Occlusion Detection in a Cluster of Objects using RefineNet

Lab Cognitive Robotics

Tridivraj Bhattacharyya

tridiv.b@uni-bonn.de;Matriculation Number: 3035538

Abstract Classifying Objects according to their classes might not be enough while segregating them from a cluster. It is often necessary to determine which object lies on the top so that it becomes easier to make decisions for tasks like picking. Here we implement an algorithm for Occlusion Detection alongside Semantic Segmentation using RefineNet and evaluate it on the datasets used for the Amazon Robotics Challenge 2017.

1 Introduction

Object Classification is an active area of research in the field of Computer Vision. This is challenging especially in cluttered or randomly placed objects. It is important to identify and classify objects from clutter to carry certain tasks like sorting or picking. However, in such cases only classifying objects as per their class may not solve the problem efficiently. Occlusion Detection becomes useful here. It would be helpful if the objects lying at the very top are prioritized during picking.

Every year, Amazon holds a Robotic Bin Picking challenge. In the challenge, the bot is required to sort through a clutter of objects, pick and place them in proper order. Our system has been developed based on the criteria of this competition. The process to classify the objects had already been implemented with Semantic Segmentation using a RefineNet model. We augment this by including our method to detect Occlusion among randomly placed objects inside a bin and mark a grasping point for the bot with the highest probability of non-occlusion. Therefore, utilizing Semantic Segmentation and Occlusion Detection simultaneously should make the decision making process more efficient for the Robot.

2 Background

2.1 RefineNet: A Brief Overview

RefineNet or Refinement Network is a state-of-the-art method for Semantic Segmentation. It overcomes the conventional disadvantages of segmentation

like loss of finer image structure. In earlier methods of Semantic Segmentation, the low level visual information would be lost as the image is processed through the different layers of a Convolutional Neural Network (CNN). It is essential to store these finer details of image features. This helps in more accurate prediction of the class of the object being labeled. RefineNet achieves this by exploiting features at multiple levels of abstraction resulting in a high resolution output. It combines the fine-grained low level features with the low-resolution semantic features to output a high resolution semantic feature map.

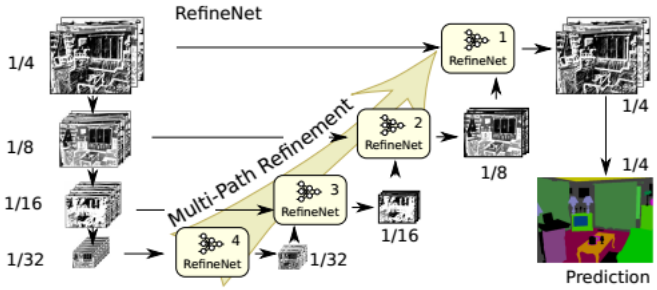


Figure 1. Model of the RefineNet shown to exploit various levels of detail at different stages of convolutions and fuse them to obtain a high-resolution prediction without the need to maintain large intermediate feature maps

The RefineNet Model has been tested on multiple public datasets like PASCAL VOC 2012, NYUDv2, Cityscapes etc. The results have been highly satisfactory as it outperforms all other systems in the highly challenging Cityscape dataset.

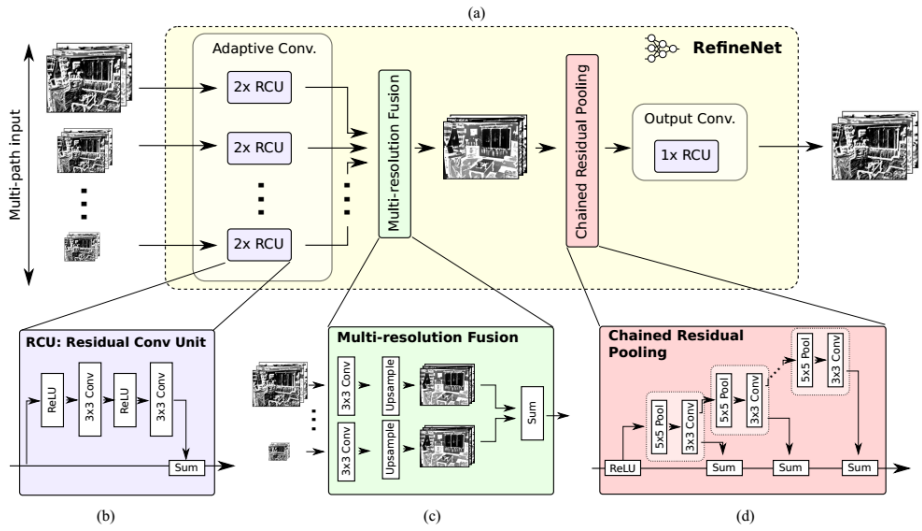


Figure 2. RefineNet Architecture

2.2 Existing Model

The RefineNet architecture has already been implemented for the Object Detection and Semantic Segmentation. This has been done in line with the rules of the Amazon Robotic Challenge 2017, which requires a Robot to perform bin-picking and stowing tasks. The RefineNet model here has been re-implemented based on the ResNeXt-101 network. It follows the primary concept of RefineNet by merging the feature map obtained after each layer with the larger map obtained in the previous step. This ensures the high resolution output of feature maps. The final classification is done using a linear layer and Softmax operation upon each pixel.

3 Proposed Method

We propose a method to include Occlusion Detection in the same RefineNet model mentioned above. In a cluttered environment, an object is to be considered occluded if a certain section of it's surface area is obstructed by one or more objects. Therefore, we prioritize the ones which are not occluded or are lying on top of the pile.

3.1 Algorithm

Let us consider the area of two Objects, once occluded and the other non-occluded.

$$\text{Area of the Section which is Occluded} = A_{oc}$$
$$\text{Area of the Occluded Object} = A_{ob}$$
$$\text{Threshold} = \delta$$

If the Area of the Section which is Occluded multiplied by the threshold (δ), is greater than the total Area of the Occluded Object, then the Object is considered to be Occluded.

$$Occlusion\_Flag = \begin{cases} \text{true}, & \text{if } A_{oc} * \delta > A_{ob} \\ \text{false}, & \text{otherwise} \end{cases}$$

This flag is set for every pixel in the image. Therefore, combined with the label id of each pixel, we can conclude if a particular object is occluded or not.

3.2 Metric Calculation

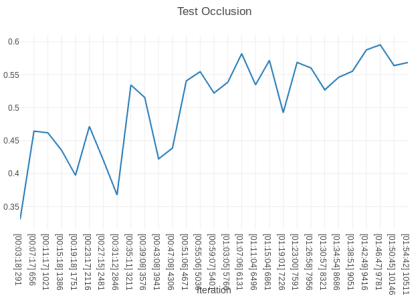
We compute the metric in two sections for this method. One is for both Occluded and Unoccluded Objects and the other is only for the Unoccluded ones.

For each metric calculation, first we take the intersection and union of the predicted and target set of pixels for occlusion and sum them up. Then, the mean of the Occlusion and Un-Occlusion is calculated for the first graph. The second graph is made up of only the values obtained for the Unoccluded objects.

Predicted Set of Pixels for Occlusion = P_v
Target Set of Pixels for Occlusion = T_v
$I = P_v \cap T_v$
$U = P_v \cup T_v$
Metric = $\frac{I}{U}$

For each pixel, we consider a $height * width * 3$ Matrix. The third dimension of the matrix stores the Occlusion Value for each pixel. The first column being for occluded objects, the second for non-occluded and the third for don't care or background condition. The above calculation takes place for all the three classifications.

a)



b)

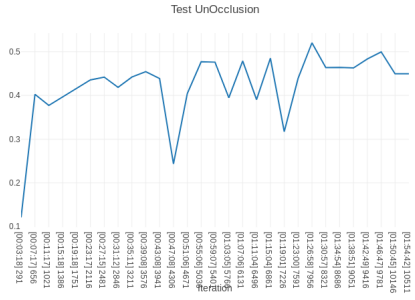


Figure 3 a) Metric for Mean of Occluded and Un-Occluded b) Metric for Non-Occluded

4 Implementation & Evaluation

We implement the above algorithm in a simplified manner for convenience. The training images are already annotated with labels for the Object Detection and we add the flag for Occlusion to these images. Since, the objects are annotated

by using multiple polygons to achieve pixel wise precision, we do not calculate the area of the object explicitly. Instead, we count the number of pixels covered by the Object Mask. The threshold (δ) is set to 20, such that an object is considered to be occluded only if greater than 5% of its surface is covered by one or more objects.

$$20 * A_{oc} > A_{ob}$$
$$\text{or, } A_{oc} > 0.05 * A_{ob}$$

We render the binary masks of objects to a background image (Figure 4a) and use this to train the network. The background image is pre-annotated for both Object Classes and Occlusion. Ideally, this contains a few objects inside a tote or bin.

a)

b)

c)

Figure 4 a) Background with Rendered Object Masks b) Annotated RGB Image for Occlusion with Blue as Un-Occluded, Yellow as Occluded and Green as Don't Care c)Predicted Binary Masks for Occlusion

Additionally, we mark the point in the image, where the probability of non-occlusion is highest. The pixel with the highest probability of un-occlusion is considered and the nearest neighbours around it (the pink rectangle in Figure 5b and 5d) are marked for the Bot to recognize a gripping point. During evaluation of the test images, we mark both the ground truth (Figure 5b) and final output image (Figure 5d) by comparing this pixel having maximum probability of occlusion with that in the ground truth. If the ground truth states that this pixel is un-occluded, then we consider the point to be feasible for gripping. In real time, this comparison will however not take place as the ground truth will not be available to the bot. In a similar way, the noise from Figure 5c i.e. where some sections of the tote are identified as un-occluded objects, is filtered out from the output image (Figure 5d).

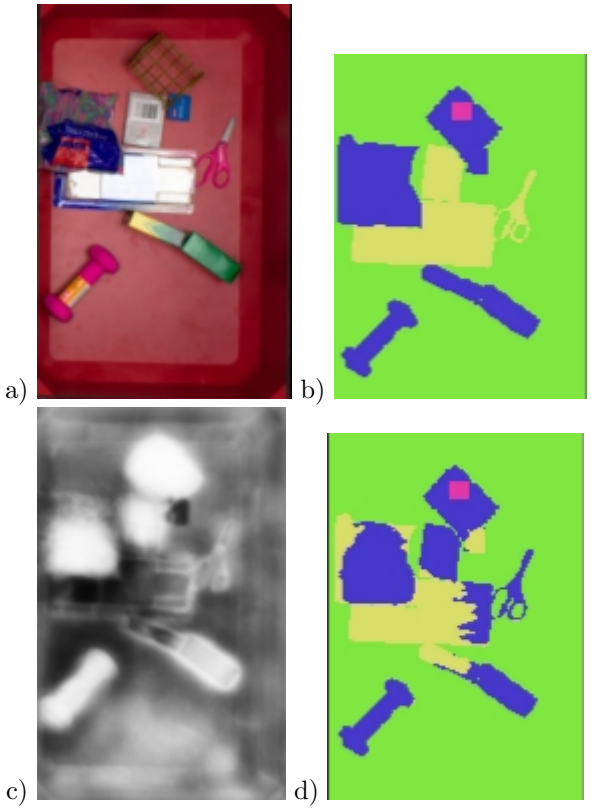


Figure 5 a) Original Test Image b) RGB Image containing Annotations for Occlusion c)Heat Map for Probability of Occlusion of each Pixel d) Final Object Mask with Predicted Occlusion Values

The following images (Figures 6 and 7) are the output over the set of test images after 1021 and 10511 iterations respectively.

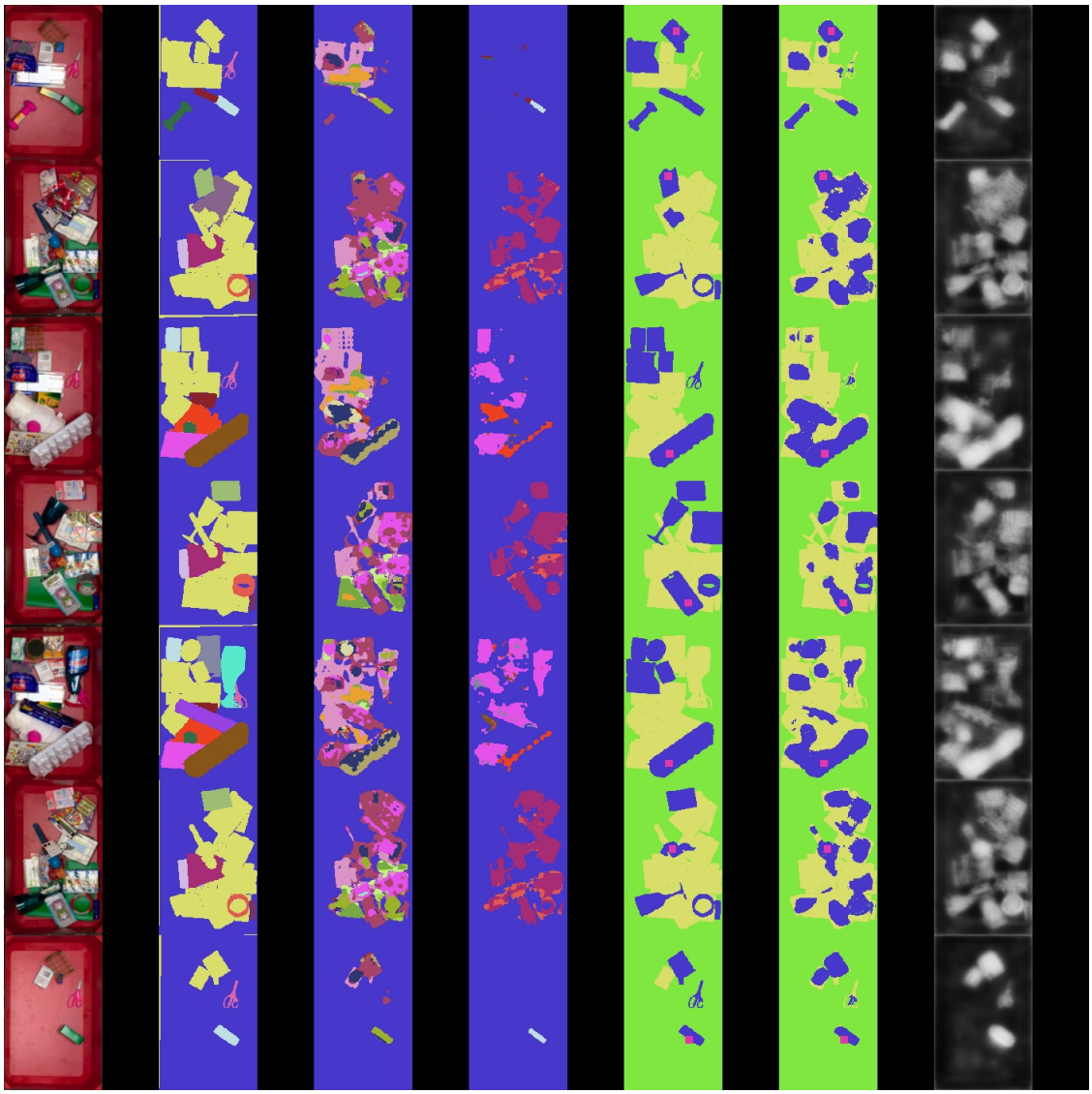


Figure 6: Test Images and their output after 1021 iterations (From Left to Right) Original Test Image, Ground Truth for Semantic Segmentation, Prediction for Semantic Segmentation, Output Image with Semantic Segmentation, Ground Truth for Occlusion, Output Image for Occlusion, Heat Map with Probabilities for Occlusion

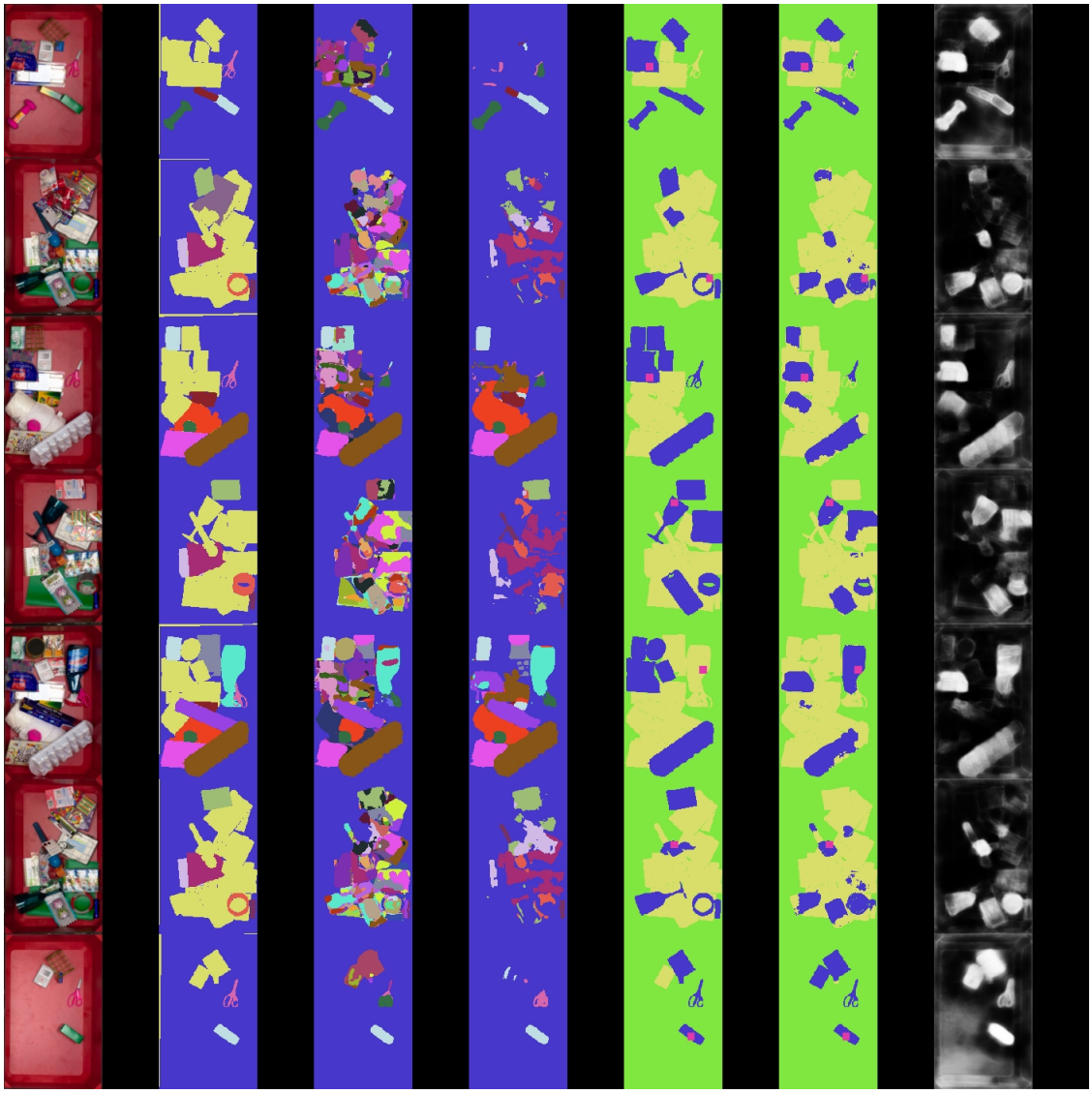


Figure 7: Test Images and their output after 10511 iterations

5 Conclusion & Future Work

It is apparent from the figures, that the network learns to detect the occlusion very early in its cycle. Following this there is little to no learning. The metric output is approximately 0.5 for the mean of the Occlusion and Non-Occlusion parts while it is approx 0.45 for the Un-occluded Parts. Even though this value seems low, the output of the test images is quite precise as it fits well with the ground truth. This could possibly be improved by providing larger datasets for the network to train on.

There are a few special cases like where the object on top might be very thin or may not have a large surface area and the one below it is considerably big. In such a case, the occluded object could be marked as un-occluded. However, if the first object is fragile like a wine glass, it may happen that while picking the second object, the first object gets damaged. Therefore, the value of the threshold has to be carefully calibrated.

It is important to point out here, that the entire algorithm is based on the assumption that no Object will occur more than once in a clutter. This adheres to the rules of the Amazon Robotic Challenge 2017. However, this would restrict the algorithm from being made generic. In the future, this could be achieved by adding more information like instance id to the objects.

6 References

1. "RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation" by Lin *et. al.*
2. "Fast Object Learning and Dual-arm Coordination for Cluttered Stowing, Picking, and Packing" by Max Schwarz *et. al.*
3. Torch: http://torch.ch/