4/4/2020 - Eduardo Davalos

**THESIS WORK REPORT**

**1. BACKGROUND INFORMATION**

The goal of this thesis (still seeing if this is possible) is to create an RGB category-level 6D pose estimation neural network (NN). The field of computer vision is neatly cut into different sections by task, such as object detection, image segmentation, face recognition, and pose estimation. Fig. 1 illustrates where this thesis fits along the body of literature already present.

--- Computer Vision

|--- Pose Estimation

|--- Rigid Body 6D Pose Estimation

|--- RGB Input Only

|--- Category-Level 🡨This is the location of this thesis’ work

Figure 1: How this Thesis fits along with the Literature

6D pose estimation is the estimation of two main attributes of an object: translation and rotation. Note that 6D pose estimation is different from human body estimation, for 6D pose estimation only addresses rigid bodies. Each attribute requires distinct 3D information, therefore resulting in 3D translation + 3D rotation = 6D pose. In 6D pose estimation, rotation and translation are determined concerning the camera’s position and angle.

A picture containing table

Description automatically generated

Figure 2: Rotation and Translation Visualized

1.1 CATEGORY-LEVEL VS. INSTANCE-LEVEL

One of the important distinctions in 6D pose estimation is category-level and instance-level pose estimations. Instance-level pose estimation focus on a single instance of an object’s category. Category-level pose estimation focus on an entire object’s category, not just one instance of it. For example, an instance-level pose estimation NN would only be able to estimate the pose of the single camera on the left side of Figure 2, while a category-level pose estimation NN would be able to estimate the pose of all the cameras on the right side of Figure 2.

A close up of a camera

Description automatically generated VS. A close up of a camera

Description automatically generated

Instance Category

Figure 3: Instance-Level vs. Category-Level

Currently, for testing purposes, I am using [NOCS\_CVPR2019](https://github.com/hughw19/NOCS_CVPR2019), an RGBD Category-Level Pose Estimation NN, to achieve the task of category-level classification. One of the main ways that NOCSnet achieves category-level classification is by its use of NOCS, Normalized Object Coordinate Space, to represent an object’s category with a single 3D model. This 3D model is constructed by averaging multiple instances of an object to achieve a form, like Plato’s theory of form, that represents the object’s general properties and attributes. Figure 3 shows an example of a category’s 3D model, in this case, a camera.

A picture containing light

Description automatically generated

Figure 4: Category-Level 3D of ‘Camera’ Category (NOTE: you can move the second 3D model)

Additionally, instance-level applies multiple restrictions to the NN. Instance-level requires that the objects detected have been scanned and reconstructed into a 3D model for every object it detects, while category-level can simply construct generalized models from large CAD model datasets. In general, the category-level NN is more scalable for detecting more objects.

1.2 RGBD vs. RGB

The main difference between this thesis’s NN and NOCSnet is that NOCSnet requires depth as an input. My goal is to remove this requirement. Since NOCSnet’s representation of objects within the NOC Space is normalized, obtaining the true scale of the object is difficult. This is why depth is used by NOCSnet. Therefore, this thesis’s NN intends to incorporate depth prediction into the network’s design to remove the depth input requirement.

The neural network I am using to create depth images is [DenseDepth](https://paperswithcode.com/paper/high-quality-monocular-depth-estimation-via). Its input is an RGB image of size (480, 640) and its output is a depth image with size (240, 320). Simply the image is scaled up to match the size of the input.

**2. CURRENT SITUATION**

The main challenge that I am trying to do is format the depth generated by DenseDepth to use it for the NOCS network. By formatting, I am referring to changing the size, encoding, and scaling of the depth image to match the excepted original depth images.

Main Differences (that I know thus far)

DenseDepth Depth

* Size: (240, 320)
* Encoding: float32
* Scaling: ≈1/14
* Trained on NYU dataset

NOCS RGBD Camera Depth

* Size: (480, 640)
* Encoding: uint16
* Scaling: 1
* Captured for the REAL dataset

The test I conducted is to evaluate the performance difference between camera/source depth and DenseDepth/generated depth. The diagram below illustrates how I was able to compare these two different depth schemes.

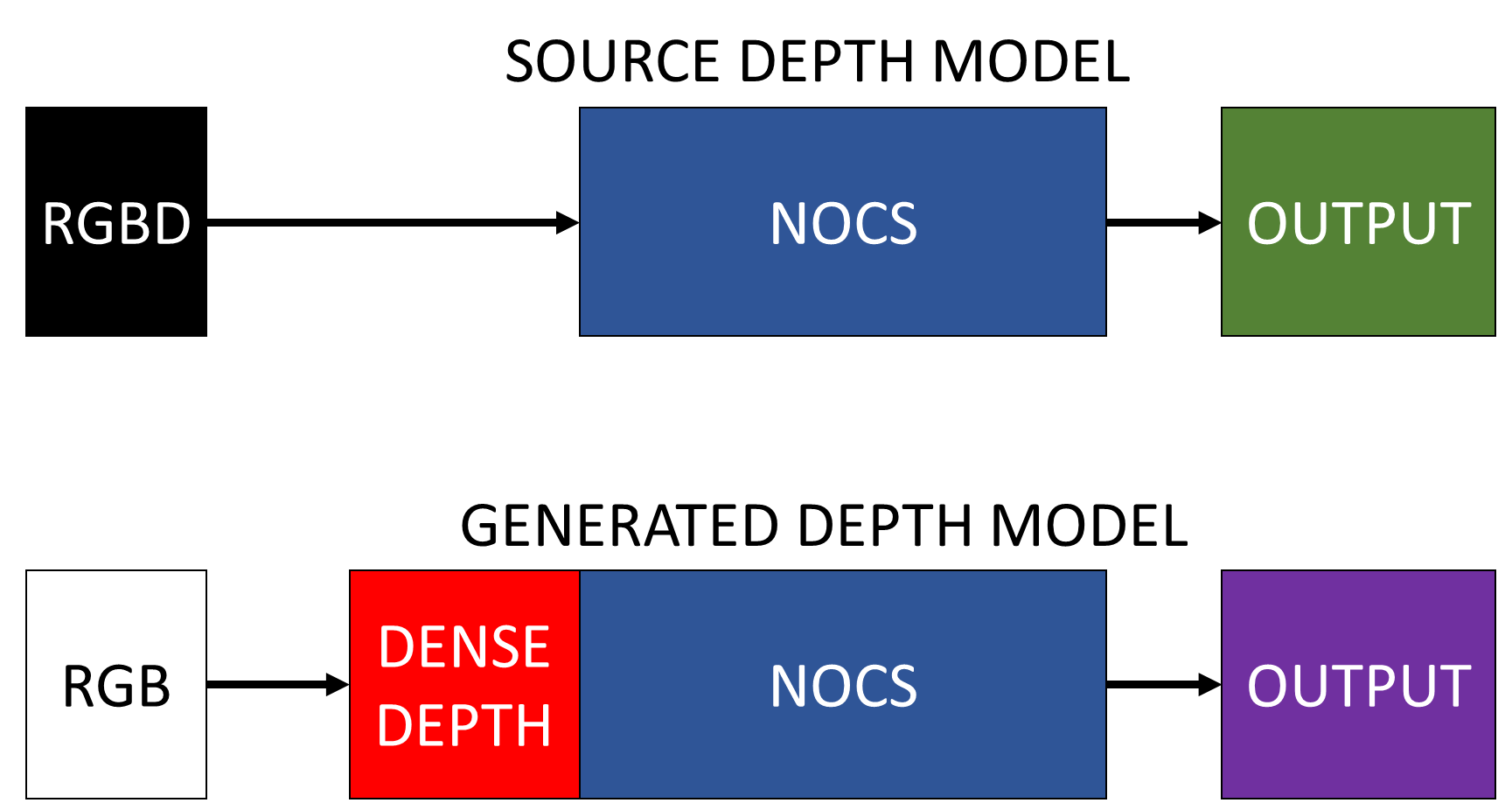
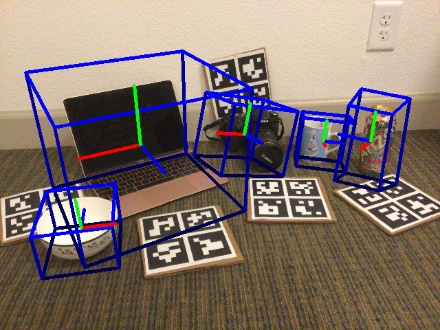
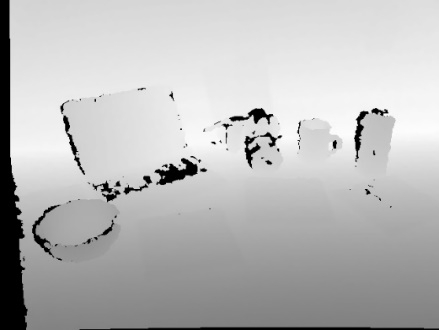


Figure 5: Source vs. Generated Depth Models

In the following page, I will show multiple images of various experiments to explain my investigation for how to make the depth generated by DenseDepth functional with the NOCS network.

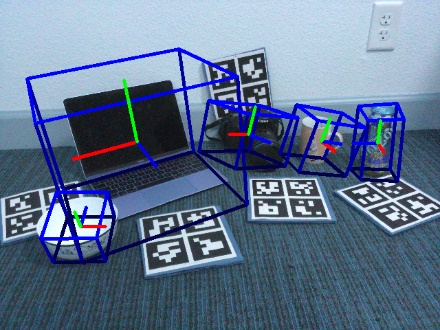
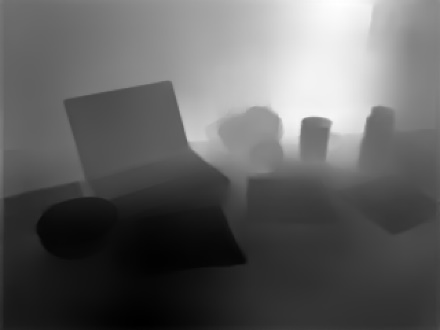


C

B

A

Figure 6: NOCS Input and Output. (A) = RGB input image, (B) = grayscale of source depth image, and (C) = output pose, including the rotation and translation.



B

A

C

Figure 7: NOCS Input and Output with DenseDepth, (A) = RGB input image, (B) = grayscale of the generated depth image, and (C) = output pose, including rotation and translation.

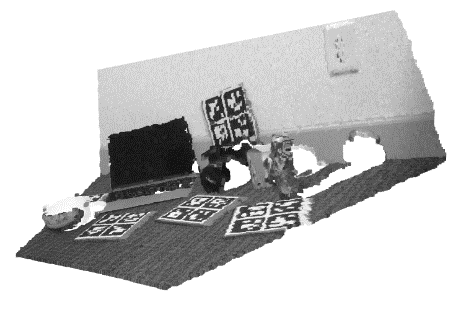


Figure 8: Point Cloud 3D Visualization of source RGBD image

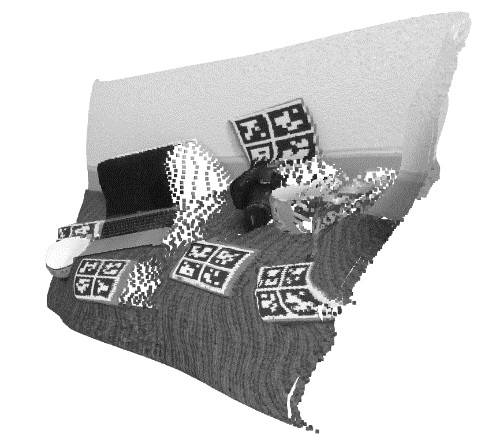
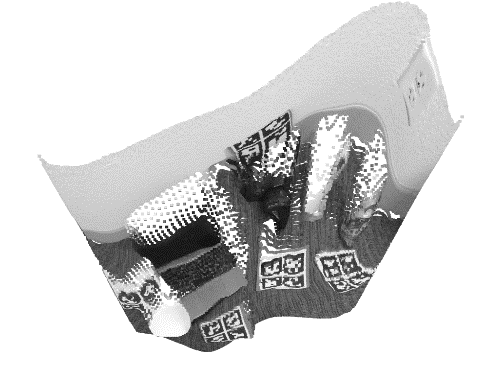


Figure 9: Point Cloud 3D Visualization of generated RGBD image

At first glance in Figures 1 and 2, it seems that the depth generated by DenseDepth contains more information, especially around object’s edges. Additionally, it contains more contrast between the foreground objects and the background, leading to the speculation that DenseDepth data was of higher quality. The point cloud illustrations in Figure 3 and 4 tells us otherwise. I realized that DenseDepth is good at capturing small details and contrast between objects, but DenseDepth is not capable of detecting the correct depth values for the background. This point will be elaborated more in section 3 (future work and conclusion).

To obtain a better understanding of the performance difference between the source and generated depth models, I determined the Average Precision (AP) of all the objects and the mean Average Precision (mAP) of all the categories, along all possible 3D Intersection over Union (IoU) thresholds. This 3D IoU is based on the predicted and ground truth 3D bounding boxes. The mAP and AP (explained [here](https://medium.com/@pds.bangalore/mean-average-precision-abd77d0b9a7e)) are a calculation of how many times the neural network correctly or incorrectly predicts the location of the 3D bounding box. The 3D IoU is the threshold for when a 3D bounding box is considered correct or incorrect, and a higher AP along a higher threshold implies better performance.

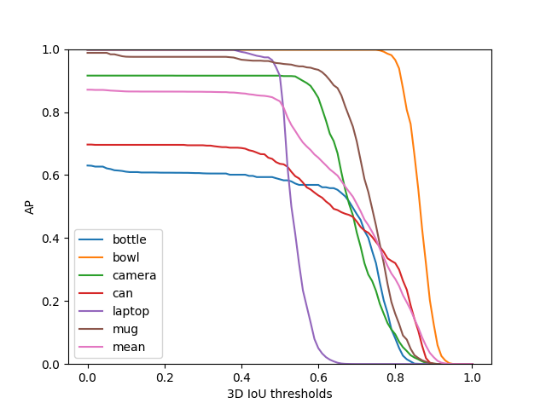
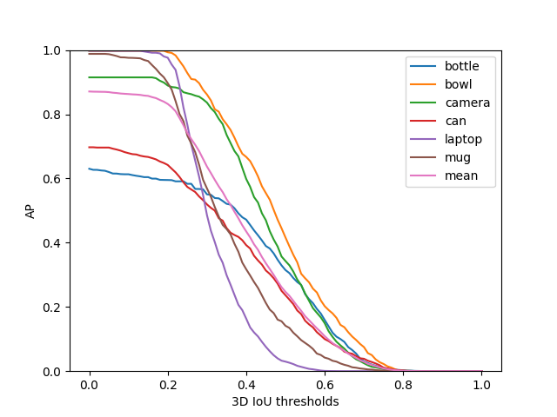
 

Figure 10: AP Metric for (left) camera depth and (right) DenseDepth depth



Figure 11: AP Metric for camera (source) depth and DenseDepth (generated) depth

It is clear from the contrast between the plots in Figure 10 that the performance of the NN is significantly lower with generated depth. The 3D bounding boxes created with the generated depth lack accuracy when the 3D IoU threshold is high. Furthermore, if you look at Figures 12 and 13, we can also see how the translation and rotation of the pose estimation perform differently in the two depth models.

The translation from the generated depth outperforms the translation metric from the source, likely due to the continuous spectrum of the generated depth. However, the rotation metric from the generated depth is just terrible, likely due to the highly stretch depth values from the generated depth image.

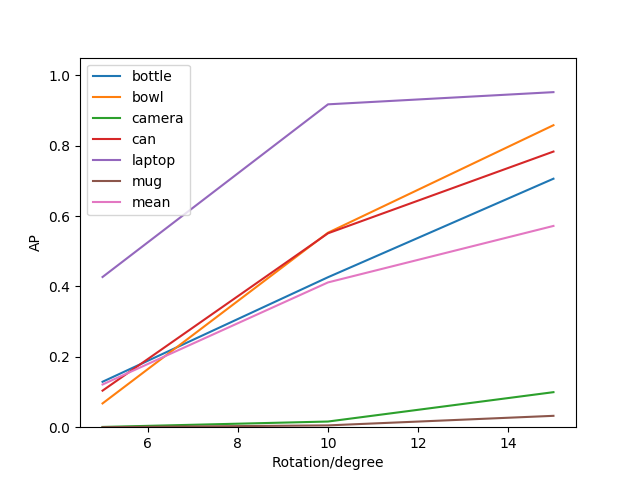
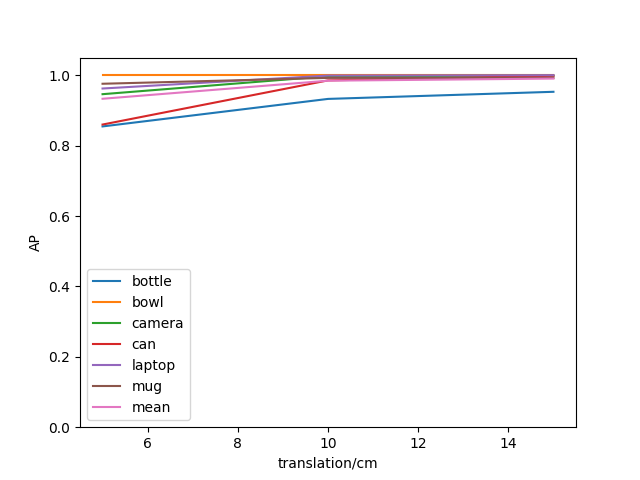
 

Figure 12: AP for translation (left) and rotation (right) for source depth

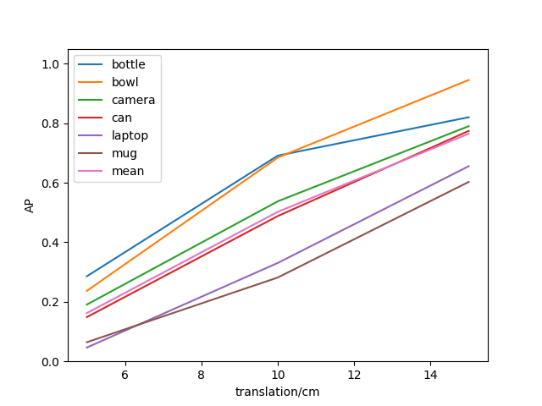
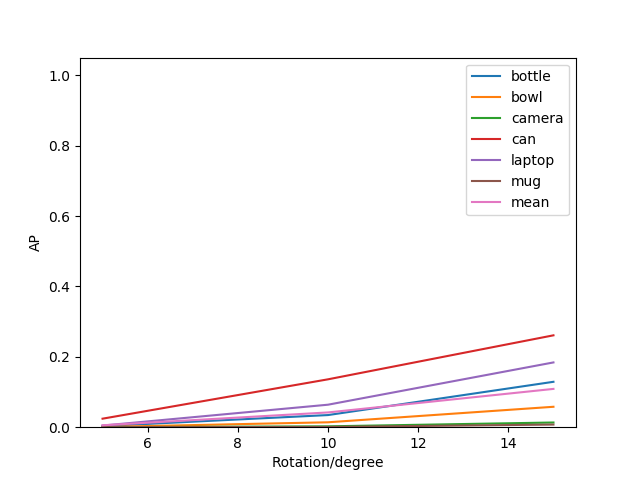
 

Figure 13: AP for translation (left) and rotation (right) for generated depth

I speculated that this can be attributed to the fact that DenseDepth was trained on the NYU dataset, a dataset that is mostly composed of medium to large indoor landscapes. There is a significant domain gap between what DenseDepth has been trained to do and the input that I am feeding into DenseDepth. In the following figure, I will illustrate the difference between the NYU (training material of DenseDepth) and the REAL (NOCS input) dataset.

****

Figure 14: NYU dataset example images



Figure 15: REAL dataset example images

**3. FUTURE WORK & CONCLUSION**

After evaluating the performance of the generated depth model, I see that there is significant work ahead to optimize and improve the generated depth model to be competitive. Still, it is important to consider that an RGB category-level pose estimation NN still does not exist; therefore, there is no fierce competition to outperform. I believe this can be a proof of concept or a pioneer in this section of the computer vision and machine learning field.

My intention for the upcoming weeks is to do the following:

* Train DenseDepth neural network with the REAL dataset and to re-evaluate the performance of the generated depth model
* Utilize various other depth-generating models to ensure that DenseDepth is not an outlier when it comes to generating depth.
* Further investigation of the NOCS model to determine how possible it is to append the task of depth generation.

Please do let me know if something was not clear or sufficiently explain. Also thank you for your time.