Sparse Neural Generative Inference Based Pose Estimation

Stanley Lewis, Manas Buragohain, Danish Syed, Bahaa Aldeeb
{stanlew, manasjb, dasyed, baldeeb}@umich.edu

Abstract

Current state-of-the-art pose estimation techniques often rely on end-to-end trained neural network architectures to infer a 6-DOF pose. These architectures, while effective, are computationally expensive and often require a large number of parameters. Instead, we propose formulating the pose estimation problem as a differentiable particle filter. This approach learns a per-particle latent embedding which infers its respective pose, object likelihood, and re-sampling objective iteratively. Our proposed method should decrease the tight coupling between input size and model architecture, and also improve the training and inference time scalability of the network. Furthermore, the proposed approach allows for reasoning across a distribution of inferred poses represented by the different particles.

1. Introduction

Computer vision related research communities that attempt to interact with the outside environment - such as robotics, HCI, and others - require the ability to know the spatial location and orientation of objects within their system's vicinity. The task of obtaining this information from an image is known as pose estimation. The determination of an object pose is necessary for any robotic manipulation task. Furthermore, these systems are required to be able to determine the pose of relevant objects in natural environments that contain difficult conditions such as lighting variations, partial observations due to clutter, sensor variations, etc. These difficult conditions often necessitate the use of belief space planners, which require not just a singular pose output (which traditionally would represent the highest likelihood estimate of, for example, a particle filter), but instead required a distribution of belief across candidate poses.

We propose formulating the pose estimation problem using a differentiable particle filter. The proposed method learns the component functions of a traditional particle filter, where each component is represented as a multi-layer Perceptron (MLP), and each task (likelihood, etc.) becomes a learned function. This approach naively allows the use of belief space via the distribution of likelihood amongst the candidate particles, but it also allows for increased robustness to occlusions and other natural environment difficulties.

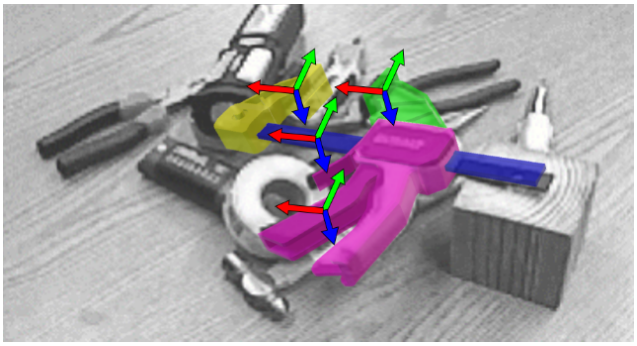

Figure 1. Example of pose-estimation task in cluttered environment.

In more technical terms, our proposal is as follows. Given a color and depth image of an object (which may potentially be articulated), the system will generate a latent embedding for a set of randomly sampled particles. This latent embedding is used to predict per-particle pose, object likelihood score, and an image space offset. The offset is used to select new particle patch/pixel inputs which are then re-evaluated to improve the likelihood estimate of the particle. After a set number of iterations, a final pose is extracted from the refined latent embedding as shown in Figure 1.

We intend to address three problems associated with current SoTA pose estimators - training time, robustness to clutter, and ability to reason about belief. Our proposed method should be more amenable to clutter and occlusions within the observation space. This improved robustness to image perturbations should also increase the scalability of pose estimation techniques by decreasing the computational and temporal requirements necessary to train a model on novel rigid-body objects, due to the decreased number of training data samples required to identify clutter agnostic features. Additionally, by utilizing a generative inference method, we would allow roboticists the ability to investigate the confidence of the estimator in various proposed poses, thus allowing for the use of higher quality manipulation techniques in downstream pipelines.

2. Related Work

There have been several works that demonstrate the ability of neural networks to predict the pose of rigid body objects in the RGB and RGB-D domains such as PoseCNN, DOPE, and PoseRBPF [7][9][2]. While these techniques are useful, they are all styled after an end-to-end neural network, and thus possess certain limitations. In particular, they all are of fixed input image size, possess zero or limited ability to reason about belief, and all require extremely large/diverse training datasets necessitating high training time. It is important to note that while PoseRBPF does allow for a certain amount of reasoning about belief, it only permits it within a small number of axes. This reasoning about belief, however, is extremely important when manipulating certain objects such as screwdrivers, pencils, etc. which all possess high degrees of radial symmetry, but whose pose about the symmetric axis may be extremely important to the objects' manipulation task. In contrast to these approaches, our generative inference based proposal maintains a joint belief across all six pose axes.

Other works such as NOCS seek to perform pose estimation on categorical instances of objects [8]. Our proposed method here does not explicitly address the notions of categorical pose estimation, and we leave its extension to a future project. However, the NOCS and other similar systems still retain the drawbacks inherent in end-to-end trained neural networks, including lack of ability to reason about belief, and large training dataset requirements.

There do exist some works that are explicitly targeted at performing belief space estimation of object pose [5]. Many of these works, however, maintain a reliance on a CAD model of the relevant objects, which not only hurts their ability to be extended to other objects, but also hinders their run-time performance (as manipulations of the CAD model can be computationally expensive or memory inefficient).

3. Proposed Method

We propose a collection of networks work in concert to both update their own latent state, as well as to make a final prediction. The proposed approach is analogous to a particle filter, with the likelihood and update functions being learned via neural networks as proposed in [3]. It represents a unique avenue of research allowing for the inference and training time speed benefits of neural network based approaches, along with the belief space reasoning abilities of historical generative inference approaches.

Our proposed set of networks work in three phases: initialization, refinement, and prediction. Given an image and a set of 2D coordinates we refer to as particles, the network initializes its internal latent parameters, then iteratively refines them by integrating the latent representations with sections of the image that particles point to. Through this process, a well taught system could navigate the image by carefully propagating particles and building an understanding of possible poses of objects particles encounter as well as those poses' their likelihoods. This pipeline is depicted in figure 3 and described in more detail below.

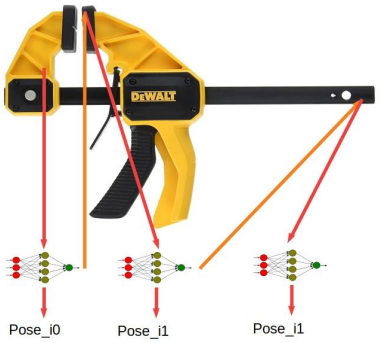

Figure 2. Example of pixel visitations for single particle across three iterations.

Initially, our networks are intialized with a zero latent vector and a zero offset for each particle. Given an RGB-D image and particles, the network offsets the particles and extracts a pixel or a patch from the image at the current particle locations. The extracted patch is concatenated with its associated latent vector and then passed through a multilayer perceptron (MLP) to produce an updated latent vector.

The new latent representations encode image features collected from different locations that the particle visited. Particle motion is not constrained allowing the latent vector to encode features of arbitrary Euclidean distance in pixel space. Each particle accumulates information from different locations linking image features in a form akin to a dynamic graph. Latent features are then processed by a pair of MLP layers to produce a pose and a likelihood estimate. All of the pose vectors, likelihood vectors, and their corresponding latent vector are concatenated and processed by a final MLP that predicts the best next offset. With knowledge of the object, this offset could direct a particle towards the next image location that would elucidate the predicted pose, or it could propel that particle into an undiscovered region if data indicates that an object is not likely in the investigated location. With that new offset prediction, the network is ready to undergo another iteration.

For a set number of iterations, the network is allowed to refine its latent information before a final set of pose and likelihood predictions are produced.

For training, pose estimates are compared to ground truth poses using ADD loss on transformed object meshes. YCB and FAT data sets provided object meshes and ground truth poses for objects in every image. Although ADD-S loss is superior for supervising pose estimation of symmetric ob-

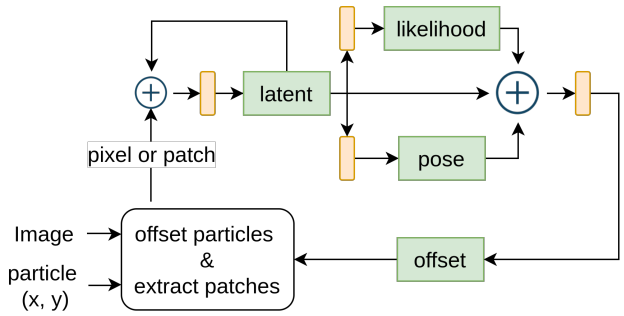

Figure 3. This image depicts the initial network architecture. Green blocks identify pieces of data , orange rectangles are multi-layer Perceptron networks, and the plus signs indicate concatenation. The network data is initialized to trivial values.

jects, ADD is sufficient for objects with a lower degree of symmetry and distinct color patterns. For training, an initial version of the network objects with no symmetric color patterns is used for simplicity.

4. Experiments

In this section, we will discuss the data we used, how we evaluated our system, and the different abstractions of particle filters that we have devised to find an object pose in an RGBD image. We have mainly experimented with three levels of particle filter which use information derived from pixels or patches of pixels.

The main rendition of the network used 1000 particles spread uniformly over the image. The particle filter is run for 10 iterations. A latent vector is set to have 256 parameters. All MLP networks were configured with 4 layers of 512 input and output dimensions each. Our MLP networks sequentially stacked linear, batch normalization, and ReLU for every added layer.

4.1. Data and Setup

We utilized two different datasets in an attempt to demonstrate the capabilities of our proposed method. We used the FaT (Falling Things) dataset from [6], which presents a collection of photorealistic rendered scenes from the YCB object set [1]. Those datasets were intended to serve as a starting point for the demonstration and initial buildup of our technique. Additionally, the dataset from [5] presented a more complex challenge which we wanted to use to show edge, failure, and outsized-success cases. The latter dataset contains significantly more clutter and is real-world data (as opposed to simulated), so it would be able to show practical generalizability as well.

Experiments were judged using three primary evaluation metrics. The first was parts-based bounding-box intersection-over-union (typically referred to as Jaccard Similarity). Secondarily, the ADD and ADD-S matching scores utilized in [7] were presented for comparison purposes. These scores represent the average 1-1 correspondence distances between the vertices of the object mesh at its ground truth and estimated poses. The primary difference between ADD and ADD-S scores being that ADD-S has a slightly different formulation to allow for idempotent poses to have the same output score. The equation for ADD is given in (1) and the equation for ADD-S is given in (2). While ADD-S would ultimately be used to accommodate the highly symmetric objects in our data sets, the simpler ADD was used in our initial tests on objects with distinct colors to further verify that color, as well as the depth, is contributing to our results.

$$m(P_{gt}, \hat{P}) = \frac{1}{\mathbb{N}} \sum_{(p_{gt}, \hat{p}) \in (P_{gt}, \hat{P})} ||\hat{p} - p_{gt}|| \tag{1}$$

$$m_{\text{sym}}(P_{gt}, \hat{P}) = \frac{1}{\mathbb{N}} \sum_{\hat{p} \in \hat{P}} \min_{p_{gt} \in P_{gt}} ||\hat{p} - p_{gt}|| \tag{2}$$

Lastly, we perform ablation studies on the effect of the number of particles initialized and the number of iteration for the particle filter versus the performance of our proposed approach.

4.2. Pose From Pixels

As the network can propagate its own particles around and accumulate information, we started by exploring the use of a single-pixel to inform our latent updates. At every iteration, a single RGB-D pixel was concatenated with the latent vector.

Results After training our network with this setup were not promising. A memorization experiment, using a single object category and testing on training data, showed that the network failed to converge to any meaningful pose. By plotting the progression of particles over time we noticed that all particles were moving in one direction and away from any object. The result was deemed random and hence encouraged some changes to the network.

4.3. Pose From Patches

Since surveying a pixel at a time failed to provide sufficient information, we hypothesized that integrating a patch at a time into the latent vector could improve results. We extract patches around the pixel locations at every iteration to achieve this objective. These patches are flattened and appended with pixel locations and prior latent vector information. The resultant vector was fed through the same MLP based pipeline. Patches of sizes 3×3, and 6×6 were tested, as bigger patches would likely demand an additional encoder layer to process and condense the patch into a feature. We initially train the network by maximising the intersection between the ground truth segmentation mask of

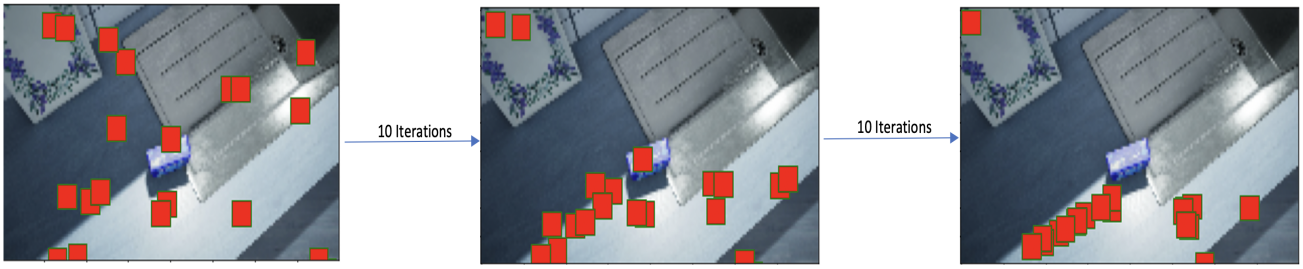

Figure 4. Patches of size 9×9 are initialized as particles for performing a detection task for the cracker box in the frame. A convolution network is used for detecting the offset at each iteration which is supervised using $l2$ distance between the particle and ground truth mask.

the object, M and the particle patch, P. We supervise the model use a binary cross entropy loss formulated as follows:

$$l_{bce} = \frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i)) \tag{3}$$

where,

$$y_i = \begin{cases} 1, & \text{if } i \in M \cap P \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

and $p(y_i)$ is the predicted likelihood of the patch.

While testing, the number of particles used was varied between 50 and 100. The particle filter was run for a 100 iterations to test each of the patch sizes. We evaluated models trained using both BCE loss and ADD loss to compare the meshes transformed using the ground truth and the predicted object poses.

After testing while using random initialization, we attempted to inform our sampling method while training. As demonstrated by [4] especially when the number of positive samples is small, sampling particles mainly on or around the objects of interest could greatly benefit learning.

Results We observe that overfitting experiments using BCE loss resulted in particles being offset towards image edges. We theorize this might be caused by the large imbalance between the positive and negative samples during loss calculation. The objective function learns to optimize by minimizing the loss contributed negative samples rather than maximizing the intersection between the ground truth mask and particle patch. To tackle that we tried to initialize a portion of the particles on or around the object but that did not render better results.

In addition, similar memorization experiments on a model trained using ADD loss were run to predict the pose of a scene with a single object and it was observed yet again that the network was not producing sufficient outcomes. The network acted unpredictably and learned to move particles in some random direction. Since we failed to see a pattern in the motion of the particles, we concluded that the image patches were still not a sufficient source of information.

4.4. Using an Image Patch as Particle Filter

After the failure of our previous experiments with particles initialized on pixels and patches, we decided to narrow down our experiment to just detect the object in the frame by running a small convolution neural network on patches of size 9×9. These patches can now be considered as particles initialized randomly in the pixel domain and a small convolution network with three Conv-Batchnorm-ReLU blocks are used to extract features and output three values – offsets (x and y) and likelihood.

For our experiment, we initialize 50 patches (particles) of size 9×9 and updated the offsets iteratively for 30 iterations to reach towards the object similar to previously mentioned experiments. To supervise the particles we used a variant of chamfer distance between the 9×9 particle patch and ground truth mask patch of the object. We define the chamfer distance for the two patches $P, Q \in \mathbf{R}^{N_* \times 4}$ where N_* is the number of pixels in the corresponding patch with $\Lambda_{P,Q} = \{(p, \arg\min_q \|p - q\|) : p \in P\}$ be the set of pairs (p, q) such that q is the nearest neighbor of p in Q, as:

$$\begin{aligned} l_{cham}(P, Q) = |P|^{-1} \sum_{(p,q)\in\Lambda_{P,Q}} \|p - q\|^2 + \\ |Q|^{-1} \sum_{(q,p)\in\Lambda_{Q,P}} \|q - p\|^2 \end{aligned} \tag{5}$$

The chamfer distance l_{cham} minimizes the distance between the closest neighbors in the two patches, effectively forcing the model to maximize the overlap between the two patches.

Results Similar to section 4.2 we ran a memorization experiment (using a single object category and testing on training data) to detect the object using the patches as particle filter with a shared convolution network to learn the offsets to travel toward the object boundaries. However, as

shown in figure 4, we saw that particles are not yet converging at the object in the frame. There are mostly two kinds of patterns that we are seeing with this method: 1) Either the particles line up along an arbitrary axis 2) or they get accumulated at a corner of the frame.

4.5. Downsized Image for Patch as Particle Filter

After observing those unsatisfactory particle motion pattern in section 4.4 while using a 9×9 convolution kernel we decided to examine whether having our kernel consume a larger portion of the object would facilitate its ability to learn how the object looks and thus its ability to better characterize it. While increasing the complexity of the feature extractor and kernel size could serve a similar purpose, we elected to conserve our kernel size to maintain a reasonable run-time. Images downsized to 160 resolution were tested using the same pipeline detailed in 4.4. We limited our test set to a single category (cheez-it crackers boxes) which has distinct color patterns thus sparing us from having to deal with issues associated with highly symmetric objects. We tested the networks ability to detect the object using the chamfer distance loss as outlined in 5.

Results Although we effectively increased the receptive field of the patch with some information loss due to resizing, the model gave similar results as those in section 4.4.

5. A New Methodology: Pose-Space Learned Likelihood Particle Filter

In the interest of attempting a new approach that may yield something approximating success, a new learned-likelihood particle filter method was developed inspired by occupancy networks [4].

In this new method, each particle represents a pose hypothesis. Each pose hypothesis coordinate frame is projected into the u,v space of the image and a series of randomly selected u', v' pixels are selected from a two-dimensional Gaussian distribution centered around the proposed pose. These pixels are fed through the camera projection matrix to yield a series of points $\begin{pmatrix} x & y & z & r & g & b \end{pmatrix}_{world}$ world, which is then multiplied by the proposed pose's transform to yield a point $\begin{pmatrix} x & y & z & r & g & b \end{pmatrix}_{object}$ which represents a hypothetical observation within the relevant object's coordinate frame. This $\begin{pmatrix} x & y & z & r & g & b \end{pmatrix}_{object}$ is then fed into a neural network model M whose architecture is described in table 5. This neural network then returns a scalar value which represents the likelihood of that pixel/point's observation conditioned on the proposed pose. The likelihood is then taken to be the sum of all likelihoods for each $\begin{pmatrix} x & y & z & r & g & b \end{pmatrix}_{object}$ point. This function is otherwise couched within a typical re-sample/perturb/re-weight particle filter loop with 1000 candidate particles/poses and ran for 100 iterations.

Layer #	Layer Type	Input Size	Output Size
1	Linear	6	128
2	ResnetFC	128	128
3	ResnetFC	128	128
4	ResnetFC	128	128
5	ResnetFC	128	128
6	ResnetFC	128	128
7	Linear	128	1

Table 1. Table describing the layers used in the learned likelihood particle filter model

To train the network, 30 $\begin{pmatrix} x & y & z & r & g & b \end{pmatrix}_{object}$ samples are taken along the epipolar ray for each segmentation map pixel member within the original FAT training data-set, with the starting location being placed 10cm from the camera's focal plane, and the ending location being placed 5cm beyond the observed pixel/point. Generated $\begin{pmatrix} x & y & z & r & g & b \end{pmatrix}_{object}$ samples farther than 1cm away from object's surface towards the observer is assigned random r,g,b values and are assigned a label of 0.0, as these are deemed to be possible occlusions, and therefore neither contribute to nor falsify the proposed pose. Generated $\begin{pmatrix} x & y & z & r & g & b \end{pmatrix}_{object}$ samples farther than 1cm away from the object's surface away from the observer are also assigned random r,g,b values, but are assigned a label of -100.0, as these interior points are not valid occlusions, and therefor falsify the proposed pose. Samples within $\pm$ 1cm of the objects generate 2 new training instances. In one instance, the observed r,g,b value is retained with a label of +100.0 (indicating a highly likely observation). For the other instance, a training sample is generated with a random r,g,b value with a label of 0.0, as such an observation may be considered a valid occlusion. The resulting model M is analogous to an occupancy network, but instead of simply returning $\pm$ 1.0 depending on if a point lies within the object bounds, it instead indicates if a point is a confirming or falsifying point cloud observation.

5.1. Results

Unfortunately, this approach did not yield a positive result - the final maximal likelihood poses did not correspond to the expected ground truths even remotely close enough to justify the reporting of ADD or ADD-S metrics. This was confirmed to be due to the lack of fine-tuning of the likelihood model. Although hand inspection of individual points on a sample scene produced values as to be expected (see figure 5), for some test data scenes, the MVE found by the particle filter was discovered to have a higher likelihood than the ground truth value. It is likely that a more principled approach to training the likelihood model could yield

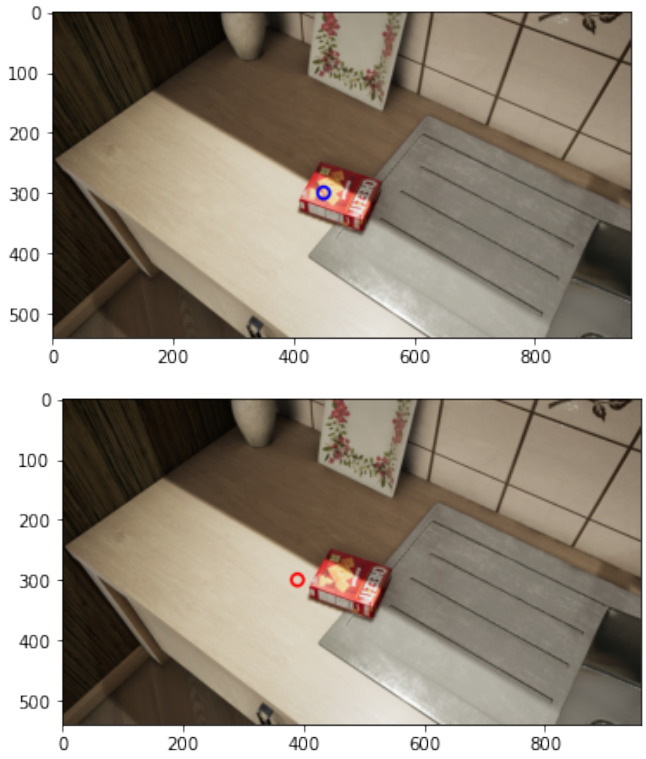

Figure 5. A pair of example samples of the likelihood field output. Subfigure 5 indicates a point on the surface of the object, which yielded a likelihood value of +95.77. Subfigure 5 indicates a point not on the surface of the object, which yielded a likelihood value of -115.43.

fruit - one the forgoes human tuned labels in favor of learned or inferred ones. However, such an approach was perhaps beyond the scope of this exercise's time constraints.

6. Future Direction

From our current set of experiments, it has become evident that it would be very difficult to estimate the pose of the object just by using solely the local context of the particles. Since our particle placement is stochastic in nature while placing them on the pixel plane, it becomes very difficult for the particle to estimate the pose let alone detect the object. Currently, all of our experiments are unsuccessful to estimate pose using particle filter in pixel domain. However, in the future we are thinking of pursuing other directions to estimate pose using particle filter:

- Using global context as used in [2] followed by particle sampling to have the right mix of both global and local context.
- Separate the pipeline and train it in parts. By training the likelihood and pose detection networks on pre-labeled segmentation inputs, a greater exposure of known-good labels should improve the later end-to-end training process. Additionally, having a known-good local likelihood function should improve the ability for the motion update network to detect object pixels in the relevant region of the image.
- Using a portion of a pre-trained ResNet to extract features from our image patches. Our latent vector, particle motion predictor, and likelihood predictor critically rely on retrieving descriptive features. Relying on a well-tested encoder could improve our ability to train the other portions of the network.
- Augment re-sampling with a global prediction of region relevance. We saw that particles that away from the object itself lack the global context that would inform their motion and guide them to the object. If particles were re-sampled based on how likely they are to predict an object pose or how likely they are to be sampling features from the object, we could avoid having to deal with training random particles to find the object of interest.

References

[1] 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. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

[2] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. Poserbpf: A rao-blackwellized particle filter for 6d object pose tracking. *arXiv preprint arXiv:1905.09304*, 2019.

[3] Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. Differentiable particle filters: End-to-end learning with algorithmic priors. *arXiv preprint arXiv:1805.11122*, 2018.

[4] Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *CoRR*, abs/1812.03828, 2018.

[5] Jana Pavlasek, Stanley Lewis, Karthik Desingh, and Odest Chadwicke Jenkins. Parts-based articulated object localization in clutter using belief propagation. *arXiv preprint arXiv:2008.02881*, 2020.

[6] Jonathan Tremblay, Thang To, and Stan Birchfield. Falling things: A synthetic dataset for 3d object detection and pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2038–2041, 2018.

[7] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018.

[8] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019.

[9] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.