The sensor data is directly mapped to the control input to the system by the deep learning network. An problem statement is formulated.Previous works in the field relating to the project undertaken are explained in brief. The various hardware and software components to be used in the project are listed,along with a short description of each. Additionally the training and testing details are explained.Visualization of intermediate convolutional layers is provided and analysed.**

***Index Terms*—Deep learning, Autonomous Navigation, Convolutional Neural Networks, Quadrotor Indoor Navigation**

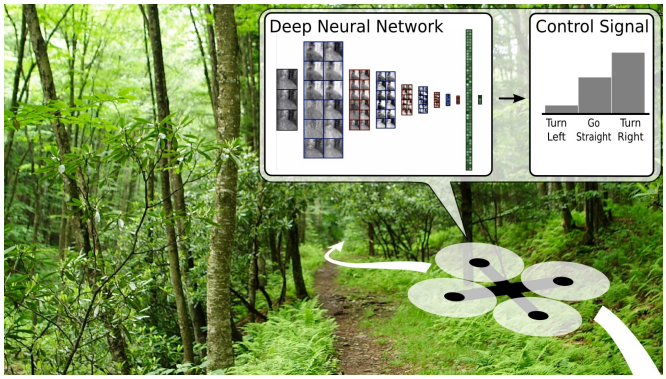


Fig. 1. Neural network controlled drone

## I. Introduction

The purpose of this project is to make a quadrotor learn navigate through an indoor environment autonomously using deep learning. In other words, the author hopes to map sensor input directly to motion commands.The quadrotor must learn to detect and avoid obstacles in its path. The application of such a learning system is immense. The quadrotor market is currently largely focused on outdoor quadrotors that rely heavily on GPS for positioning and way-points navigation. There is however a dire need for quadrotors that can operate in an indoor environment. There are several problems that need to be addressed in order for a quadrotor to successfully fly indoors. The problem of navigating through a cluttered indoor space such as a warehouse or an office is yet to be solved. This project aims at using deep learning to solve this problem.

The majority of current research in this field of indoor navigation relies heavily on lidars and creating a 3D map of the environment. Creating a 3D in itself is both computationally expensive and memory intensive. SLAM algorithms require a huge amount of computing power to build the maps and localize based on the maps. Navigating such a mapped environment is a whole other topic. Motion planning involves collision checking which takes the majority of the planning time and is again computationally expensive.

Unlike the above mentioned methods that use features that are handcrafted by humans in order to localize and map a place, Deep learning networks can do all this implicitly. Deep learning networks have been proven time and again that they are better at tasks such as object recognition, classification and segmentation than the other methods. Such networks with minor tweaking can be used to give commands to UAVs based on visual input data. The process of controlling a MAV can be modelled as a classification problrm that the DNNs are really good at.

Convolutional Neural Networks(CNN) have been used in object recognition with high success rates.Before CNNs object recognition was done by using hand crafted features which required domain experts and required special features to be developed for each specific application which was laborious. The development of Deep Neural Networks(DNN) led to the process of classification becoming more general.

Section 2 talks about the present research in the field of autonomous navigation. Section 3 talks about the hardware and the software components used in this project. Section 4 talks about data collection,network training and testing and overview of the system. Section 4 talks about the various tests that were performed to validate the network.Finally section 6 talks about the conclusion and the future work that the author hopes to complete.

### A. Problem Statement

In this project the author hopes to train a deep neural network to detect obstacles and navigate in an indoor environment avoiding the detected obstacles. Initially the plan was to use as inputs to the network image and depth information from a RGBD camera(Intel Real Sense) along with IMU data as labels. Since the Real sense camera was rendered useless due to an accident, a parrot AR Drone with a monocular camera was used and the images from the camera along with the commands will be given as inputs to the drone.

## II. Previous Work

There have been several previous attempts at using Deep Neural Networks for navigation of a robot.[1] used DNN to

navigate a forest trail autonomously. The training data was obtained by using three cameras mounted on the helmet of a hiker. One of the cameras was pointing straight,one towards the left and one towards the right. The hiker was made to follow the trail while always turning towards the direction the trail was heading in.The cameras pointing towards the left always had the trail towards its right and the camera pointing towards right always had the trail to its left.The images were also flipped and the dataset was augmented.The problem was treated as a classification problem. The final network took an image as a input and gave the direction towards which it thought the trail was in as output. The final output was something like a probability. The difference between the probability of turn left and turn right commands was used as the yaw command and the probability of going straight was used as the velocity.

[5] uses a somewhat similar approach to control a car using an input image. The input from the sensor is mapped to control commands for the vehicles to follow. The data acqisition setup has three cameras ,one pointing straight and the other two at a fixed angular offset on either sides. The dataset was further augmented by distorting the input images by applying fixed transformations. The data was obtained from various roads under various weather conditions,to ensure good learning and generality. The learned parameters were first tested on the simulator and subsequently on a real car.The authors claim that the car was autonomous 98% of the time. This shows that it is feasible to map the sensor inputs directly to control outputs

[2] uses the depth information along with the rgb data for the process of object detection. the rgb images and the depth information are processed as different channels. Pretarined model as are first used an then the separate channels are trained separately and finally the networks are combined and is trained one final time.. one more novel approach is hoe they do scaling of the images ,instead of just warping the image , the authors use tiling the borders along the axis of the shorter side. Since the depth data also contains features such as edges and corners that are visible in rgb images the depth channel was also trained using the image net database. This paper also talks about the various methods of encoding the depth data namely 1)rendering the depth data as greyscale images and then replicating across three channels, 2)using surface normals 3)HHA encoding,where the three channels are height above the ground ,horizontal disparity and pixelwise angle between surface normal and gravity direction.

[3] explored the depth parameter of the neural network to improve the accuracy. They experimented with a variety of network architectures. VGG 19 and VGG 16 two of the most successful networks according to the experiments conducted by the authors ,performed exceptionally well at classifying and also a variety of other tasks

## III. Hardware and Software Used

### A. Parrot AR Drone

Parrot ARdrone is a small lightweight platform that is also inexpensive,making it the ideal platform for research. It is

Fig. 2. Parrot AR Drone

equipped with a forward facing camera that can record in 720p, a downward facing camera for optical flow,a downward facing ultrasonic sensor for altitude measurement. The drone flies stably out of the box. the availability of a software development kit also makes this an attractive choice for students. Due to this drone being small and light,it is perfect for an indoor drone flying in warehouses. Even crashig into stuff wont do much damage to either the drone or the products at the warehouse.

### B. keras

Keras[7] is a highlevel deep learning library that runs over theano or tensor flow in the backend. it is designed with ease of development in mind. This lets us create neural networks with different kinds of layers seamlessly. It also allows for easy debugging and experimenting with different architectures. It can run on both CPU and GPU hardware without any external changes to the configutration. the backend can be changed from theano to tensor flow or vice versa by modifying the .json file.

### C. Robot Operating System(ROS)

Robot Operating System(ROS) is a software framework for development of software for robots specifically. It is an opensource implementation and has a very active community. Using ROS simplifies communication between the various parts of the system and help visualize the entire system better. There are hundreds of packages available in ROS ranging from perception to motor drivers.

### D. Parrot ROS Package

Parrot provides a SDK that lets programmers create their own applications for the drone. There is a ROS package available that transmits data from the drone in separate topics and also sends data back to the drone in topics. This makes development much easier for a roboticist. This also makes the drone readily interfaceable with other open source ROS packages available.

TABLE I
DATASET

| | Go_straight | Slide_left | Slide_right | Turn_left | Turn_right | Total |
|---|---|---|---|---|---|---|
| **Train** | 1776 | 541 | 409 | 213 | 521 | 3460 |
| **Test** | 178 | 151 | 114 | 100 | 152 | 695 |
| **Total** | 1954 | 692 | 523 | 313 | 673 | 4155 |

## IV. BODY

### A. Data Collection

There exists indoor image datasets online,but they are not suitable to train the deep learning system that we need.For the purposes of this project a data set with corresponding flight commands was needed. This meant that a custom data set had to be recorded. The indoor environment chosen was warehouses.This was perfect for a deep learning navigation network because of the aisles and regularly arranged boxes. This permits for the DNN to learn the necessary features to track and follow. This also avoids the drone from getting close to the shelves and avoid collisions.

Initial idea was to give the IMU data directly as input, but this was discarded as the IMU data was too noisy and the final system trained using such noisy data would not be robust.The drone was flown along multiple aisles and commands were recorded along with the images from the front facing camera. Human pilot gave the drone instructions to move forward,left right or yaw left, right. This will be the output of the classifier as well. This approach of recording the data made the training unnecessarily complicated. In order to make use of flow_from_directory feature of the image data generator in keras,a different approach to data collection was used.

The drone was hand flown to record the relevant data for a particular class at a given time. This was repeated to collect data for all the five classes. A total of 4115 images were collected by hand flying the drone along two aisles. The drone was carried in such a way to simulate an actual flight with sudden pitch motions . Care was taken to record all the edge cases.

The data was recorded in ROS bags. A ROS package called image_view was then used to convert these rosbag files into jpgs and were stored in the corresponding folders.Of the 4115 images 255 images were initially separated out to be used as testing data and the remaining were used as training data. This resulted in validation accuracy being always close to a 100%. The size of the testing data was then increased to 690 images and the remaining images were used for training. The separation of the data set was done manually at random.

### B. Networks

The original problem is now redefines as a classification problem.There are many networks available that can do the process of classification with high accuracy. VGG 19 was chosen as the classifier for the problem at hand. VGG 19 and VGG 16 were a result of a study [3] at xford university to investigate the influence increasing the depth has on the accuracy of a deep neural network.

Fig. 3. Parrot in warehouse

*1) Architecture:* The input to the neural network during training is 224x224.The convolutional layers all have filters of size 3x3. There are five max poooling layers in total between some of the convolutional layers. The max pooling is done on 2x2 window with a stride of 2. The max pooling and the convolutional layers are followed by three fully connected layers. the first two of which have 4096 channels each and the last one has five for each of the motion primitive classes. All the hidden layers use ReLU non-linearity. The width of the convolutional layers start out at 64 in the beginning to 512 at its maximum.Because of this the number of weights in this network is smaller than some shallower wider networks.

The smaller size of filters (3x3) have smaller receptive fields. But as the authors note in the paper a stack of three 3x3 filters has an effective receptive field of 7x7. The advantage of using these smaller filters is that there are three times more ReLU units ,which make the network more discriminative. Not only that, the smaller windows also reduce the number of parameters significantly.

### C. Training

For our network categorical cross entropy loss was used with ADAM optimizer. The network was trained by changing a few of the parameters and finding the best weights. The weights for the network was initialized using the original vgg19 weights. Then only the final fully connected layers were trained on the new data set. Table:II describes the various testing configuration used. The parameters that were changed for the various configurations are the rotation shift,horizontal shift and vertical shift. Keras allows the user to augment the data set by introducing these transformations to the original image. Four different configurations were chosen and the corresponding training and validation accuracy and loss were recorded. All the four configurations did well with this data set. This might be due to the small size of the data set being

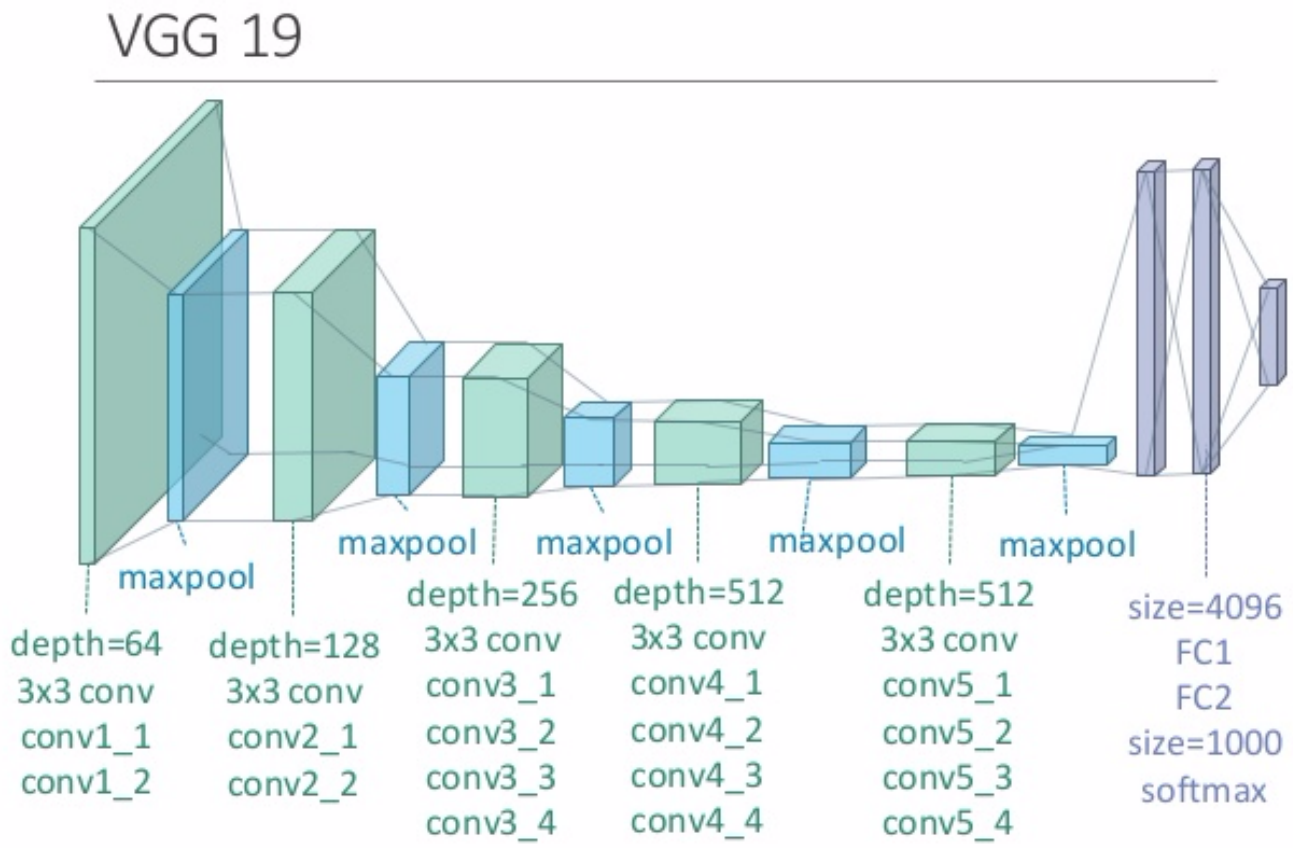


Fig. 4. vgg_19 network


Fig. 5. Validation loss for each config

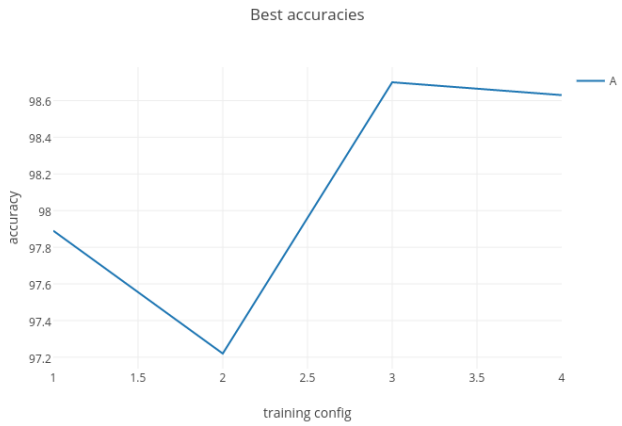


Fig. 6. Validation Accuracy for each config

used for training and also the small test validation set being used.

### *D. Analysing the output of the layers*

Keras has functionalities that lets us look at the activation of the neurons. This gives us some insight as to what features the network sees. This further helps us in fine tuning the kernel

TABLE II
TRAINING CONFIGURATIONS

| | Rotation range | Height Shift Range | Width Shift Range |
|---|---|---|---|
| **Conf 1** | 15 | 0.1 | 0.1 |
| **Conf 2** | 5 | 0.1 | 0.1 |
| **Conf 3** | 5 | 0 | 0 |
| **Conf 4** | 0 | 0 | 0 |


Fig. 7. Training loss for each config

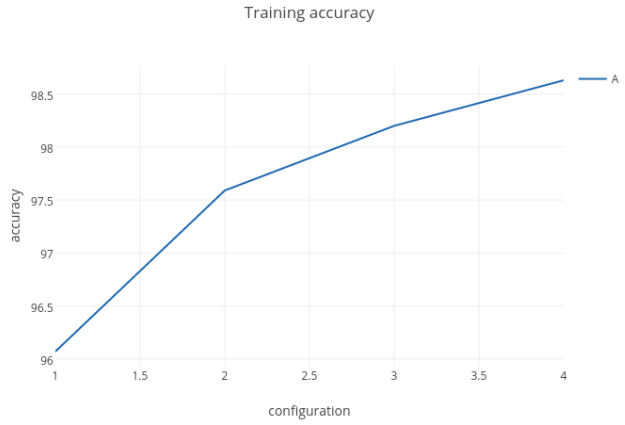


Fig. 8. Training Accuracy for each config

sizes and the depth of the layers in order for optimal feature extraction and classification.

The images of the weight layers are interesting to look at. It can be clearly seen that the network zeroes in on the edges of the aisles that run towards the centre of the image. As we go deeper the edges become more and more prominent and is the most dominant feature being tracked. In addition to the edges the network also tracks the barcodes and labels on the boxes or tubes. It is also interesting to note that the network is able to classify images that were taken really close the aisles with empty spaces(boxes) with just the beam of teh shelf being visible.

### *E. System Overview*

The drone communicates with the Neural network through the ardrone_autonomy ROS package. The package publishes

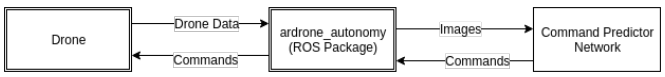


Fig. 9. System Overview

all the relevant information about the drone in separate ROS topics and the network input is the image from the front facing camera. Once the images are received from the drone,they are converted from ros images to cv images and sent as input to the predictor. The network then uses the image to generate a list of confidences for each class. The confidences can then be used to calculate the velocity commands that needs to be sent to the drone. The difference between the confidences of Turn_right and Turn_left classes can be use to scale the yaw command sent to the drone,similarly the difference between the slide_left and slide_right can be used as scaling for the y velocity of the drone. The Go_straight confidence can be used as the forward velocity of the drone.

However for now the class corresponding to the topic with the highest confidence,that is higher than a threshold, is used as the command. The ROS node converts the commands to the corresponding twist values and the resulting Twist commands are sent to the drone over another ROS topic $cmd_vel$through the ardrone_driver node.

**Data:** image from drone
**Result:** command to be executed
Take off;
**if** *drone in air* **then**
  Read frame from drone;
  Send frame to classifier;
  Receive predicted command;
  **if** *predicted command has low confidence* **then**
    Hover Drone;
  **else**
    Send command corresponding to prediction for the drone to execute:
    go_straight,slide_left,side_right,turn_left,turn_right;
  **end**

**Algorithm 1:** Algorithm for autonomous navigation in warehouse

The ROS node that sends commands to the drone is written in python

## V. Experiments

six images were picked from a dataset that was not originally used in testing or training. This dataset does not have labels and human intervention is required to know the correct class. The six images were given as inputs to the best of the four configurations.

Each image was sent ten times through each configuration. That is sixty classifications per configuration. The results are tabulated in table III . Accurate classifications are the correct classifications with confidence values grater than 0.75 . Accurate low confidence are the correctly classified instances with confidence value less than 0.75 and the misclassification refers to the instances that were classified under the wrong class.

As we can see from the results the classifier was really robust to any kind of images at what ever scale. All the configurations performed equally well in classifying the images

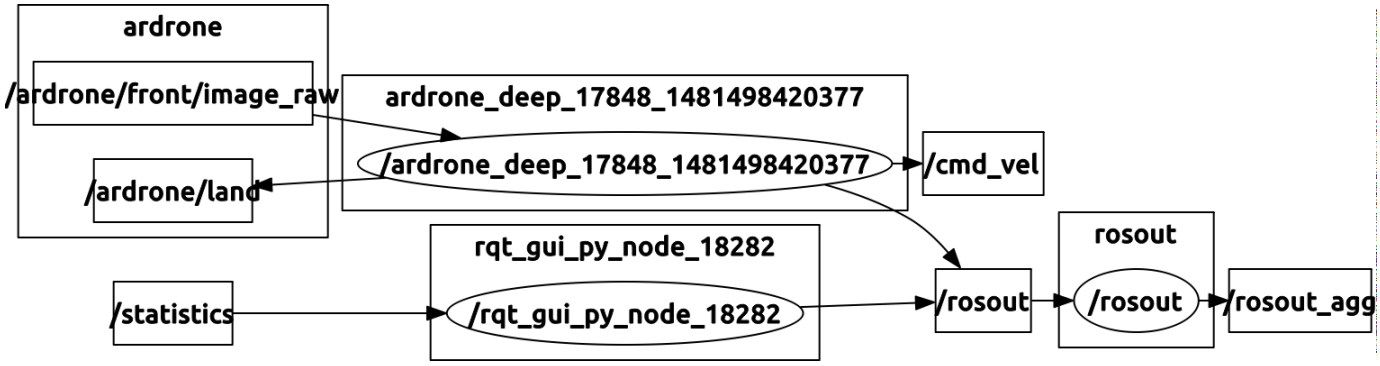


Fig. 10. Image showing how the various nodes are connected

TABLE III
TEST RESULTS

| | Accurate Classification | Accurate low confidence | Misclassification |
|---|---|---|---|
| **Conf 1** | 58 | 2 | 0 |
| **Conf 2** | 59 | 1 | 0 |
| **Conf 3** | 59 | 1 | 0 |
| **Conf 4** | 59 | 1 | 0 |

Fig. 11. weight layers image 1

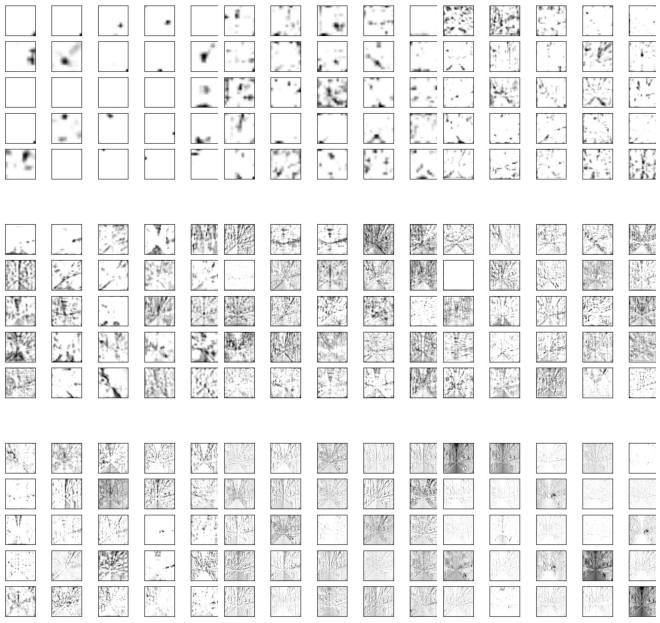

Fig. 12. weight layers image 2

to the respective classes. This shows that such system can be deployed at warehouses to reliably traverse along the aisles autonomously.

## VI. CONCLUSION

An autonomous system based on deep neural networks that is capable of navigation without bumping into obstacles was developed. This system uses just a single camera as input and navigates an indoor environment without the need of GPS or huge 3D maps. The trained network was tested with various images from a warehouse and the results are promising. The network was able to classify all the images given to it correctly and with high confidence.

In the future I hope to implement this system on a real quadrotor. Though the code for integration is already ready, Difficulty in installing keras on a local system and the restrictions in getting access to a warehouse limit has limited such experiments. I would also like to develop a system with rgbd sensors, that can can theoretically avoid dynamic obstacles. If possible I would aslo like to improve the data set to include other warehouses to make the system more general.

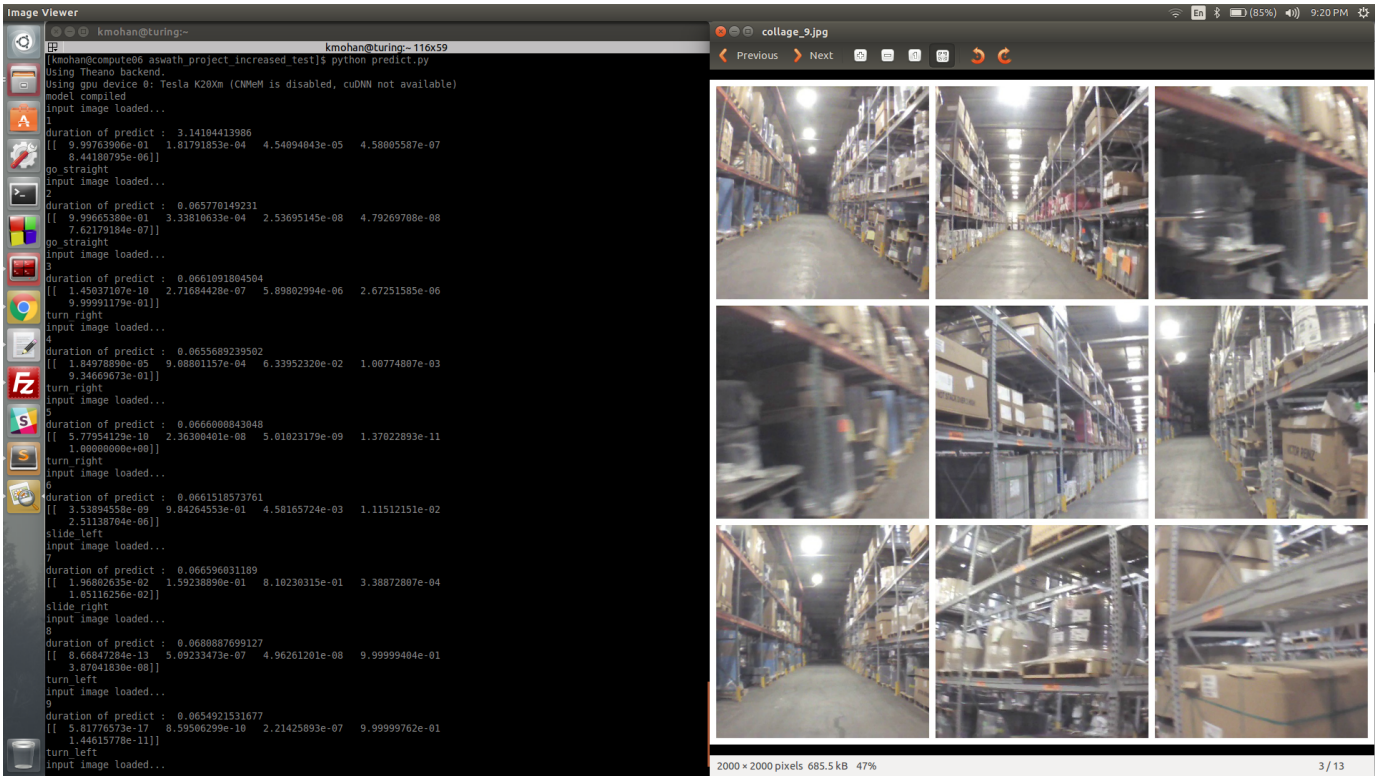


Fig. 13. Terminal on the left shows the result of the classification and the corresponding images are shown on the right. The top left one being the first image folowed by image 2 to its right

## VII. Acknowledgements

I would like to thank professor Carlos Morato for his guidance and invaluable inputs.The assignments and paper reviews were instrumental in helping me understand the concepts of deep learning.I would like to thank the CEO of 89Robotics, Mr.Jackie Wu for permitting me to use the company property for the purpose of the class project and also for driving me to the warehouses and helping in the collection of data. I would also like to thank the CTO Mr.Spencer Williams for being patient with me and working around my schedule for the Deep learning course to assign me tasks. I would also like to thank my friend Kiran Mohan for his help throughout this course.

## References

[1] Alessandro Giusti,Jerome Guzzi,Dan C. Cirezan,Fang Lin HE, *A Machine learning Approach to Visual Perception of Forest Trails for Mobile Robots*, IEEE Robotics and Automation Letters,Nov 2016
[2] Andreas Eitel, Jost Tobias Springfield,Luciano Spinello, Martin Reinmiller, Wolfram Burgard,*Multomodal Deep Learning for Robust RGB-D Object Recognition*
[3] Karen Simonyan , Andrew Zisserman, *Very Deep Convolutional Networks For Large-Scale Image Recognition*
[4] J .Engel, T.Schopps, and D.Cremers *LSD-SLAM Large-Scale Direct Monocular SLAM*
[5] Mariusz Bojarski,David Del Testa, Daniel Dworakowski Bernhard Firner, *End to End Learning for Self Driving Cars*, Nvidia Corporation
[6] DJI website, *www.DJI.com*,DJI 2016
[7] Deep learning, *https://keras.io/*
[8] Deep learning, *http://deeplearning.net/software/theano/*
[9] Intel real sense, *https://software.intel.com/en-us/realsense/devkit*