High-Quality GPU-based Deliberative Perception for Object Pose Estimation with RGB data

Shanshan Xie

CMU-RI-TR-21-11

May, 2021

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Maxim Likhachev, Chair
Oliver Kroemer
Sha Yi

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

Known object pose estimation is essential for a robot to interact with the real world. It is the first and fundamental task if the robot wants to manipulate the object. This problem is particularly challenging when the environment is complicated with clutters or the object itself is occluded. Changes in lighting and difficult orientations of the objects also bring challenges to the pose estimation algorithm. Most of the modern approaches need to obtain a large number of training data with accurate ground truth annotations to find the correspondence and output predictions. An alternative is to use a search-based algorithm that finds a pose best explains the scene in all possible rendered poses, which does not require prior knowledge or training except the model of the targeting object. PERCH(PErception Via SeaRCH)[21] is an example that uses depth data to converge to a globally optimal solution by searching over a specific space.

In this work, we propose a PERCH color-only version, a pose estimation algorithm that needs an RGB-only image and the mesh model of the target object. It finds the best explanation for the observed scene by rendering images for all possible poses and evaluating them using a designed cost function that takes into account both image similarity measurement and the rarity for each feature in the scene. The experiment results both from a publicly available dataset and our synthetic dataset show that our algorithm achieves high accuracy, especially in high occlusion scenes without the need for any annotation and training.

Acknowledgments

I would like to express my sincere gratitude to my advisor Prof. Maxim Likhachev for his patience, insightful suggestions, and support throughout my research. His constant encouragement and extraordinary guidance have been immensely valuable to me and will keep motivating me in the future. I would also like to express my gratitude towards Prof. Oliver Kroemer and Sha Yi for their valuable feedback as members of my thesis committee.

I would like to thank my colleagues Aditya Agarwal, Sandip Aine, and Rishi Veerapaneni who worked with me on different aspects of this research and all members of Search-Based Planning Lab for their contribution to maintaining a conducive research environment in the lab.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

(a) (b)

Figure 1.1: Examples of Robot Manipulation

For a robot to interact with objects in the real world safely and reliably, an accurate estimation of the pose (location and orientation of the object) is essential. Our goal is to develop a framework that detects and recovers the 3D pose of the targeting object from an RGB-only image with the knowledge of its mesh model in clutter environments. Many current devices have built-in RGB cameras but no or have low-quality depth-sensors (i.e. Phones, webcams, etc). If it is sufficient to solve the pose estimation problem by using only RGB images, we can avoid adding cost and complexity to the devices.

Most industrial robotic manipulators can reach and grasp the object with high precision when the pose of the object is known. For instance, in factories or autonomous warehouses, objects of interest can be static (on tables or shelves) or dynamic (on a conveyor belt). For both scenarios, robust pose estimation for the object of interest ensures the manipulator can successfully reach and grasp the object. Another application for object pose estimation is in household object manipulation. Robots aim to manipulate common objects, for example, a sugar box or a meat can. These objects can be placed on a table, on shelves, or in household appliances such as refrigerators.

Even though the domains of the applications are different, the underlying pose estimation task and challenges are common. In many cases, the robot has knowledge about the model of the targeting object, and the object is placed in an upright orientation on a horizontal plane. Thus, the main task for pose estimation is to find the location(x, y) and the rotation(yaw) of the object. There are multiple aspects of the environment that can introduce challenges for the pose estimation algorithm. Different settings will cause the observed images to have different lighting, resolutions, and background. Another common challenge is object occlusion. Target objects can be occluded by themselves, other objects in the scene, or limited camera field of view. All of them can have a negative impact on the performance of the pose estimation algorithm.

1.2 Existing Approaches

Traditional object pose estimation focused on the matching hand-crafted local features(e.g. SIFT[18]) between 3D models and images. Feature-based methods [2, 8, 24] normally requires rich textures on the objects and fails to find good estimations when features are occluded. With the rapid development of convolutional neural networks(CNNs) on 2D object detection, many learning-based methods have been proposed to estimate objects in 3D space[5, 9, 16, 17, 23, 29, 31]. However, these methods usually require a large number of training data for accurate pose estimation. It is also hard for them to scale with the number of objects since the network needs to be trained on each object from multiple viewpoints and occlusion conditions. The annotation for the ground truth pose is non-trivial and often labor-intensive.

Methods that match synthesized scenes with observed scenes overcome shortcomings of feature-based and learning-based methods but are normally slow. More specifically, Perception Via Search(PERCH)[1, 20, 21, 22] is a recent work that introduces a globally cost function. This cost function can guide the search and find the pose that best explained the observed scene.

1.3 Proposed Approach

This work focuses on the pose estimation of an object in the clutter from an RGB-only image. Building on the prior deliberative methods such as PERCH, we combine advantages from newly developed computer vision algorithms and RMR (render, match, refine) philosophy. The previous approaches in the deliberative method category primarily use depth data and color information is only used to distinguish between objects with the same shape, while our proposed method aims to achieve high accuracy using color information alone.

The key contributions of our work can be listed as follows:

- Deliberative 3-Dof pose estimation framework from a RGB-only image
- Fully parallel and scalable GPU-based deliberative pose estimation method that achieves speedup over the previous method like PERCH[21]
- Designed a cost function that utilizes pre-trained neural networks for feature extraction and the concept of rarity to reduce the requirement on textures of the object and improve robustness especially in occluded scenes

Chapter 2

Related Work

Methods for pose estimation of objects can be broadly divided into traditional methods (section 2.1), discriminative methods (section 2.2) and deliberative methods (section 2.3).

2.1 Traditional Methods

Traditional computer vision algorithms use hand-crafted local 3D features to establish correspondence between 2D image and 3D model which can be used to recover the object pose.[2, 3, 15, 19, 24, 25, 26, 30] Other traditional methods used template matching technique which computes similarity scores across different locations in the input image with a rigid template constructed by rendering a 3D model and the best match is obtained by comparing these similarity scores[7, 11, 12]. Feature-based methods require rich textures to be presented in the input image so that they can find a match. While some template matching methods can deal with textureless objects by comparing surface normals and colors, they usually fail to detect accurate poses under occlusion.

2.2 Discriminative Methods

With the development of deep learning, state-of-the-art approaches employ learning techniques to detect object key points or learn better feature representations for pose estimation.[5, 17] These methods can be further divided into 3 main categories: 1. Methods that utilize deep learning to detect key points or features and then solve the PnP problem for the final 6D pose [23, 29] 2. Methods that directly regress to its 6D pose after detecting the object in the input image using deep learning.[16, 31] 3. Methods that discretize the 6D space and score each pose using a classifier[9]. Overall, learning-based methods achieve better performance than traditional methods, largely due to a more powerful feature representation. However, all learning methods require extensive annotation of ground truth 6D poses in training data. Moreover, the dataset is very hard to scale with the number of objects since the neural network needs to be trained on every object from multiple viewpoints with varying degrees of inter-object occlusions.

Recent research works [10, 28] utilize an autoencoder to map the object in the image to a vector and search for the most similar vector in a pre-generated codebook for pose estimation. Even though this method avoids the heavy pose annotation requirement, these methods still require training of a pose estimation neural network in addition to training more standard tasks such as instance segmentation and object detection.

2.3 Deliberative Methods

Analysis-by-synthesis or deliberative methods[1, 4, 20, 21, 22, 27] rely on rendering and verification. The aim is to find the best-rendered image that explains the observed scene. They will have the access to the 3D model and a renderer which can be used to generate candidate images. Past work on Perception via Search (PERCH)[1, 20, 21, 22], has demonstrated the capabilities of combining rendering with the search for multi-object 3-Dof pose (x, y, yaw) estimation under occlusion and clutter. However, PERCH[21] and PERCH 2.0[1] rely heavily on depth images for accurate pose estimation, and color information is only used to distinguish between objects with the same shape.

Chapter 3

PERCH Color Only

In this section, we describe our proposed approaches and the interpretation for each design. In Section 3.1, we described the problem formulation and the assumptions we made for our approach. In section 3.2, we describe our method including metric for image similarity and the concept of rarity.

3.1 Problem Formulation

Perch color only is a pose estimation algorithm that takes in a RGB-only image and the mesh model of the target object. The problem statement is as follows: Given the 3D model of the target object, we are required to find the 3 DoF pose (x, y, theta) of the object in a observed RGB image.

3.1.1 Assumptions

Perch Color Only has several assumptions about the environment.

- all objects are located on a known plane with known background, meaning that we know the parameters of the table plane or shelf respected to the camera.
- The targeting object varies only in position (x, y) and yaw (theta) with respect to the 3D model.
- We have the mesh model of the target object and the intrinsic parameters of the camera so that we have the ability to generate candidate images for all

possible poses using our renderer. (renderer will not render any effects such as lighting, shadows or different resolutions)

In factories or autonomous warehouses, the environment is relatively stable and tasks are repeated, meaning that there will not be huge changes every day. Thus it is reasonable to assume that we have the knowledge of all the plane parameters, camera poses as well as all the models for target objects.

3.2 Method

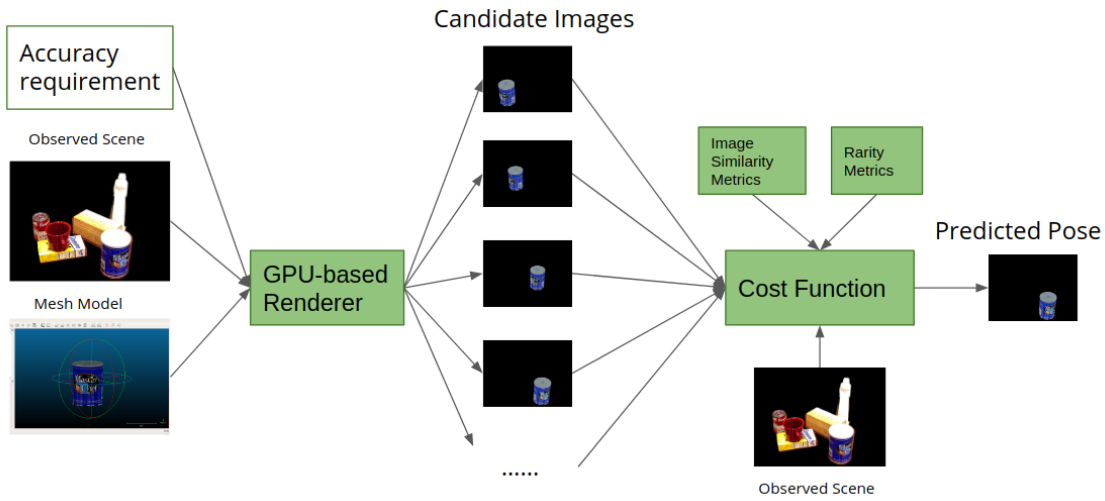


Figure 3.1: Visualization for Perch Color Only Pipeline

Perch Color Only is a deliberative method that builds on a previous paper: Perception Via Search[21]. As shown in Figure 3.1, it first defines a search space which can be a known plane such as a tabletop and search resolution for x, y, and yaw axes so that the renderer can generate candidate images for all possible poses with the mesh model utilizing parallelization capabilities of GPU. After rendering images for all possible poses, we need to construct a cost function that outputs the best score when the candidate pose is closest to the ground truth pose, meaning that the designed cost function needs to find the global optimum solution among all the candidate images. We design it from two aspects: Image similarity and Rarity, which we will discuss in

Algorithm 1 Perch Color Only Pipeline

```
function PERCH COLOR ONLY(Observed Image, Mesh Model, Search Space)
    Candidates = [ ]
    for every pose in Search Space do
        Rendered Image= Renderer(Mesh Model, pose)
        Candidates.append((Rendered Image, pose))
    end for
    Max Score = 0
    Predicted Pose = None
    for every (Candidate Image, Candidate Pose) in Candidates do
        Score = CostFunction(Candidate Image, Observed Image)
        if Score > Max Score then
            Max Score = Score
            Predicted Pose = Candidate Pose
        end if
    end for
    Return Predicted Pose
end function
```

detail in the following sections. In other words, we define a score for every feature vector:

$$S(v) = \sum_{\alpha} \text{Image Similarity}(R_v, I_v) \times \text{Rarity}(I_v) \tag{3.1}$$

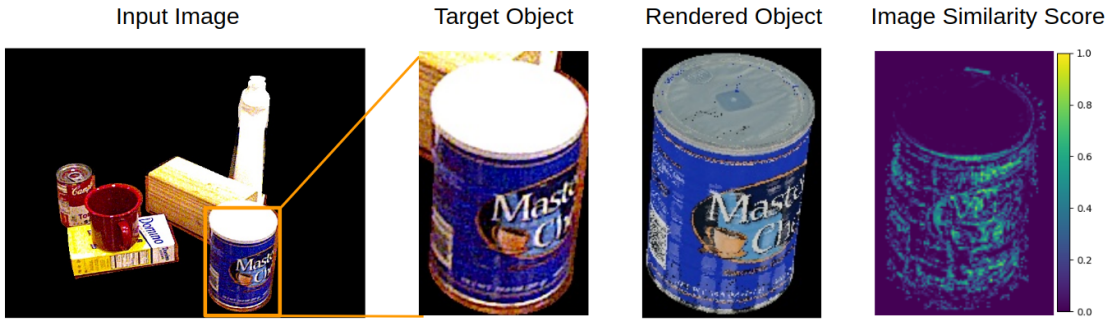
where v is the feature vector, R is the rendered image for the pose and I is the input image. For each rendered image, we calculate the score for every feature and the final score is the mean of top α percent of the scores, where α is the occlusion limitation ratio (at least α% of the object needs to be non-occluded). We exhaustively search for every rendered image to find the highest score, and the corresponding pose for that image is our predicted pose. Algorithm 1 shows the pseudocode for Perch Color Only, the Renderer function generates candidate images from the mesh model and the pose in camera frame which is a standard function that can be run on GPU. The CostFunction is our designed function and its pseudocode is shown in Algorithm 2.

Algorithm 2 Cost Function with Image Similarity and Rarity

function CostFunction(Candidate Image, Observed Image, Occlusion Limitation Ratio α)
    Candidate Image = AlexNet First Layer(Candidate Image)
    Observed Image = AlexNet First Layer(Observed Image)
    Image Similarity Score = Cosine Similarity (Candidate Image, Observed Image)
    Rarity = 1 - Normalized frequency from Kmeans(Observed Image)
    Total Score = Image Similarity Score $\times$ Rarity
    Final Score = Mean(top α% in Total score)
    Return Final Score
end function

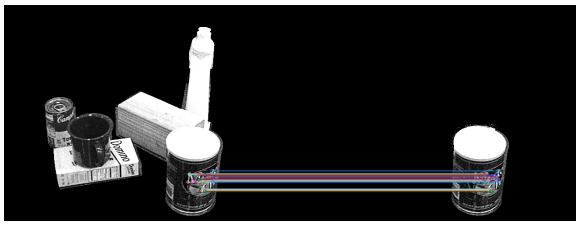
3.2.1 Image Similarity

If we have the perfect renderer, intuitively, we only need a pixel-wise comparison since at the correct pose, part of the rendered image should match exactly with the input image. However, because of the imperfection render, change in light conditions, occlusion, etc., the construction for the cost function is not trivial. As shown in Figure 3.2, a cost function that purely relies on pixel values is not good enough and the output for the accurate pose is perceptually bad. One step further, comparing kernels instead of pixel values may alleviate this problem since a kernel takes into account multiple pixel values and might be able to capture features instead of individual values. The natural question here is what should the kernels be. Traditionally, kernels are manually designed to capture edges or lines, such as Scale-Invariant Feature Transform(SIFT)[19]. Nevertheless, rendered images and input images have different resolutions and detailed features, which can make a huge difference in edge or line descriptors, shown in Figure 3.3. Furthermore, in highly occluded scenes, the traditional feature descriptors will fail to gain any useful information due to lack of edges and corners, while color information is presented and ignored by the feature descriptor.

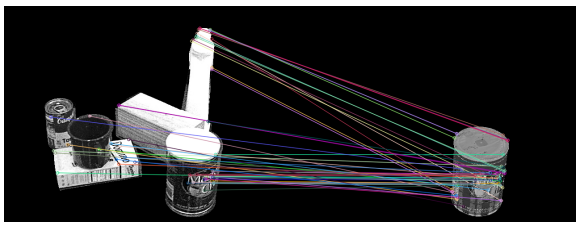


(a)

Figure 3.2: Visualization for Pixel-wise Image Similarity Score



(a) key points correspondence with perfect render



(b) Key points correspondence with real render

Figure 3.3: Visualization for traditional feature-based Image Similarity Score

We need a more general metric, which takes in both color and features such as edges and corners, to capture image similarity under different conditions. From the literature, we know that CNN filters learn both colors and geometric features from the dataset. In this pipeline, we use the first layer of the pre-trained AlexNet model to convert the images to their feature spaces. The AlexNet is trained on more than a million images from the ImageNet database but not trained on our specific object dataset. In the first layer of AlexNet, the network has learned a variety of frequency- and orientation-selective kernels, as well as various colored blobs that resemble some

traditional filters, shown in figure 3.4. We use this layer to extract general features which can be compared later. Since it contains both color kernels as well as geometric feature kernels, under high occlusion, even if there are no geometric features shown, it still captures the color similarity. More specifically, every pixel goes from R^3 (r,g,b color space) to R^{64} (Feature space). After converting both the rendered images and the input image to feature space, we compute the cosine similarity between corresponding features as our image similarity score, shown in Figure 3.5. The actual computation for image similarity is shown as the top half portion of pseudocode in Algorithm 2

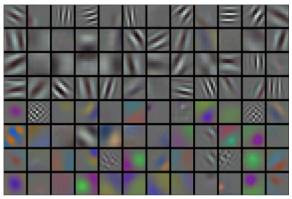
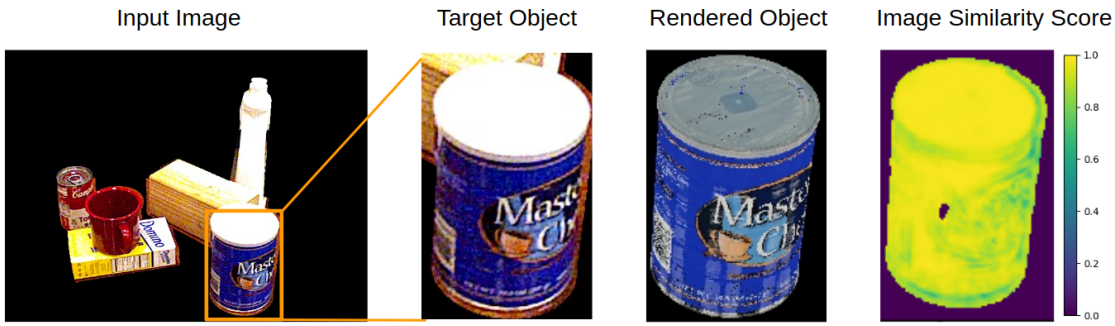


Figure 3.4: First Layer visualization of AlexNet



(a)

Figure 3.5: Visualization for CNN based Image Similarity Score

3.2.2 Rarity

Rarity score is inspired by the TF-IDF(term frequency-inverse document frequency) which was invented for document search and information retrieval. In TF-IDF, words that are very common in the text such as “if”, “the” and “what” rank low even

if they appear multiple times. We borrowed the idea and applied it to the image domain. Common colors are weighted less than rare colors. For example, if multiple objects have white with large regions, matching white does not mean much for the cost function. However, if red is very rare in the picture, it should be contributing more to the cost function.

For the actual computation of the rarity score, if we are only considering different colors, then the most intuitive method is to build a histogram of colors and count for their frequencies. However, we convert all the images to the feature space which has 64 dimensions thus, binning every vector to a histogram is too expensive computationally and extremely inefficient. Thus, we used the KMeans algorithm to estimate the rarity score. Since we have high dimensional data and aim for a reasonable processing time, we only consider 20 clusters for an input image. It assigns every vector to one cluster which can be further calculated to the frequency distribution of the image. Rarity score is derived from normalized frequency ranging from 0 to 1 where a higher score suggests the feature is rarer. The actual computation for rarity is shown as the bottom half portion of pseudocode in Algorithm 2

Chapter 4

Experimental Results

4.1 YCB Dataset

Since there is no publicly available dataset for 3D(x, y, yaw) only pose estimation, we used a subset of YCB[6] dataset(BOP version)[13] where the objects are placed upright on a plane. The BOP: Benchmark for 6D Object Pose Estimation version of the YCB dataset remove redundancies and avoid images with erroneous ground-truth poses. For each video sequence, 75 images are selected to represent the scene. Among all the test scenes, we choose 4 objects to be evaluated: 004 sugar box (75 images), 005 tomato soup can (300 images), 009 gelatin box (75 images) and 010 potted meat can (225 images).

Figure 4.1: Examples input images from YCB dataset

4.1.1 Baseline

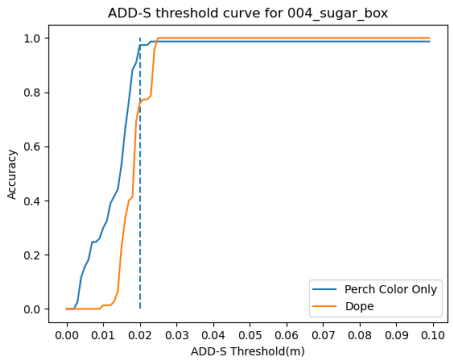
DOPE: Deep object pose estimation for semantic robotic grasping of household objects[14] is a leading RGB-based discriminative key point localization method. More importantly, it is the first deep network trained only on the synthetic dataset that is able to achieve state-of-the-art performance on 6-DoF object pose estimation. It is trained on ∼60k domain-randomized image frames (per object) mixed with ∼60k photorealistic image frames (same for all objects). This is a fair comparison for Perch Color Only since both do not require real training data and annotation. Since DOPE is a 6-DoF pose estimation algorithm and Perch Color Only is focusing on 3-DoF prediction, we feed the z, roll, pitch ground truth values to DOPE, and only x, y, yaw from the prediction are used in evaluation such that the comparison is fair.

4.1.2 Metrics

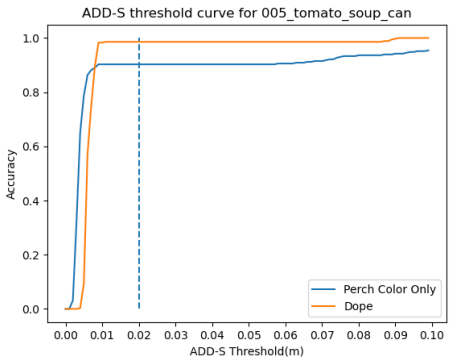
We use the ADD-S[12, 31] metric for evaluation which computes the average of pairwise distances between the object model transformed by the ground truth pose and the model transformed by the predicted pose. We vary the ADD-S distance threshold up to 0.1 m and obtain the area under the accuracy-threshold curve (AUC) for all methods. We also compute ADD-S $\leq$ 2cm, which denotes the percentage of poses with less than 2cm ADD-S error. We chose 2 cm because most of the modern manipulators such as Baxter fail to grasp or manipulate the object if the predicted pose has an error larger than 2 cm.

objects	Perch Color Only		DOPE[14]	
	AUC	$\leq$ 2cm	AUC	$\leq$ 2cm
004 sugar box	**86.33**	**98.67**	80.82	76.00
005 tomato soup can	87.21	90.2	**91.48**	**98.40**
009 gelatin box	**87.37**	**100.00**	83.38	**100.00**
010 potted meat can	**92.23**	**99.60**	45.72	46.20
All Objects	**88.29**	**97.13**	75.40	80.15
Mean Runtime	106.8s		1.0s	

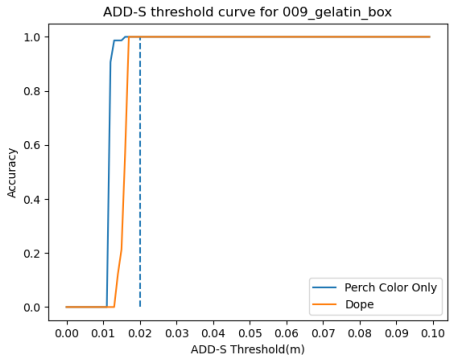
Table 4.1: Evaluation of 3D pose estimation on YCB dataset



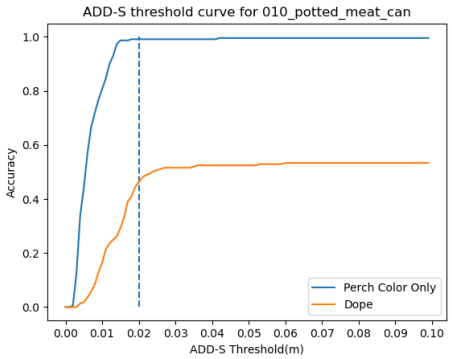
(a) Sugar Box



(b) Tomato Soup Can



(c) Gelatin Box



(d) Potted Meat Can

Figure 4.2: ADD-S threshold curve for YCB dataset

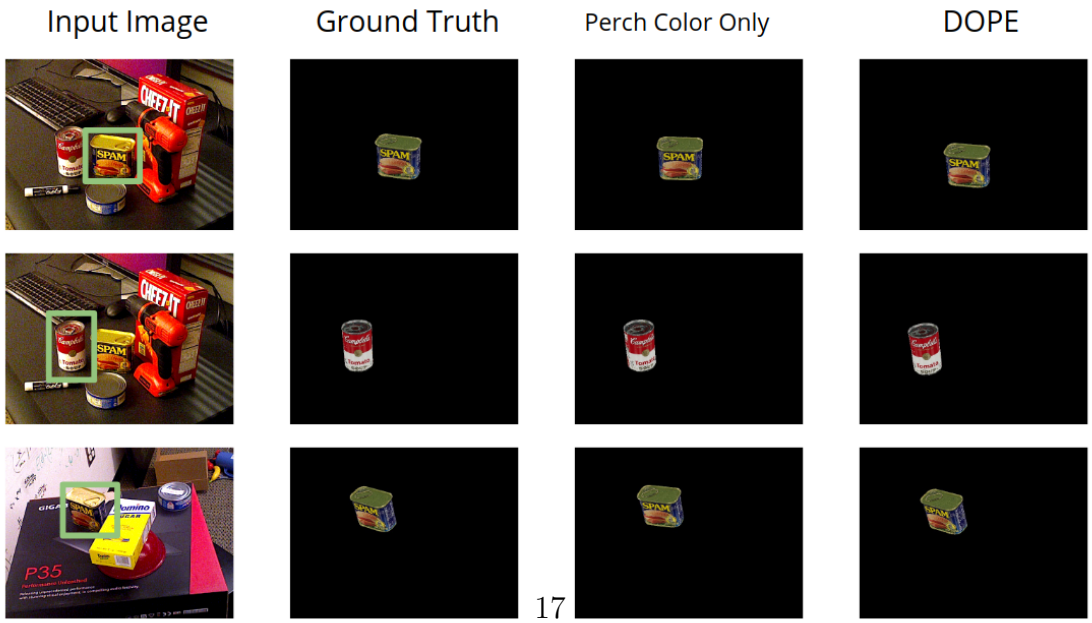


Figure 4.3: Examples of qualitative output for both methods

4.1.3 Accuracy

As shown in Figure 4.2 and Table 4.1, Perch color Only achieves relatively high accuracy on all objects especially on the potted meat can while DOPE has better performance on the tomato soup can. We will further analyze the results in the following sections. Qualitative results are shown in Figure 4.3. The first column shows the input images and the target object in the green box, the second column shows the ground truth pose for the target object. The third and fourth columns are results from two methods.

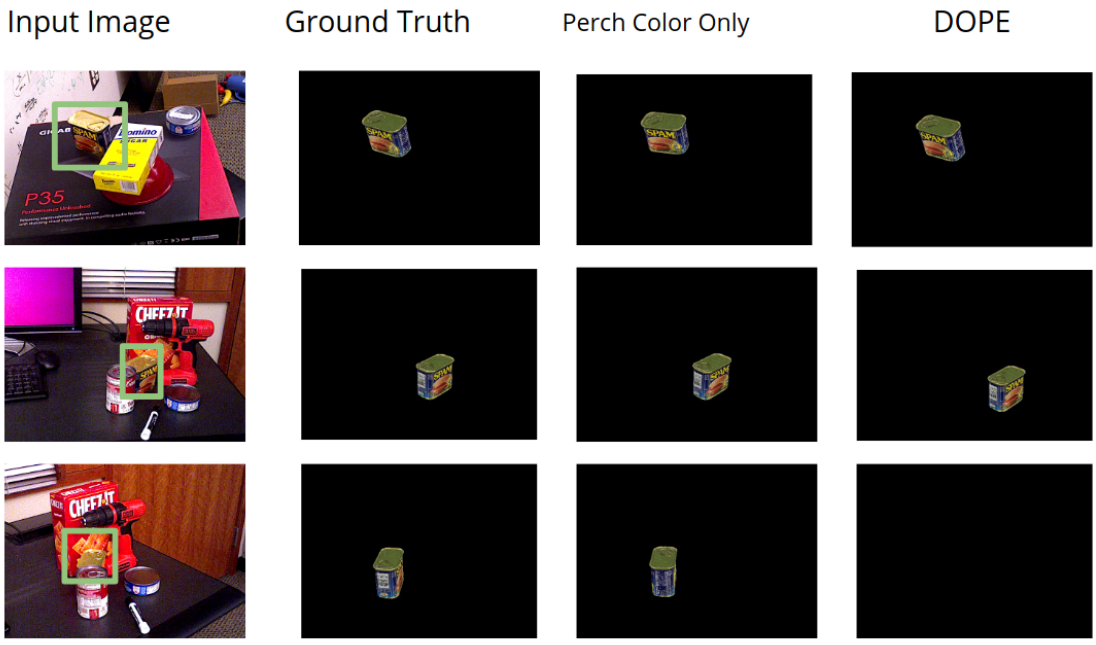


Figure 4.4: Examples of qualitative output with occlusion for both methods

4.1.4 Strengths and Weaknesses

The strength of the Perch Color Only is to work in the scenes that the object is heavily occluded while the target object has a relatively unique color or feature shown in the observed image. The weakness for the algorithm is the run-time, shown in Figure 4.5. Even though the pipeline is GPU-based, because of the large search space and the fine resolution for high accuracy, the overall run-time is still high and

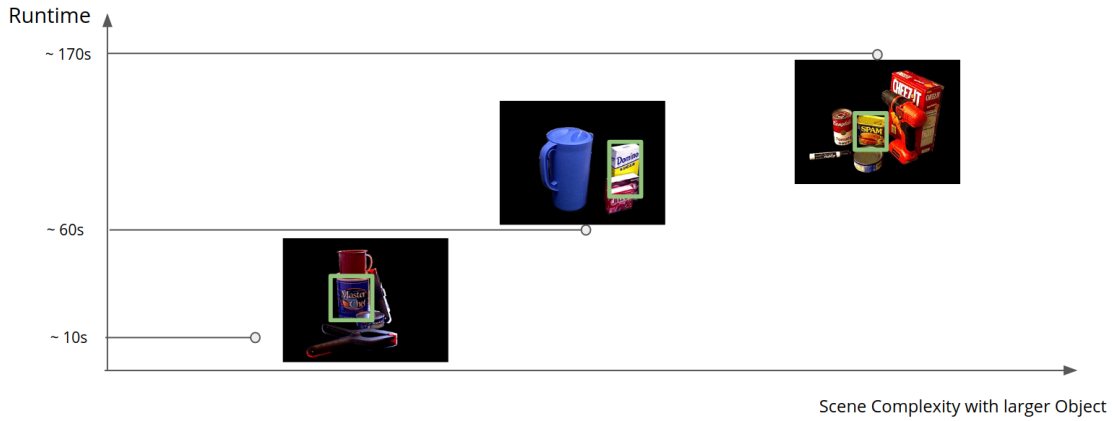


Figure 4.5: Run-time Analysis for Perch Color Only

it heavily depends on the scene complexity and the objects' sizes. If the scene is relatively simple and the target object is one of the largest objects in the image, then the run-time is relatively low. If the scene is more complicated and other objects in the scene increase their sizes, the run-time will also significantly increase. In Figure 4.5, we see that the rightmost scene is complicated and the target object is relatively small compared to other objects, thus the run-time is very high (about 170s).

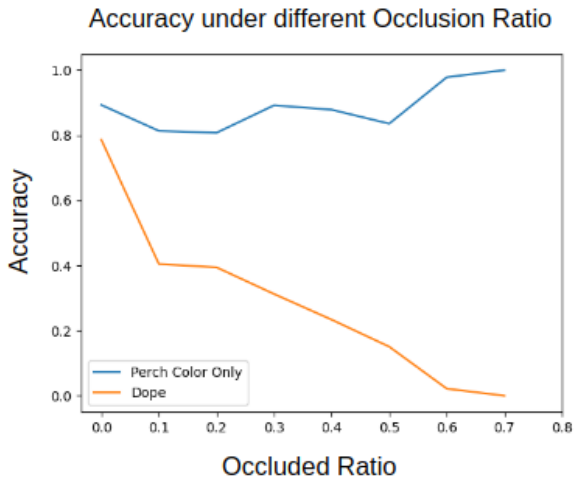


Figure 4.6: Occlusion Analysis on YCB dataset

4.1.5 Robustness under occlusion

In order to see why Perch Color Only achieves far better results on Potted Meat Can where the object is highly occluded in most of the scenes, we analyzed performance under different occlusion levels shown in Figure 4.6. The x-axis shows the occlusion ratio for the scenes and the y-axis shows the percentage of correct pose prediction (threshold is 2 cm). We see that up to 70 percent of occlusion will not have a significantly negative impact on Perch Color Only while it causes DOPE to have a sharp decline in accuracy. Note that for the Perch Color Only methods, the accuracy increases with the occlusion ratio. We think it is due to the limitation of the dataset where only a small amount of the data have such high occlusion. The scenes with very high occlusion are in the Perch Color Only's strength field so that the accuracy seems to be increased. We will discuss more concrete experiments for analyzing performances in the future work section.

Figure 4.7: Lighting Analysis on YCB dataset

4.1.6 Robustness under different lighting conditions and shadow

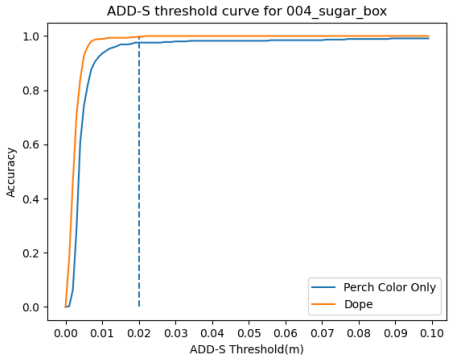
Since the YCB dataset is an indoor dataset that does not contain images in extreme lightings or under large shadows. We cannot do a thorough experiment on Perch Color Only under different lighting conditions as well as under big shadows. However, we manually changed the lighting and applied some random shadow to some of the images, examples are in Figure 4.7. Perch Color Only achieves similar performance under normal lighting. A more carefully designed experiment needs to be performed for the concrete conclusion.

4.2 Synthetic Dataset

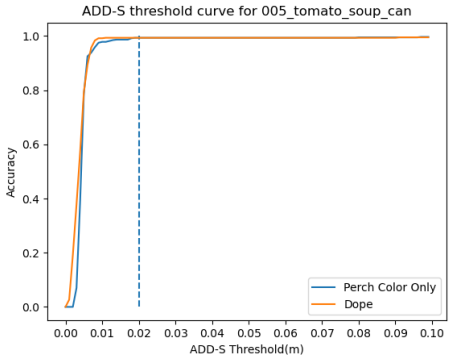
We constructed a synthetic photo-realist dataset of 800 scenes(including 150 scenes with 1 object, 150 scenes with 2 objects, 200 scenes with 3 objects, 250 scenes with 4 objects, and 50 scenes with 5 objects) with corresponding RGB images using the recently released NVidia NDDS plugin for Unreal Engine 4 (objects shown in Figure). Within the plugin, we randomly vary the 3D pose (x, y, yaw) of every object on a tabletop while keeping (z, roll, and pitch) constant. The plugin allows the generation of images with realistic lighting conditions and inter-object occlusion. We used the same metrics and baseline comparison as in the YCB dataset. The results are similar to the YCB dataset, Perch Color Only achieves high accuracy on all objects and has a better average ADD-S AUC and $\leq$ 2cm error.

objects	Perch Color Only		DOPE[14]	
	AUC	$\leq$ 2cm	AUC	$\leq$ 2cm
004 sugar box	93.35	97.80	**96.45**	**99.80**
005 tomato soup can	93.91	**99.3**	**94.70**	**99.3**
009 gelatin box	**90.71**	**93.00**	87.40	**90.70**
010 potted meat can	**90.80**	**99.40**	83.00	88.80
All Objects	**92.19**	**97.28**	90.39	94.65
Mean Runtime	95.4s		1.0s	

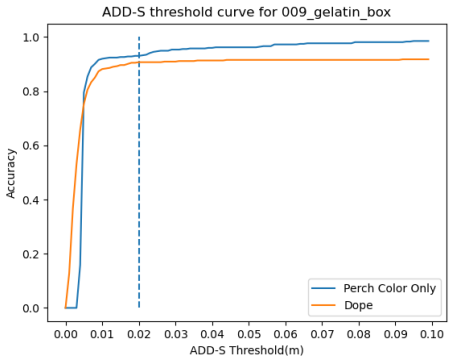
Table 4.2: Evaluation of 3D pose estimation on synthetic dataset



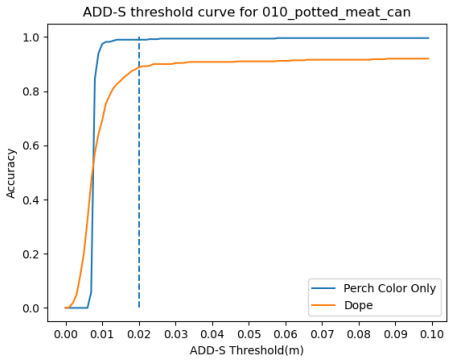
(a) Sugar Box



(b) Tomato Soup Can



(c) Gelatin Box



(d) Potted Meat Can

Figure 4.8: ADD-S threshold curve for Synthetic dataset

Chapter 5

Conclusions

In this work we introduced Perch Color Only, a GPU-based parallel deliberative RGB only pose estimation algorithm that searches over all possible poses and finds the best explanation of the observed scene. The proposed algorithm is able to detect and accurately predict the pose of the target object under high occlusions, different lighting conditions, and environments. Within the proposed pipeline, we use the pre-trained neural network for feature extraction and compute the image similarity since CNN captures both color features as well as features such as lines and corners. The cost function also consists of a rarity score which is inspired by the TF-IDF algorithm from natural language processing. Thus, the algorithm is directed to match unique colors or geometric features. The experimental results both from the publicly available dataset and our synthetic dataset show that our algorithm achieves high accuracy, especially in high occlusion scenes.

5.1 Future work

Our proposed framework provides several directions for future work on deliberative pose estimation.

The main part of the pipeline utilizes parallelization capabilities of the GPU, but it is not scalable with the search space without pruning or prioritization. Before we generate rendered images on a particular pose, we already know the model and

the observed scene. If we can prune away all the poses that have no matching color or features, and re-prioritize the rest with a confidence score and a lower bound, we can significantly shrink the search space thus scalable to larger environments. Currently, Perch Color Only detects objects 3 DoF which means objects need to be placed upright in a known plane. However, 6 DoF object pose estimation is more commonly used and can be applied to more scenarios. With the accelerated pipeline, the algorithm will be able to render a full 6 DoF poses and found the accurate pose among them.

Image similarity measurement is an ambiguous metric and is still an active research area. In our pipeline, we used the first layer of AlexNet to extract features. However, the size of the filters in the first layer is 11*11 which may be too small to capture larger features (for example letters or patterns) in the image considering image size in the YCB dataset is 640*480 with the camera close to the objects. Different pre-trained neural networks have different filters and finding a quantitative way to evaluate all the design choices is also an essential task that needs to be done.

YCB dataset has limited high occlusion scenes with upright objects or scenes with large shadows and extreme light conditions. In order to concretely evaluate the performance of the Perch Color Only algorithm, more public data or synthetic data need to be tested from different perspectives.

Bibliography

[1] Aditya Agarwal, Yupeng Han, and Maxim Likhachev. Perch 2.0 : Fast and accurate gpu-based perception via search for object pose estimation. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. doi: 10.1109/iros45743.2020.9341257. 1.2, 2.3

[2] Aitor Aldoma, Markus Vincze, Nico Blodow, David Gossow, Suat Gedikli, Radu Bogdan Rusu, and Gary Bradski. Cad-model recognition and 6dof pose estimation using 3d cues. *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011. doi: 10.1109/iccvw.2011.6130296. 1.2, 2.1

[3] Aitor Aldoma, Federico Tombari, Radu Bogdan Rusu, and Markus Vincze. Our-cvfh – oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6dof pose estimation. *Lecture Notes in Computer Science Pattern Recognition*, page 113–122, 2012. doi: 10.1007/978-3-642-32717-9_12. 2.1

[4] Aitor Aldoma, Federico Tombari, Luigi Di Stefano, and Markus Vincze. A global hypotheses verification method for 3d object recognition. *Computer Vision – ECCV 2012 Lecture Notes in Computer Science*, page 511–524, 2012. doi: 10.1007/978-3-642-33712-3_37. 2.3

[5] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. *Computer Vision – ECCV 2014 Lecture Notes in Computer Science*, page 536–551, 2014. doi: 10.1007/978-3-319-10605-2_35. 1.2, 2.2

[6] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. *2015 International Conference on Advanced Robotics (ICAR)*, 2015. doi: 10.1109/icar.2015.7251504. 4.1

[7] Zhe Cao, Yaser Sheikh, and Natasha Kholgade Banerjee. Real-time scalable 6dof pose estimation for textureless objects. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016. doi: 10.1109/icra.2016.7487396. 2.1

[8] Alvaro Collet, Manuel Martinez, and Siddhartha S Srinivasa. The moped framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research*, 30(10):1284–1306, 2011. doi: 10.1177/0278364911401765. 1.2

[9] Enric Corona, Kaustav Kundu, and Sanja Fidler. Pose estimation for objects with rotational symmetry. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018. doi: 10.1109/iros.2018.8594282. 1.2, 2.2

[10] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. Poserbpf: A rao-blackwellized particle filter for6d object pose estimation. *Robotics: Science and Systems XV*, 2019. doi: 10.15607/rss.2019.xv.049. 2.2

[11] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):876–888, 2012. doi: 10.1109/tpami.2011.206. 2.1

[12] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Kurt Konolige, Gary Bradski, and Nassir Navab. Technical demonstration on model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. *Computer Vision – ECCV 2012. Workshops and Demonstrations Lecture Notes in Computer Science*, page 593–596, 2012. doi: 10.1007/978-3-642-33885-4_60. 2.1, 4.1.2

[13] Tomáš Hodaň, Martin Sundermeyer, Bertram Drost, Yann Labbé, Eric Brachmann, Frank Michel, Carsten Rother, and Jiří Matas. Bop challenge 2020 on 6d object localization. *Computer Vision – ECCV 2020 Workshops Lecture Notes in Computer Science*, page 577–594, 2020. doi: 10.1007/978-3-030-66096-3_39. 4.1

[14] B. Sundaralingam Y. Xiang D. Fox J. Tremblay, T. To and S. Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *CoRL*, 2018. 4.1.1, **??**, **??**

[15] A.e. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999. doi: 10.1109/34.765655. 2.1

[16] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. doi: 10.1109/iccv.2017.169. 1.2, 2.2

[17] Alexander Krull, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Learning analysis-by-synthesis for 6d pose estimation in rgb-d images. *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015. doi: 10.1109/iccv.2015.115. 1.2, 2.2

[18] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. doi: 10.1023/b:visi.0000029664.99615.94. 1.2

[19] D.g. Lowe. Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999. doi: 10.1109/iccv.1999.790410. 2.1, 3.2.1

[20] Venkatraman Narayanan and Maxim Likhachev. Discriminatively-guided deliberative perception for pose estimation of multiple 3d object instances. *Robotics: Science and Systems XII*. doi: 10.15607/rss.2016.xii.023. 1.2, 2.3

[21] Venkatraman Narayanan and Maxim Likhachev. Perch: Perception via search for multi-object recognition and localization. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016. doi: 10.1109/icra.2016.7487711. (document), 1.2, 1.3, 2.3, 3.2

[22] Venkatraman Narayanan and Maxim Likhachev. Deliberative object pose estimation in clutter. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017. doi: 10.1109/icra.2017.7989357. 1.2, 2.3

[23] Mahdi Rad and Vincent Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. doi: 10.1109/iccv.2017.413. 1.2, 2.2

[24] Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision*, 66(3):231–259, 2006. doi: 10.1007/s11263-005-3674-1. 1.2, 2.1

[25] R B Rusu, G Bradski, R Thibaux, and J Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010. doi: 10.1109/iros.2010.5651280. 2.1

[26] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. *2009 IEEE International Conference on Robotics and Automation*, 2009. doi: 10.1109/robot.2009.5152473. 2.1

[27] Mark R. Stevens and J.ross Beveridge. Localized scene interpretation from 3d models, range, and optical data. *Computer Vision and Image Understanding*, 80 (2):111–129, 2000. doi: 10.1006/cviu.2000.0821. 2.3

[28] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. *Computer Vision – ECCV 2018 Lecture Notes in Computer Science*, page 712–729, 2018. doi: 10.1007/978-3-030-01231-1_43. 2.2

[29] Henning Tjaden, Ulrich Schwanecke, and Elmar Schomer. Real-time monocular pose estimation of 3d objects using temporally consistent local color histograms. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. doi: 10.1109/iccv.2017.23. 1.2, 2.2

[30] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. *Computer Vision – ECCV 2010 Lecture Notes in Computer Science*, page 356–369, 2010. doi: 10.1007/978-3-642-15558-1_26. 2.1

[31] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *Robotics: Science and Systems XIV*, 2018. doi: 10.15607/rss.2018.xiv.019. 1.2, 2.2, 4.1.2