3D object recognition with Simulated Data

Joseph Broderick

Abstract

Within this project we attempted to create a system that is able to recognize the different depth images of 3d objects. We did this by creating a simulation in the Robotic Operating System (ROS), that allowed us to gather the data that we needed quickly. Using this data the hope is that we are able to extend out to other objects and models in the future. We find it quite difficult to get the model to recognize the objects well.

Introduction

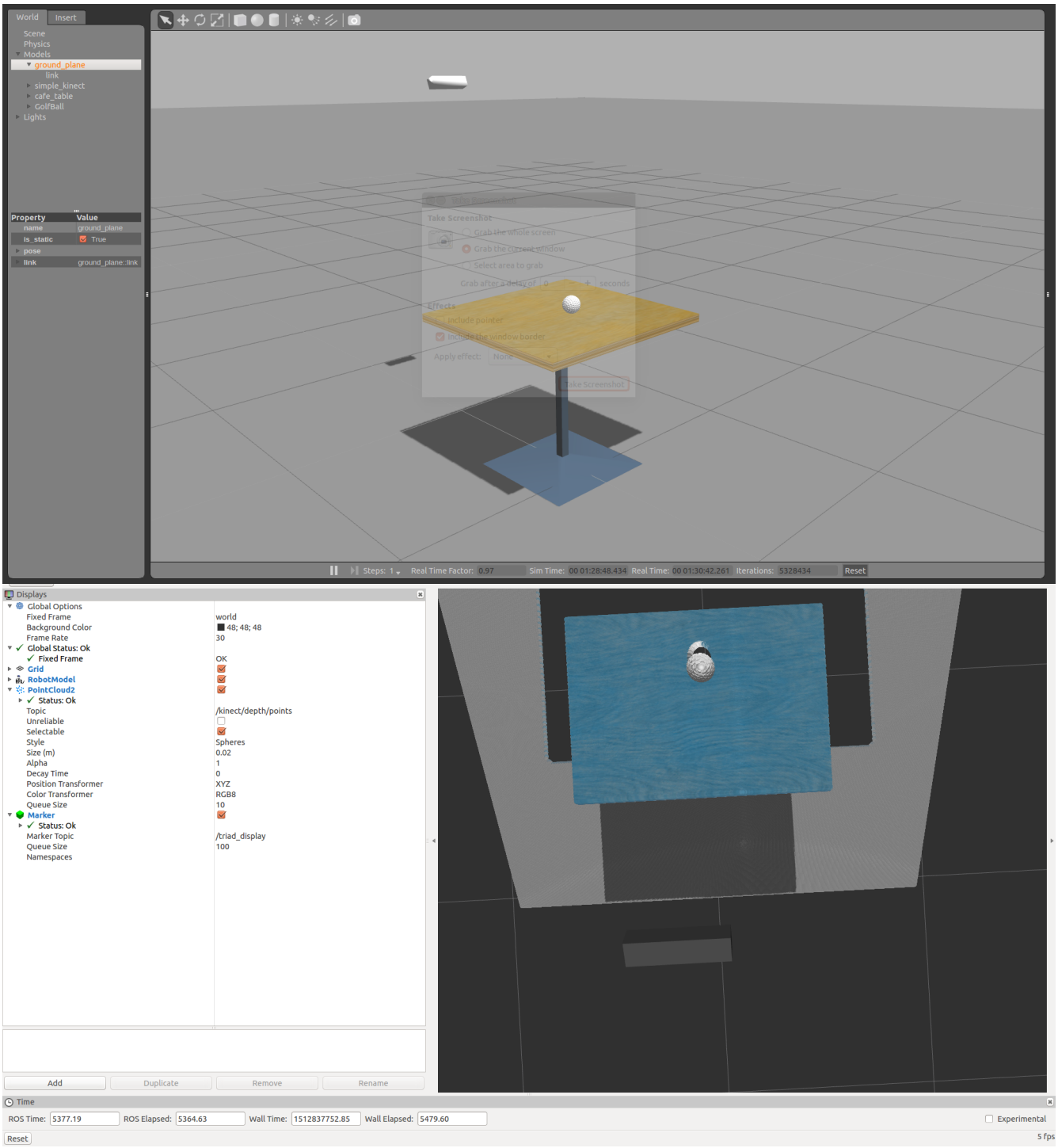
There has been a recent developments within neural networks in application in everything that human beings do everyday. From self driving cars, to medical imaging, to even voice recognition, and object recognition. Recently there has been an interest in using these deep learning methods into a 3d object recognition area[4], in order to be able to recognize objects from the shape of an object rather than the colors of it in an rgb values of an image[4] are of a new area of interest. If we could get an object to recognize something effectively using depth we could put this ability into new robotic devices, such as a self driving cars, manufacturing ,and surgical robots.

Within this project, we are interested in reworking a novel application of these neural networks.[4] While these 3d object detection methods are great, we are interested in speeding up the data acquisition process . Using ROS [2], we hope to create a simulation environment, that will allow for easy data collection of different models. With this easy data collection, we can then be able to learn any model that we want with ease of use. By trying to create a simulation of these depth objects, can we get a good learned model? This is the main goal of this project, to be able to take multiple 3d objects and be able to recognize them easily in our network. For creating the models we want to train we used the Tensorflow library [7][3]. This allowed for us to create an easy to use platform to train against different models that we are interested in.

We used the Google cloud platform in order to train our networks on.[9] This allowed us to gather the necessary computation power needed to run the deep learning networks that we have learned about in the class. The platform also allows for easy use of the Tensorflow library[7][3], for stuff like the MNIST dataset that we have already covered in class.

Finally, we also trained a couple of simpler models on the dataset. We trained a 2 layered network, this gives us a better understanding about how much more gain we get by using the convolution network over a model like the linear model.

ROS



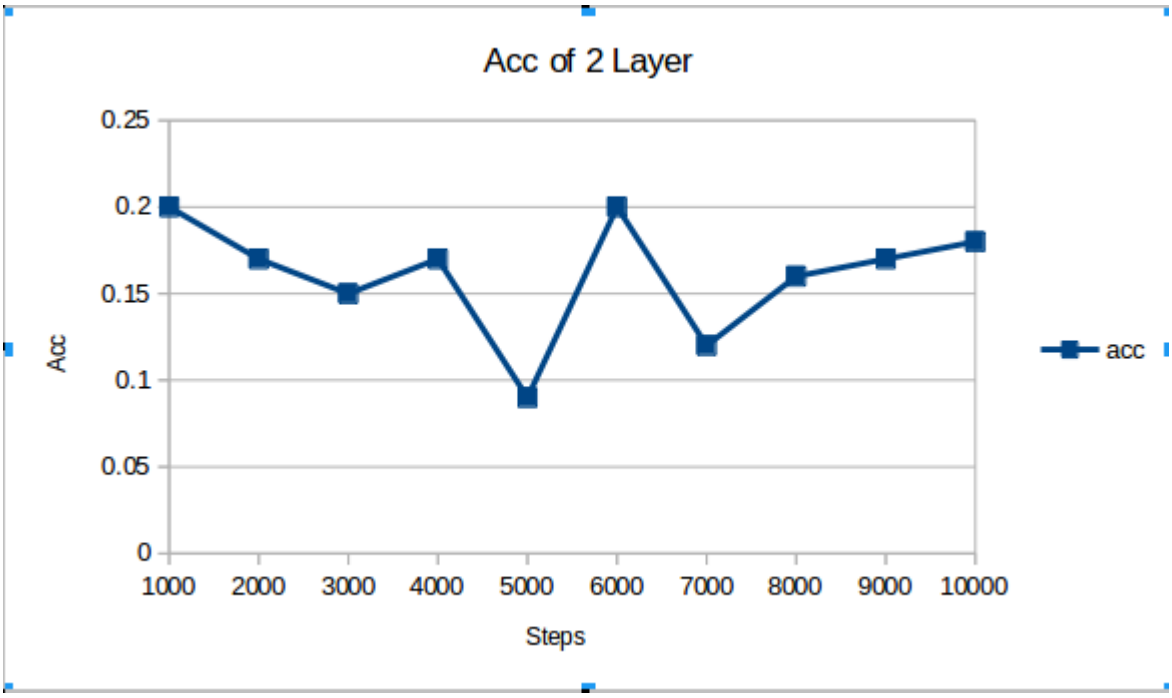
First image above is the ROS environment that is being used to drop the object, the second is the simulated camera view that is taking the phtos of the images.

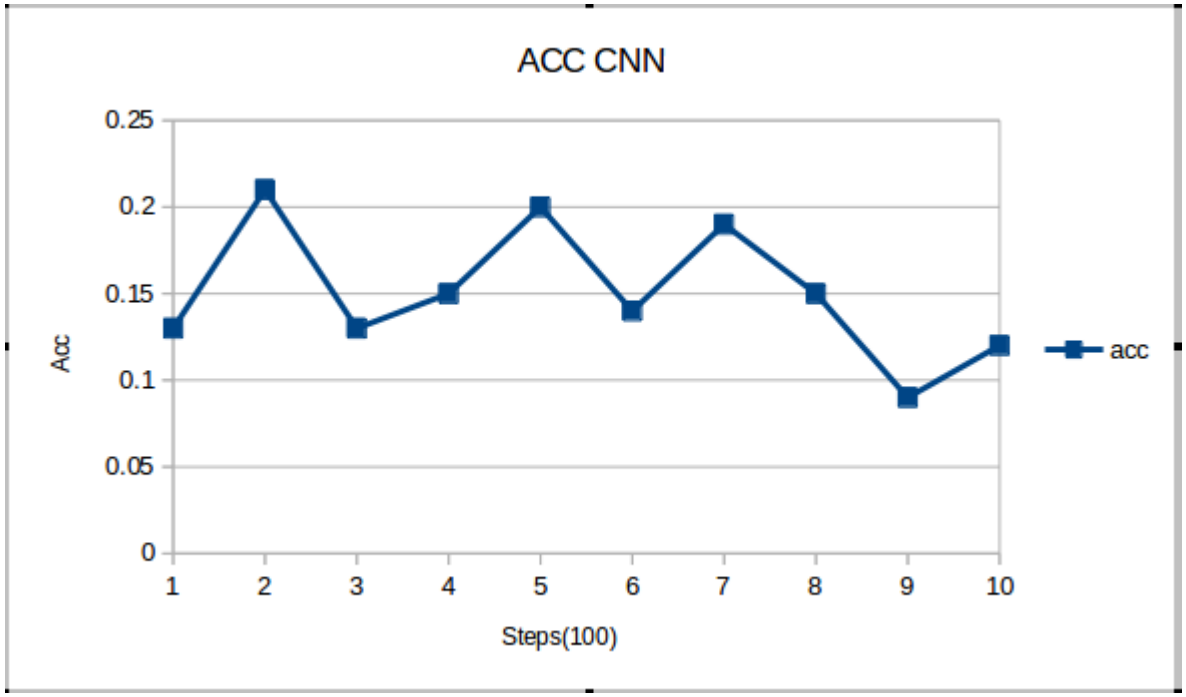
Within this project we used ROS in order to create the data that we trained our models on. In order to do this we had to create an environment to gather the data that is used in this project. The first thing we had to do is gather an STL model for the model that we are interested in. Taking this STL model we can drop it into the Ros environment like below.

In order to do this it is important to keep in mind how far away we drop the object, how large the STL model is, and where the simualted camera is. Ultimately, we want to make sure that the data we get is a good sample size of the different objects we have in the sample size.

After dropping this object we are left with the camera's view of the object. We use a simulated camera of the 3d model and have it up above the desk like in the image below. We then take that image and then transform it from the camera's viewpoint to the world's viewpoint. From this, we can then be able to have a standard frame for which we can feed into our machine learning models. With this image we can get our representation of the 3d object that we are interested in for our model to learn from. We can create a process that we can the repeat by, moving the object, taking a picture of it, and then moving it again. Doing this over a long times gives us a dataset that we need in order to have our models train on.

Results





As we can see from these results, the effectiveness of the learning algorithms are a little bit better than random chance. So it appears to be that the models are having a hard time figuring out the correct, objective function. While it is sometimes doing better than just guessing purely randomly, overall it appears to be that the network is having a hard time to discern the different objects. The convulsion network was able to at it's peak able to figure out the different objects better than random chance. So then there is still a lot of work to be done in trying to figure out good model representation for the object recognition task. While overall the two layer network had a better average over the convolution network. This could be that simply the 2 layer network was allowed to train over a longer time than the

The reasons for these difficulties can be multiple, it could just be hard for the program to distinguish between the different classes. It could be that because there are so many features in our dataset, the learners are having a hard time finding the objective function. It could also be that with more computing power, that the learners could easily find the objectives needed for the function to work. Finally, it could simply be that there is not that much of a difference between the models, and that the learner is simply always going to get confused between the different objects.

Conclusions

Within this paper we explored how to use deep learning methods to do object recognition. Overall, these methods failed at recognizing the objects simulated in the dataset. This could be for a multitude of different reasons. From the images not being correct, to the

models needing more time to being fine tuned, to even just simply needing more computation time in order to get the models working. Hopefully, in the future we will be able to make it easier for the models to learn easier, and be able to get better results for object recognition.

References

[1] (http://cs231n.github.io/convolutional-networks/)

[2] A Systematic Approach to Learning Robot Programming with ROS Wyatt Newman

[3] TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems
(Preliminary White Paper, November 9, 2015)
Martın Abadi, et al.)

[4] DEEP-LEARNING APPROACHES TO OBJECT RECOGNITION
FROM 3D DATA by ZHIANG CHEN

[5] https://www.cs.toronto.edu/~hinton/absps/NatureDeepReview.pdf

[6] 3D is here: Point Cloud Library (PCL) Radu Bogdan Rusu and Steve Cousins

[7[]https://agray3.github.io/2016/11/29/Demystifying-Data-Input-to-TensorFlow-for-Deep-Learning.html

[8] https://cloud.google.com/docs/