CSE291 HW3: Pose Estimation

Ashish Maknikar
UCSD
San Diego
amaknikar@ucsd.edu

Abstract

The problem requires the estimation of the pose of the objects in test data set in the world frame using the RGB image, depth image and image metadata such as the intrinsic camera matrix without the given semantic segmentation masks for the scenes. We must use the above information to detect the poses of the objects in the test frames.

1. Introduction

For this assignment, we do not have the semantic segmentation of the objects in the test scene. Various methodologies such as PVN3D [1] and DenseFusion [3] provide frameworks for pose estimation using Deep Learning methods combined with ICP. We can use PointNet [2] for obtaining the feature vectors of the point cloud and a UNet architecture for semantic segmentation.

2. Method

We use a custom UNet to perform semantic segmentation. The semantic segmentation is followed by a ICP to obtain the pose estimate.

2.1. Semantic segmentation

We use an encoder-decoder UNet as the one displayed in Fig. 1 architecture to segment the image. In light of limited GPU resources, the input image is down scaled by a factor of 4. The output semantic segmentation mask the we get is then up sampled to the original size.

2.2. Pose Estimation

We use an ICP pipeline to obtain the pose estimate for our results.

2.2.1 Training Processing

The point cloud is extracted from the data for both data sets using the depth image and intrinsic camera matrix. The segmentation mask for the training data is then used to segment out the object point clouds that were extracted. The extrinsic matrix in the metadata can then be used to transform this data from the camera frame to the world frame. A combined view of all the point clouds obtained by this method should intersect to display the object as shown for a spam can in Fig. 2

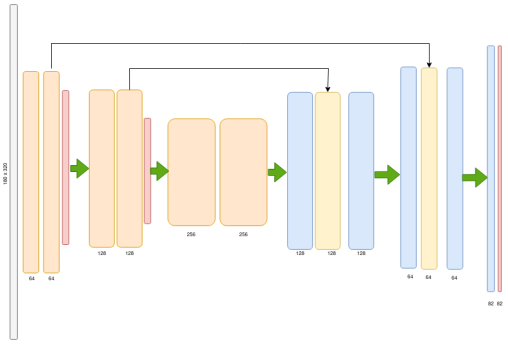


Figure 1. UNet structure for segmentation

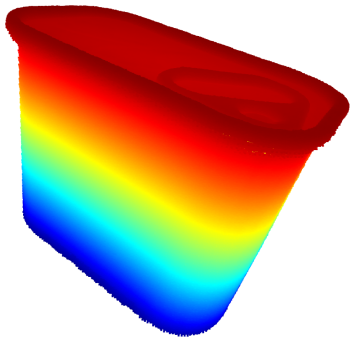


Figure 2. Combined point cloud of a can of SPAM

2.2.2 Extracting the object point cloud

Saved weights are used to segment out and process the object point clouds from the test images and test depth files into the world frame.

2.2.3 Pose prediction

For a particular object in a test scene, we loop over all the point clouds of that object in the world frame at origin from the train set, setting them as the source, and the object point cloud from the test scene in the world frame as the target for the ICP algorithm. We get $N_{train\_objects}$ number of poses and we choose the best pose amongst these factoring the correspondence and the RMSE error.

2.3. Losses

A loss of Cross Entropy is used to train the semantic segmentation UNet. No loss is required for ICP.

3. Experiments and Testing

3.1. Data

To conserve GPU memory, we down-sample the 720x1280 image to 180x320. The output mask is up sampled back to 720x1280 using the skimage resize function.
We take every 16th point in the point cloud for finding the pose in relation to the world pose point cloud of the training data scenes.

3.2. Training details

We train the UNet with an Adam optimiser at a 0.004 learning rate and cross-entropy loss.
The ICP is trained by sampling every 16th point with a 0.2 threshold.
We have $N_{train_objects}$ number of poses to choose from after ICP. The ICP algorithm by Open3D returns a rmse_inliers error, number of corresponding points among the point clods and the pose. Using merely the minimum rmse_inliers error causes many results with zero correspondence getting picked up. To counter this, our validation performance improves by dropping all zero correspondence results and extracting the set of top 20% correspondences, from which we choose pose with the minimum rmse_inliers error as the result.

3.3. Ablation studies

We also try a UNet that gives a 180x320 segmentation mask as shown in Fig. 3a and a 720x1280 mask as shown in Fig. 3a. The larger mask does not give any substantial improvement on the smaller and upscaled mask.
As known from the previous assignment, the performance of ICP peaks approximately at the point taking every 16th point.
We further attempt to ICP refine the result by performing ICP on the obtained transformation matrix with 1000000 points of the combined point cloud. The attempted post ICP refinement as shown in Fig. 5a performs poorly compared to the partial point cloud shown in Fig. 5b resulting in a severe decrease in the pose accuracy.
NOTE: There has also been an unsuccessful attempt to implement Dense Fusion to use a neural network for pose prediction. The code(incomplete) is included in the HW3_Datahub file. DenseFusion generates feature vectors for both the point cloud and the segmented RGB image and combines them to generate the pose estimate.

3.4. Visualization

As explained in the ablation studies above:

- UNet that gives a 180x320 segmentation mask :Fig. 3a
- UNet that gives a 720x1280 mask : Fig. 3a.
- ICP results(gives the best results): Fig. 5b
- ICP with refinement : Fig. 5a

3.5. Performance: username:gabbar

The best performance is given by the Unet segmentation with down scaled 180x320 mask and ICP without refinement(**entry: jai**) with a pose_acc_5deg_1cm of 0.6573. The contexts of the other submissions are:

- veeru: Using the UNet with 720x1280 outputs
- thakur's hand: Using post ICP refinement.

4. Code Files

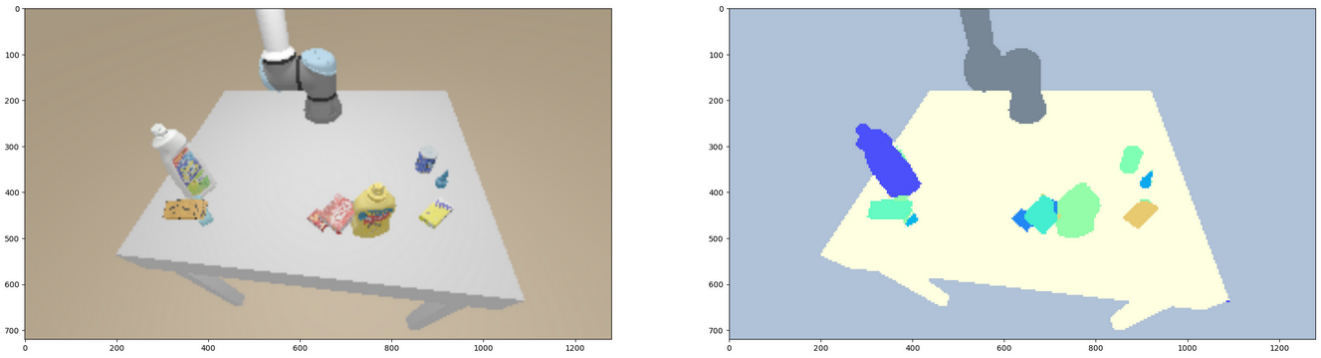
- CSE291_HW3_Colab: Training the UNet for 180x320 segmentation output
- HW3_Datahub: Training the UNet for 720x1280 segmentation output and incomplete DenseFusion code.
- HW3_local : Generating the training world frame and origin based point cloud dictionary
- test.py: ICP for pose generation

References

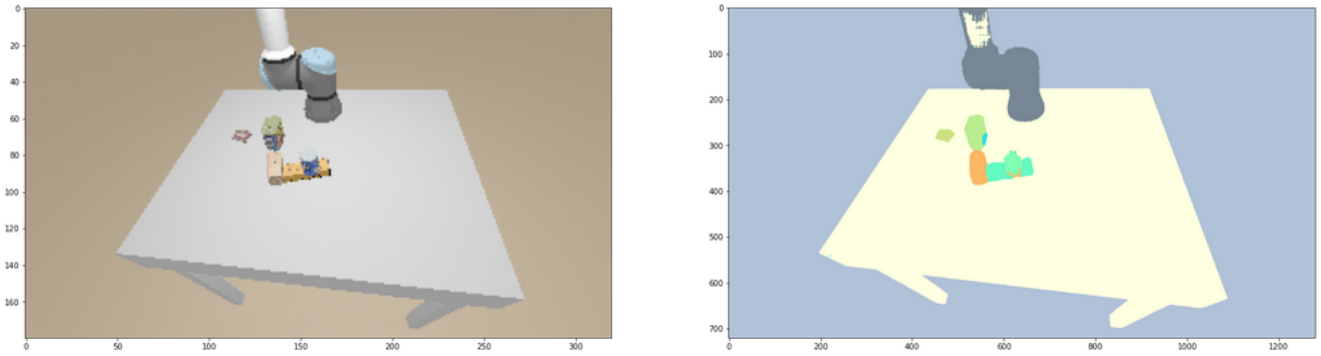
[1] Yisheng He, Wei Sun, Haibin Huang, Jianran Liu, Haoqiang Fan, and Jian Sun. Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation, 2019. 1

[2] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2016. 1

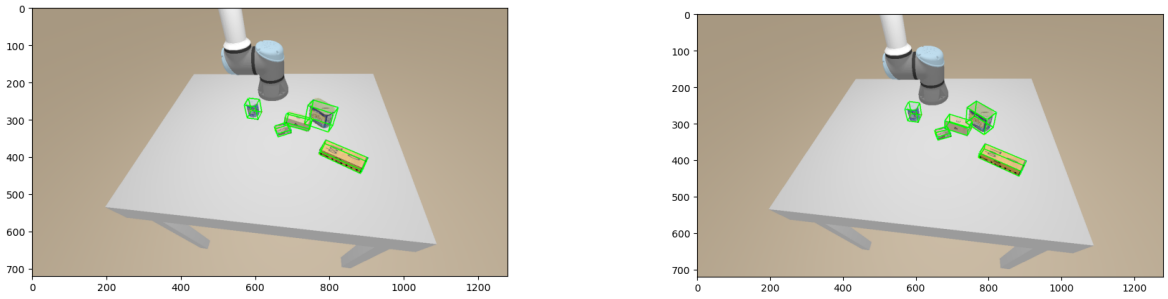
[3] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion, 2019. 1



(a) Segmentation on test image with 180x320 output



(a) Segmentation on test image with 720x1280 output



(a) Bounding Boxes with refinement

(b) Bounding Boxes without refinement