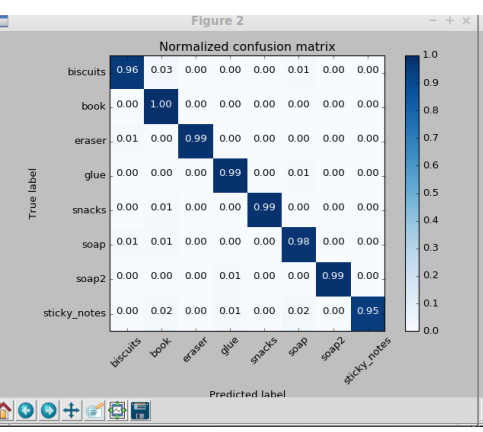
Writeup. Project 3. 3D perception

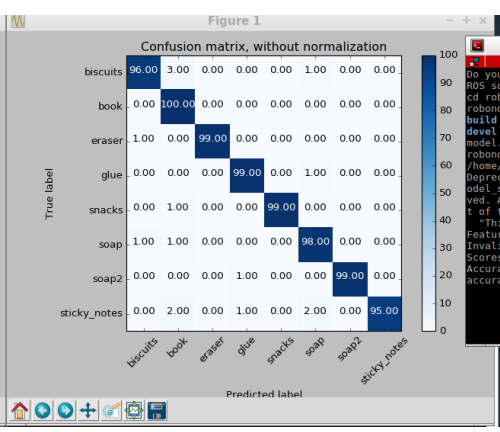
1. Exercise 1,2, and 3
    1. The pcl_callback() has three steps
        1. Filtering & RANSAC & Segmentation
            1. Read data: pcl_data = ros_to_pcl(pcl_msg)
               // ros input to pcl for image processing
            2. **Voxel Grid** down sampling to reduce # points to process
               vox = pcl_data.make_voxel_grid_filter() with LEAF_SIZE = 0.01 (1% from X, Y, Z dim), then it will create cloud of points from the object
            3. **Statistical filter**: to reduce noise from the image, stat_filter is applied based on the tutorial. There is no variable should be determined empirically.
            4. **Passthrough filter**: the most critical filter in this project. It should limit the region of interest from the image captured in camera.
               From the exercise, we create Z-axis filter (0.6~0.8). Min value is set to avoid capture edge of table (0.6) and Max value is set to avoid look at the floor and consider as “BOOK”
               In actual project, Y axis should also be cropped (-0.4 ~ 0.4), otherwise robo-arms and two bins are captured and considered as “BOOK”. I guess all of solid plate & rectangular object could be considered as “BOOK”. So we need to crop the image in both Z and Y axis just look at the objects on the table
            5. **RANSAC filter**: Same as exercise 1, RANSAC place filter is applied and max_distance is set as 0.01 to remove the table. After segmentation, Outlier become objects
        2. **Euclidian Clustering** to separate clouds into different objects. We don’t know what this object is yet. I followed the procedure from Exercise 2 but need to tune the parameters such as Cluster tolerance, Min ~ Max cluster. I found these values from the Exercise 2.
           ec.set_ClusterTolerance(0.02), ec.set_MinClusterSize(10), ec.set_MaxClusterSize(25000)
        3. Features extracted and SVM trained
           To guessing well, I captured features 100 times (a single object is rotated randomly 100 times) and image is captured in HSV format. In SVM training, it achieve 0.98 score

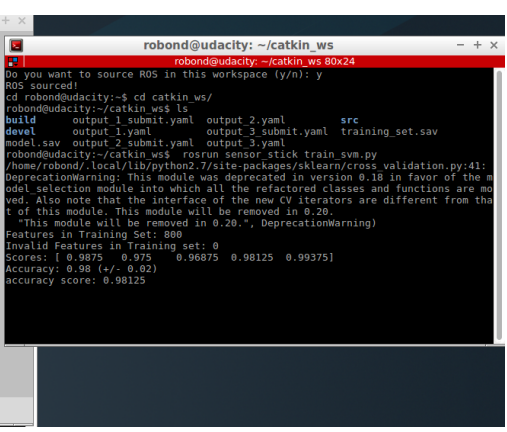
```
for model_name in models:
    print("NEW MODEL")
    spawn_model(model_name)
    num_range = 100
    for i in range(num_range):
        if (i%10 ==0):
            print "%d/%d" %(i, num_range)
        # make five attempts to get a valid a point cloud then give up
        sample_was_good = False
        try_count = 0
        while not sample_was_good and try_count < 5:
            sample_cloud = capture_sample()
            sample_cloud_arr = ros_to_pcl(sample_cloud).to_array()

            # Check for invalid clouds.
            if sample_cloud_arr.shape[0] == 0:
                print('Invalid cloud detected')
                try_count += 1
            else:
                sample_was_good = True

        # Extract histogram features
        chists = compute_color_histograms(sample_cloud, using_hsv=True)
        normals = get_normals(sample_cloud)
        nhists = compute_normal_histograms(normals)
        feature = np.concatenate((chists, nhists))
        labeled_features.append([feature, model_name])
```







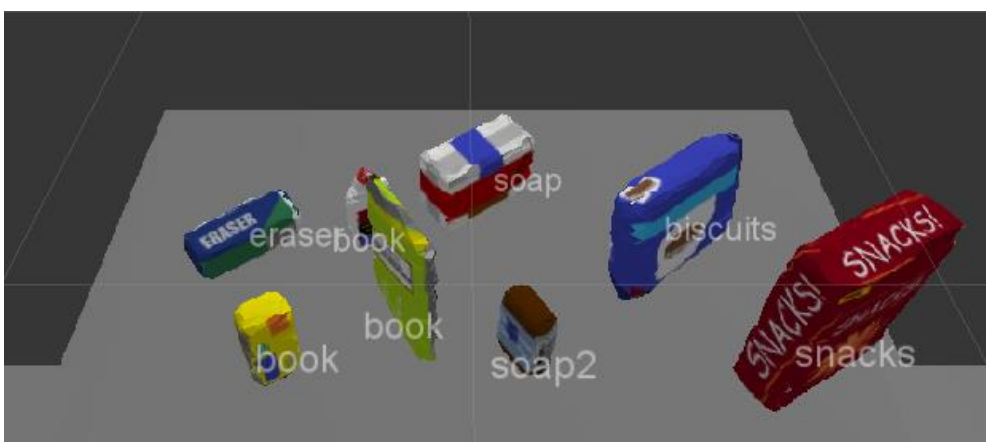
Writeup. Project 3. 3D perception

2. Pick and place
    2. For three different scenarios.

Output 1: 100%

Output 2: 80% (Glue is captured as book)

Output 3: 75% (Glue and Sticky note as captured to book)



Since it capture glue and sticky note to books, I need to consider multiple items can be considered to the same object repeatedly.

In other words, although one object from the image is considered as book and delivered to the correct bin, it may not be actual book. In the code, two nested loop is designed as below

```
For found_object in the image
        For target_object in the target_object_list // should be found
                if target_object_name == found_object_label
                        FOUND !!
```

When one object is found, I tried to skip comparing it for next found_object (example, if book was founded in previous step, I will skip book from target_object_list). However, since multiple objects (glue, sticky note, and book) are considered as book, I need to check new_found_object with book several times because I can't guarantee the previous found object is actually book.

Based on the index of target_object_list, find ['group'] to determine arm & place_position.