Object Localization with CNN

Holly Liang, Zixuan Zhou, Yuwen Zhang

Advised by Patrick Cho, Simon Kalouche

Introduction

- When training a robot to perform what may seem like simple tasks such as pick and place, a difficult step is to visually perceive the object and extract the localization information. Our goal in this project is to tackle this important task using synthesized RGB and depth images from the robot's depth camera (eyes)

Data Preprocessing

- We explored the 1,8000-images dataset generated by Unity3D simulation platform, provided by *Stanford AI Lab* and *Deepdroid*. In this dataset, each sample is made of a **299x299x3 RGB** image and a **299x299x1 depth** image.
- On each image, there are 10-20 different items, and all of the items belongs to a 55 items itemset. Currently, we are training the model to localize an old radio, shown in Figure 1(a) and (b).
- The dataset was processed using **domain randomization**, which adds different simulated light sources and hues to the generated images. As such, our model can better fit to images take in real-life situations in the future
- In one of our models, we processed the depth image using **color-map**, to convert it into RGB (299x299x3) image.

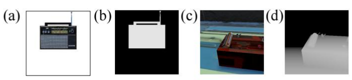


Figure 1: (a). RGB image of target item, (b). depth image of target item, (c). RGB image sample from dataset, (d). depth image sample from dataset.

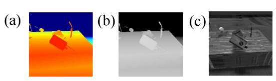
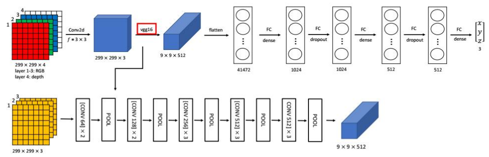


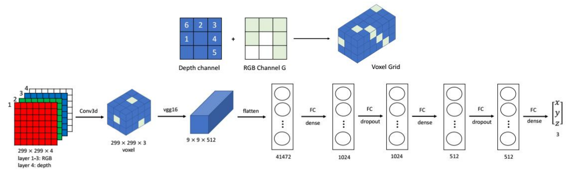
Figure 2. (a). color-mapped depth image, (b). original depth image, (c). color image

Models

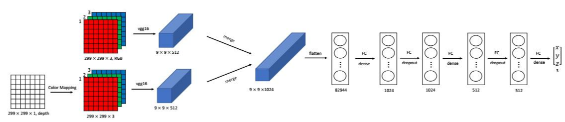
- **Model1: 2D VGG Model**. We input a 299x299x4 image file, run through a convolution layer with filter size = 3, add padding and reshape to feed it into the Keras VGG16 model. After VGG16, we added a few more fully-connected layers to generate an output of estimated (x, y, z) location of the target object.



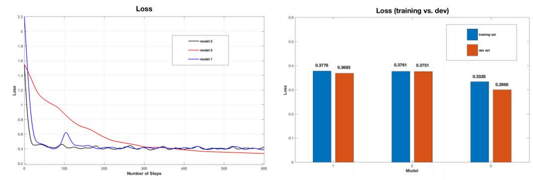
- **Model2: 3D Voxel + VGG Model.** We preprocess the depth information to produce a spatial 3D voxel representation combining depth and RGB information. The 3D voxel representation is created with the same height and width as the original image, and with depth determined by the difference between the maximum and minimum depth values found in the images. We then quantize our depth values into 10 intervals and feed our input into VGG net.



- **Model3: Color Mapping + Parallel VGG Model.** We color-mapped the depth information into an RGB (299x299x3) image using JET color matrix. In this way, depth information is represented as colors and hence can be picked up by VGG. The regular RGB image is fed into a different VGG. We then feed the output of both VGGs into several fully-connected layers to generate an output of estimated (x, y, z) location of the target object.



Results



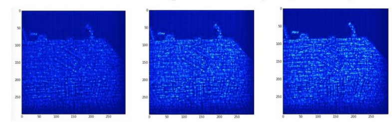
Error Analysis

•**2D VGG model:** we only apply a single convolutional layer to convert the depth information, so the error is relatively large.

•**3D Voxel VGG model**: We quantize our depth information into 10 intervals and the error mainly comes from the depth discretization

•**Color Mapping + Parallel VGG model**: This model mostly reserves the depth information through color-mapping, so it performs better than other two models.

Attention Map of Final Layer (Model 2)



Training Strategy

- We split the data set into three part, including 80% training set, 10% validation set and10% test set. All data sets contains depth and RGB images.
- All three models used Adam update method with batch size equals to 32. The learning rate is 0.0001 for the first 7 epochs followed by 0.00001 towards the end to obtain both high efficiency and good convergence.
- We applied dropout method as regularization technique with drop probability equals to 0.5.

Conclusion

- We implemented object localization based on three different models. The experiment results show that model 3 outperforms the rest and achieves 18.67% and 20.15% relative improvement in test error, compared to model 1 and model 2.
- To improve this result, we need to 1). generate more data. or 2). add boxing and pixel wise label to our dataset.