6D Pose Estimation using RGB-D

Anushka Datta, Harsh Sharma and Sachit Mahajan

Abstract

Our goal is to accurately estimate the 6DOF poses of the objects using RGB-D input. We survey different methods for object pose estimation and the different steps involved in the pipelines. We use an end-to-end network that takes in an RGB-D image and outputs a 6D pose of the object. We train and test the pipeline proposed in MaskedFusion [3] and evaluate the accuracy at different stages in pipelines and replace certain parts of the pipeline with other deep and geometric components and evaluate. We mainly experiment on the pose refinement step part of the 6DOF pose estimation pipeline as it is a key part of accuracy. We then demonstrate that just the pipeline refinement part of pose estimation can be used to track the objects assuming small motion.

Motivation

Accurately estimating the pose of objects is necessary for robots to gain information about the scene. A lot of current pose estimation methods are developed to assist robotic arms in manipulation tasks. Our motivation is the development of recent SLAM methods that use objects for odometry that require 6D pose estimation. Unlike manipulation tasks, the environment for a robot performing SLAM has a lot of noise, variations in lighting, dynamic objects, etc. The accuracy of pose estimation networks can severely affect the performance of the SLAM pipeline. This is also one of the reasons we still want to use depth, unlike newer works that operate on only RGB data, as depth-based refinement has shown more accurate results especially in cases of occlusion and noise. We want to learn what components in the object pose estimation pipeline can be tweaked to make a system that can be used robustly for SLAM/spatial-AI.

Introduction and Prior Work

The object 6DOF pose estimation pipeline mainly involved three parts as shown in Figure 1, namely

- Segmenting the objects in the image

- Estimating the 6 degree of freedom pose for the objects
- Refining the 6DOF pose output from the network

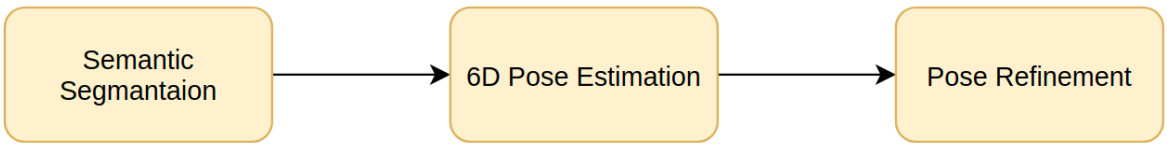


Figure 1. 6DOF pose estimation pipeline

Most of the existing methods perform the first two components but skip on the third which leads to a slight loss in accuracy. Some methods rely on geometric methods such as Iterative Closest point [5][3] for the refinement step, while some do use neural networks for it [5][1].

RGB vs RGB-D: When both RGB images and depth images are available, they can be combined to improve 6D pose estimation. A common strategy is to estimate an initial pose of an object based on the color image and then refine the pose using depth-based local refinement algorithms such as ICP. The performance of RGB-based methods is still not comparable to that of the RGB-D-based methods, this performance gap is largely due to the lack of an effective pose refinement procedure using RGB images only. It is also shown that using depth and RGB pixels for initial pose estimation is more robust than just RGB which rely on a template or feature matching and tend to fail in case of occlusions.

The semantic segmentation part of the pipeline is common for both RGB and RGB-D methods. The semantic segmentation goal is to classify each pixel in an image. In some methods, instance segmentation is used where the goal is to detect, delineate the object with a bounding box, and segment or create a mask for each object instance present in the image.

The 6D Pose estimation varies method by method and relies on template matching and feature matching. Most RGB methods detect key points on objects then rely on classical methods such as PnP[16] to estimate the 6DOF Pose. Deep learning-based methods use CNN to extract features and then match them during the inference phase. PoseCNN[11] uses RGB images and a novel convolutional neural network for end-to-end 6D pose estimation. It uses semantic labeling and regresses on center direction(in x and y) and distance then uses hough voting to vote object center direction and then estimate object pose. It also introduces a new loss function that is robust to object symmetry to directly regress the object rotation. PVNET[6] is a hybrid method that is based on key points. The voting is not only for object center but also 2D keypoint locations to get keypoint maps which are then proceeded by PnP to get the object pose. Keypose [8] predicts a probability map and then uses 3D model fitting over the map. Pix2Pose [7] no longer uses key points as discrete but normalized continuous coordinate maps. Another method DOPE [12] uses a one-shot, deep neural network-based system that infers, in near real-time, the 3D poses of known objects in the clutter from a single RGB image without requiring post-alignment by detecting keypoints using a multistage architecture and then using PnP. The main contribution is the training on synthetic data and performing heavy augmentation by using a combination of domain randomized and photorealistic data.

The RGD-D methods incorporate depth information while doing the pose estimation. PointFusion[4] fuses geometric and appearance information in a heterogeneous architecture. DenseFusion[5] presents a principled way to combine color and depth information from the RGB-D input and augment the information of each 3D point with 2D information from an embedding space

learned for the task and use this new color-depth space to estimate the 6D pose. MaskedFusion[3] fuses 3D data to 2D appearance feature while retaining the geometric structure of the input space.

The refinement step is an iterative process of increasing the accuracy of predictions. Most RGB-D methods only use the depth data in the refinement phase. PoseCNN[11] uses a heavily customized ICP to get really good results. DenseFusion[5] integrates an iterative refinement procedure within the neural network architecture, removing the dependency of previous methods of a post-processing ICP step. MaskedFusion uses the same refinement step. The refinement step requires high computation and slows down the pipeline, the use of a neural network for this task further slows down the framework at the training stage but is faster than ICP during inference.

As aforementioned, the pose refinement step is the main difference between RGB and RGB-D methods. However, one of the newer RGB methods DeepIM [1] performs RGB pose refinement and is able to bridge the gap between RGB and RGB-D methods.

We realised that this refinement step is a key part of accurate 6DOF pose estimation, hence we used the MaskedFusion pipeline and performed experiments by changing this step. We will go into further on the methods used (MaskedFusion, ICP and DeepIM) and experiments performed.

Used Methods

DenseFusion and MaskedFusion

DenseFusion is a generic framework for estimating 6D pose of a set of known objects from RGB-D images. DenseFusion is a heterogeneous architecture that processes the two data sources individually and uses a novel dense fusion network to extract pixel-wise dense feature embedding, from which the pose is estimated. It also integrates an end-to-end iterative pose refinement procedure that further improves the pose estimation achieving near real-time inference.

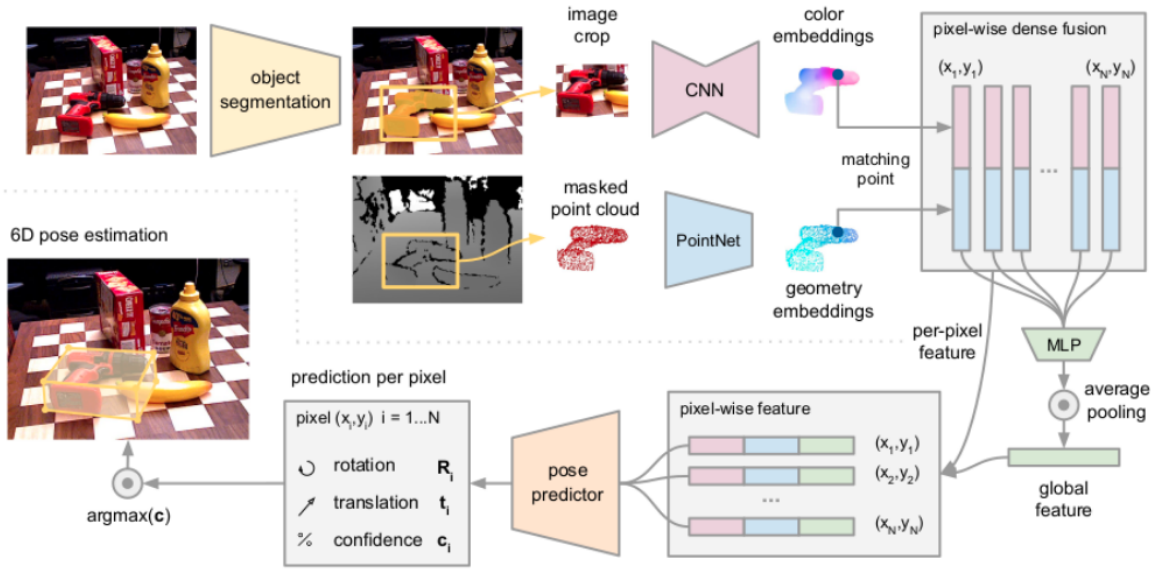


Figure 2. Overview of the DenseFusion model

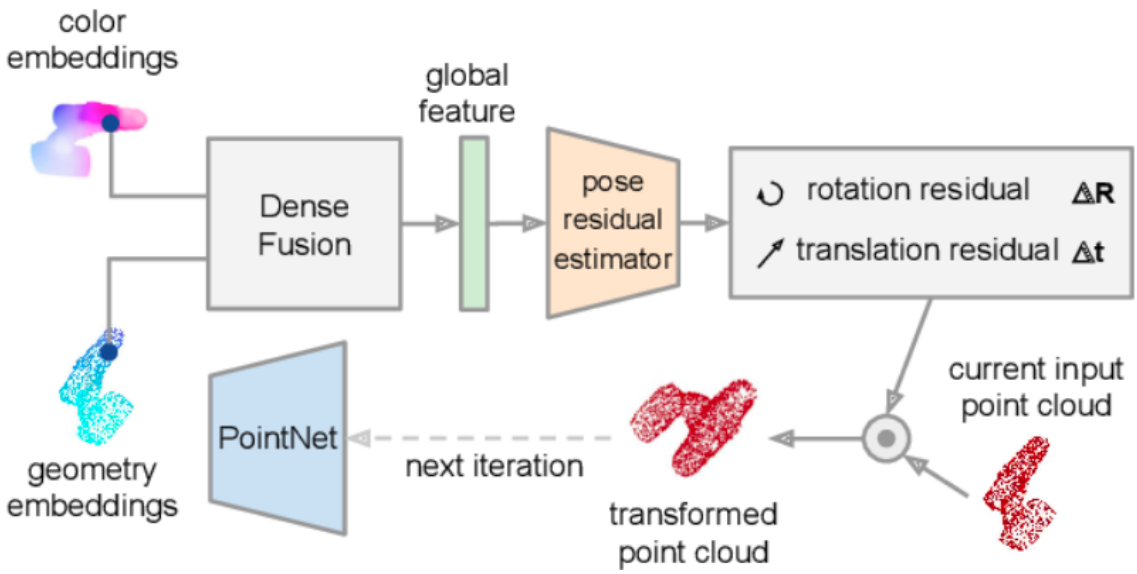


Figure 3. Iterative Pose Refinement

Architecture

The first stage performs semantic segmentation on the RGB image. For each identified object, the masked depth pixels(as 3D point clouds) and image crop are used for further processing.

The second stage processes these results to estimate the 6D pose. It comprises four components:

1. a fully convolutional network that processes the color information and maps each pixel in the image crop to a color feature embedding,
2. a PointNet-based network that processes each point in the masked 3D point cloud to a geometric feature embedding,
3. a pixel-wise fusion network that combines both embeddings and outputs the estimation of the 6D pose of the object based on an unsupervised confidence scoring, and
4. an iterative self-refinement methodology to train the network in a curriculum learning manner and refine the estimation result iteratively.

Figure 2 depicts stages 1, 2, 3 while Figure 3 depicts stage 4.

We focused on MaskedFusion which utilises the same basic architecture and performs better than the original DenseFusion(pipeline shown in Figure 4). In addition to using RGB and depth information, DenseFusion also takes advantage of the binary mask given by semantic segmentation when estimating the 6D pose. It combines steps 1 and 2 of DenseFusion into a single task.

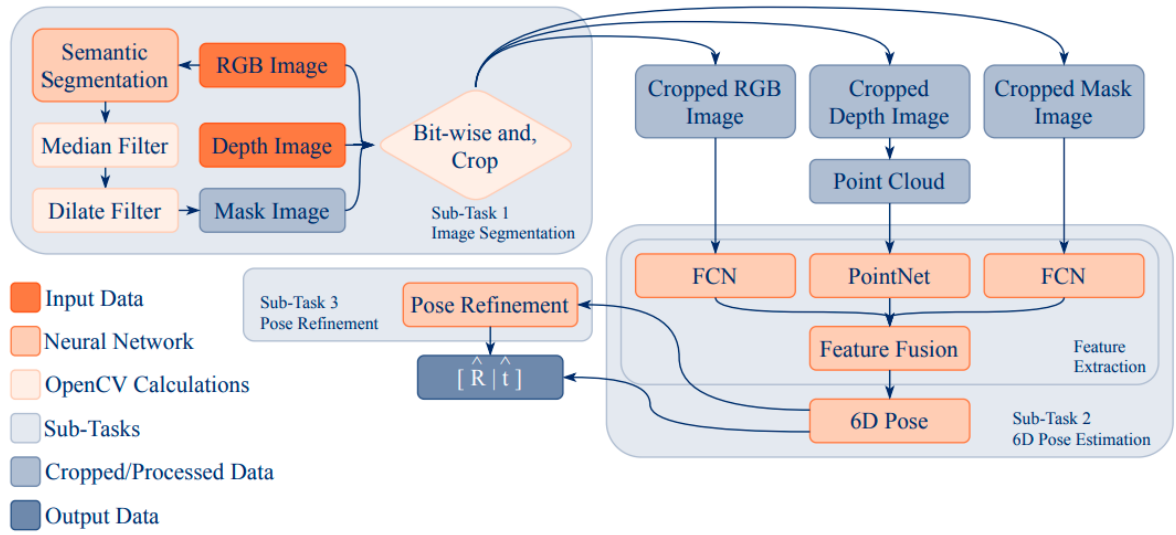


Figure 4. Pipeline diagram of the MaskedFusion architecture

Task 1: Image Segmentation

Semantic segmentation is performed on the RGB image to detect and extract a mask of each object present in the scene. This uses an FCN with encoder-decoder architecture similar to Segnet[15] as shown in Figure 5.

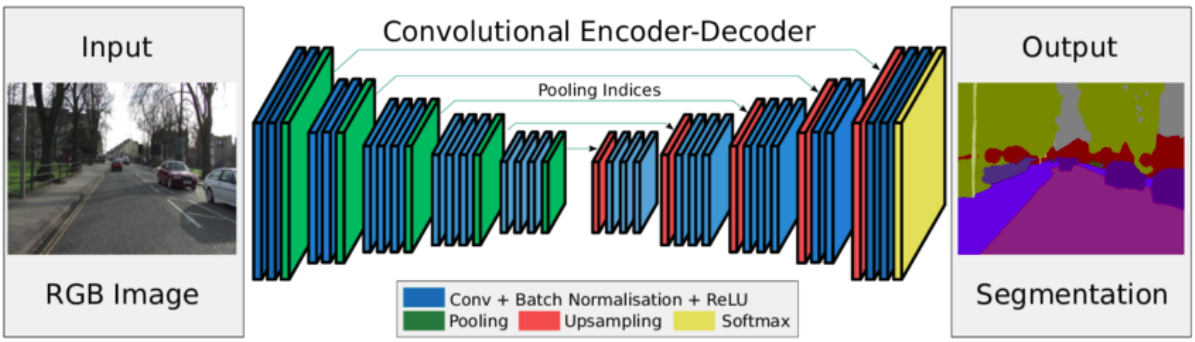


Figure 5. Segnet architecture

In this implementation of semantic segmentation, after obtaining the output mask, 2 additional filters are applied on the mask before feeding it forward to the next sub-task. First, a median filter from OpenCV is used to smooth the image with a kernel size of 3×3. Second, the mask is dilated with a 5×5 kernel such that, if the mask has minor boundary segmentation error, this operation helps to correct it.

The binary masks obtained from semantic segmentation with filters applied are used to crop the RGB and depth images. The images are cropped by applying a bit-wise AND between the RGB/depth image and the mask. This is done for each object in the scene. For the RGB image, the crop is a rectangular crop containing only the object. For the depth image, a point cloud is generated from the masked and cropped depth image. Cropping the data with the mask is a pre-processing of the data that helps the 6D pose NN on the second sub-task because it discards the background or other non-relevant data that are around the object leaving only the data that is most relevant to the 6D pose estimation.

Task 2: 6D Pose Estimation

The 6D pose NN used by MaskedFusion is inspired by DenseFusion in that it has a similar architecture where it extracts features from different types of data and fuses the information in a pixel-wise manner. The 6D pose NN can be divided into 2 stages:

1. Feature extraction: All cropped data is fed into separate NNs that extract features for each kind of data. For the point cloud, PointNet is used to extract 500 features that represent depth information of the image. For the masked RGB image, a FCN based on ResNet-18 is used to extract 500 features. The binary mask image is also passed through a similar FCN but has only one channel in the input. This extracts 500 features for the binary mask. Features from the mask represent the shape of the object which adds relevant information for the next stage and improves accuracy.
2. 6D pose estimation: The 3*500 extracted features are concatenated into a single vector and passed through a convolutional layer to fuse all the features. A neural network receives these features and regresses the 6D pose of the object(rotation matrix and translation vector of the object). The neural network has 2 paths, one to regress the translation vector t, and the other to regress the rotation matrix R. The loss function is the same as the one used in DenseFusion(ADD):

$$\mathcal{L}_i^p = \frac{1}{M} \sum_j \left\| (Rx_j + t) - (\hat{R}_i x_j + \hat{t}_i) \right\|$$

where xj denotes the jth point of the M randomly selected 3D points from the object's 3D model, p = [R|t] is the ground truth pose, where R is the rotation matrix of the object and t is the translation. The predicted pose generated from the fused embedding of the ith dense-pixel is represented by p̂ i = [R̂ i | t̂ i] where R̂ denotes the predicted rotation and t̂ the predicted translation. After training the 6D pose NN, the output of it (p̂ i = [R̂ i | t̂ i]) can be retrieved after the second stage or it can be sent to the next sub-task of the pipeline.

Task 3: Pose Refinement

MaskedFusion uses the pose refinement developed in DenseFusion. The pose refinement neural network improves upon the 6D pose previously estimated. The authors of DenseFusion came up with iterative pose refinement because even the fastest implementations of Iterative Closest Point are often not efficient enough for real-time applications. They proposed a neural network based iterative refinement that improves the final pose estimation in a fast and robust manner.

The goal is to enable the network to correct its own pose estimation error in an iterative manner. The challenge here is training the network to refine the previous prediction as opposed to making new predictions. To do so, we must include the prediction made in a previous iteration as part of the input to the next iteration. The key idea is to consider the previously predicted pose as an estimate of the canonical frame of the target object and transform the input point cloud into this estimated canonical frame. This way, the transformed point cloud implicitly encodes the estimated pose. We then feed the transformed point cloud back into the network and predict a residual pose based on the previously estimated pose. This procedure can be applied iteratively and generate potentially finer pose estimation each iteration.

A dedicated pose residual estimator network is trained to perform the refinement given the initial pose estimation from the main network. At each iteration, the image feature embedding from the

main network is reused and dense fusion is performed with the geometric features computed for the new transformed point cloud. The pose residual estimator uses as input a global feature from the set of fused pixel features. After K(we use K=4) iterations, we obtain the final pose estimation as the concatenation of the per-iteration estimations:

$$\hat{p} = [R\ K\ |t\ K\ ] \cdot [R\ K{-}1\ |t\ K{-}1\ ] \cdot \cdots \cdot [R\ 0\ |t\ 0\ ]$$

The pose residual estimator can be trained jointly with the main network. However, the pose estimation at the beginning of the training is too noisy for it to learn anything meaningful. Thus in practice, the joint training starts after the main network has converged.

Deep Iterative Matching

DeepIM(as shown in Figure 6) follows an iterative rendering-based approach to the alignment problem. The network is trained to predict a relative SE(3) transformation that can be applied to an initial pose estimation for iterative pose refinement. Given a 6D pose estimation of an object, which can be the output of other pose estimation methods like MaskedFusion's PoseNet in our case, a rendered image can be generated showing the appearance of the target object under this rough pose estimation. The network takes in the image pairs of rendered images and observed images, to predict a relative transformation (Δpose in the figure) which can be applied to refine the input pose.

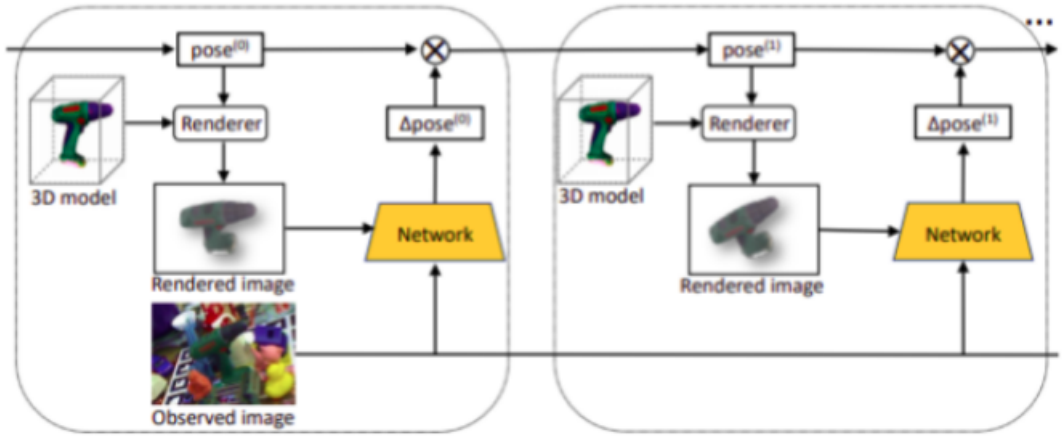


Figure 6. DeepIM Framework: Given a 3D model and estimated pose, network can iteratively refine the pose given the rendered and actual image pair

Iterative closest point (ICP)

Iterative closest point (ICP) is an algorithm employed to minimize the difference between two point clouds. It performs local registration and is used in pose refinement. We mostly focus on point-to-point ICP; the algorithm iteratively revises the transformation (translation and rotation) to minimize the sum of squared differences between the coordinates of the matched pairs of two point clouds.

The problem with ICP is that it is not very robust and highly dependent on initialization. Being iterative, it has a high compute requirement and can also get stuck in local minima. However, it is highly generalisable and can be used without any training. Deep learning-based pose refinement is usually not as accurate as ICP and has a longer training time but has shorter inference time. We used Open3D[14] implementation of point-to-point ICP.

Dataset and Metrics

We evaluate the network on 2 datasets

1. YCB-Video Dataset[11]: The dataset contains 92 RGB-D videos, where each video shows a subset of the 21 objects in different indoor scenes. The videos are annotated with 6D poses and segmentation masks
2. LineMOD Dataset[10]: This is one of the most used datasets to tackle the 6D pose estimation problem. LineMOD has 15 low-textured objects and over 18000 real images and has the ground truth pose annotated. Each object also contains the 3D model saved as a point cloud and a distance file with the maximum diameter (cm) of the object.

We used the Average Distance of Model Points (ADD) and Average Closest Point Distance (ADD-S) for evaluation for non-symmetric and symmetric objects respectively. The metric is computed as the mean of the pairwise distances between the 3D model points of the ground truth pose and the estimated pose.

$$\text{ADD} = \frac{1}{m} \sum_{x \in M} \left\| (Rx + t) - (\hat{R}x + \hat{t}) \right\|$$

$$\text{ADD-S} = \frac{1}{m} \sum_{x_1 \in M} \min_{x_2 \in M} \left\| (Rx_1 + t) - (\hat{R}x_2 + \hat{t}) \right\|$$

Where [R,t] is ground truth transformation and [R^, t^] is estimated transformation. M model with m points.

Experiments and Results

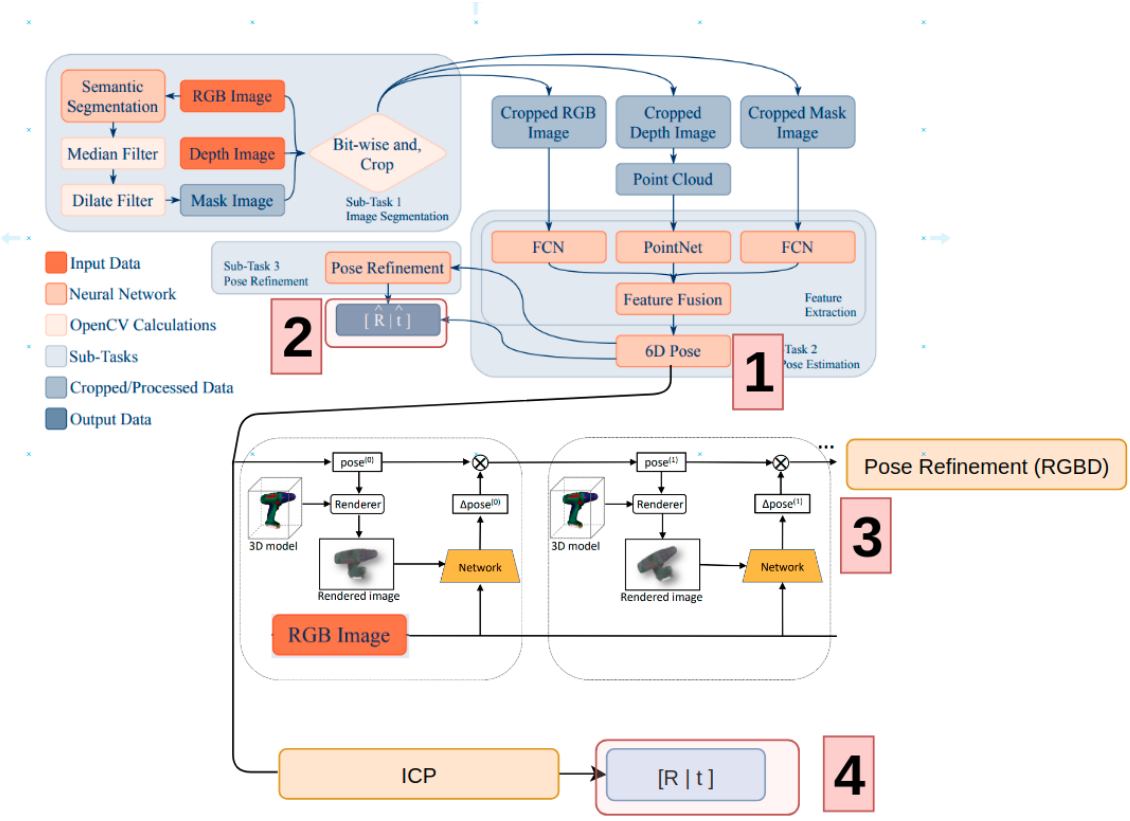


Figure 7. Experiments conducted on the refinement part

We mainly experimented with the refinement part of the pipeline(as shown in Figure 7).

1. Evaluation on MaskedFusion without the pose refinement step
2. Evaluation of MaskedFusion with pose refinement
3. Evaluation with replacing the MaskedFusion RGB-D refinement with DeepIM's RGB and model refinement
4. Using point-to-point ICP for refinement instead of deep methods

We analyse the cases where the above evaluations fail and the probable reasons. For the LineMOD dataset, we also add another ICP after the 2nd evaluation to check if final refinement is the reason when the method is unable to get accurate results.

Training: We trained the MaskedFusion for both the datasets. We trained MaskedFusion on LineMOD for 100 epochs which took ~3 days and on YCB-Video Dataset for 80 epochs which took about a week. The models were trained without the refinement step until they converged and then the refinement step was added and trained. For the DeepIM we used the pretrained version.

Since the datasets are quite different, we will present the results separately. YCB is evaluated using the scripts provided by PoseCNN and LineMOD is evaluated using the point cloud generated and results are visualized. Due to the separate training of both, the results and trends vary. We added scripts to visualize the point clouds on objects at different steps of pipeline.

YCB

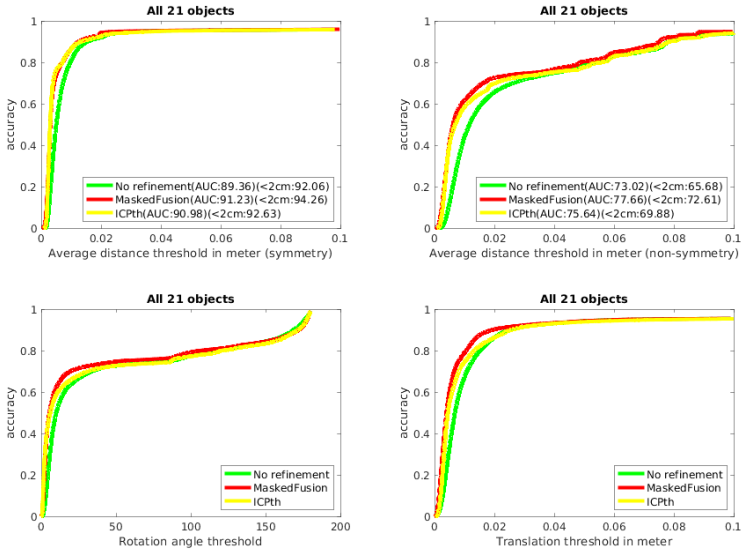


Figure 8. Results of using MaskedFusion with no refinement, with 4 iterations of refinement, and with ICP refinement on YCB Dataset

The MaskedFusion model was trained with 96189 data points and tested on 2949 data points. Figure 8 shows the performance of the baseline model with no refinement as compared to MaskedFusion and ICP. MaskedFusion gives the best results, while ICP gives some improvement over the baseline. We observed that ICP needed a better baseline to perform well. This can be observed by the graphs in Figure 9 that show the performance of the three models on images containing bowl items.

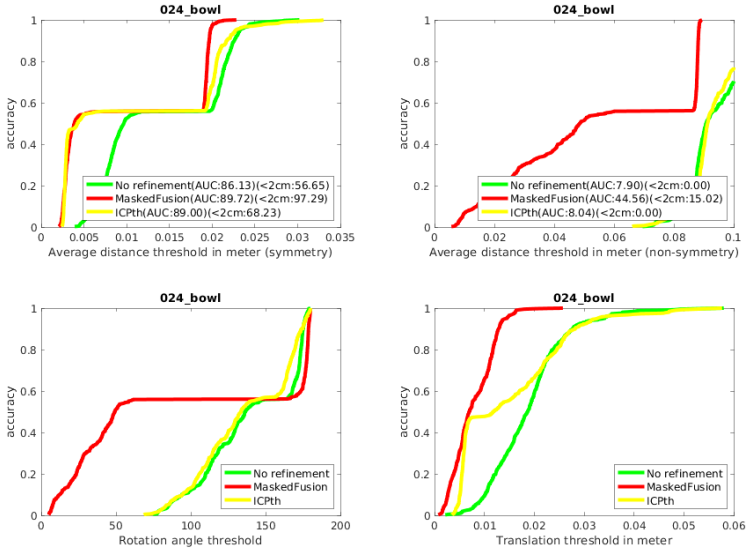


Figure 9. Results of using MaskedFusion with no refinement, with 4 iterations of refinement, and with ICP refinement on class 24 of the YCB Dataset

Implementation challenges with DeepIM-

Deep IM is trained on the YCB-Video dataset. We take a pre-trained model trained on YCB video. However, the implementation is tightly coupled with the data loader, and a lot of image manipulations including zoom-in and crop on the detected object and upsampling are done in the data loader itself. These are required downstream by the image renderer. The renderer also requires GUI support and hence the implementation challenges make it harder to work with a custom network.

LineMOD

Run for 1335 evaluation images. DeepIM was not trained on LineMOD

Results LineMOD

Sno.	Experiment	Average ADD/ADD-S	ADD/ADD-S<5cm	Percent ADD/ADD-S<diameter
1	Masked Fusion, **No** refinement	0.064	0.39	0.09
2	Masked Fusion, with **refinement**	0.05	0.99	0.979
3	Masked Fusion, with **ICP**	0.0055	0.56	0.29
4	Masked Fusion, **refinement + ICP**	0.0053	0.99	0.982

The results using no refinement and ICP are surprisingly low. It is also very different from what we observe for YCB-Video Dataset. The MaskedFusion paper did not evaluate in this way so we do not have a baseline to compare against, however, the post-refinement results match those reported in the paper. On visualising, we notice that the initial no refinement prediction of pose is rotated farther away from the target for images in the LineMOD Dataset. This could be because we trained for fewer iterations since we were getting the expected results for the full pipeline. The pose refinement was so robust that it worked on worse initial estimations. Here, we observe one of the downsides of using ICP, as without a good initial pose it failed to converge accurately. The MaskedFusion refinement took very long to train but produced good results faster than ICP. We added another ICP after the MaskedFusion refinement and observed that we were able to improve the performance by a very small number and in Figure 14.

The few cases where MaskedFusion failed were either because of very inaccurate initial estimations, but mostly because of symmetric objects viewed from ambiguous angles. This was the case for Object 13(Iron) and Object 5(Flowering Pot) as seen in Figure 10. In one case the failure was because of the object being heavily occluded, though in many cases the method worked even in presence of occlusions demonstrating the advantage of using depth. The failures can be seen below

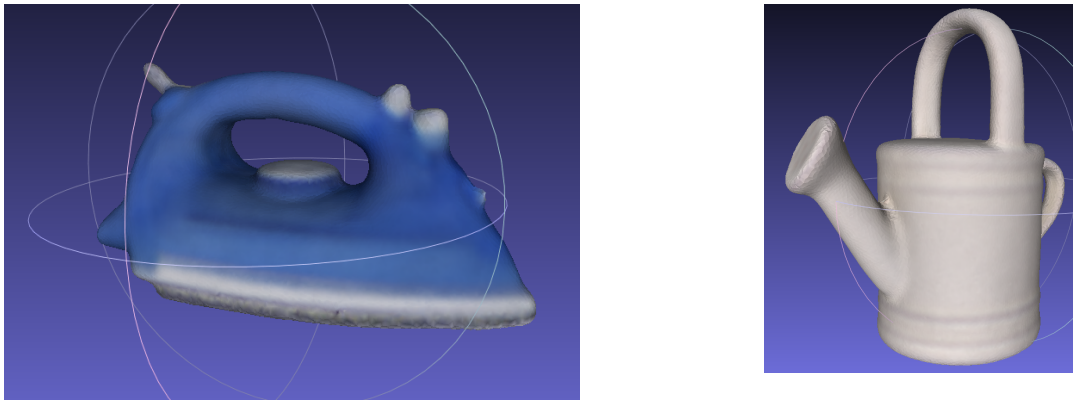


Figure 10. Difficult Objects in LineMOD

We visualise the point clouds generated by the different models against the ground truth for a few images in the LineMOD dataset.

Figure 11 compares the result of MaskedFusion without the refinement step, MaskedFusion with refinement and ground truth. The blue depicts the point cloud from the initial pose from Masked Fusion(before refinement), yellow depicts the pose after refinement, and cyan depicts the target ground truth point cloud.

Figure 11. Results on LineMOD for MaskedFusion baseline, using MaskedFusion refinement and ground truth

Aa Name	Created	Tags
Click To Expand	@May 15, 2021 4:47 PM	

Figure 12 compares the result of MaskedFusion without the refinement step, MaskedFusion baseline with ICP refinement, and ground truth. The blue depicts the point cloud from the initial pose from Masked Fusion(before refinement), the yellow depicts the pose after ICP refinement, and cyan depicts the target ground truth point cloud.

Figure 12. Results on LineMOD for MaskedFusion baseline, using ICP refinement and ground truth

Aa Name	Created	Tags
Click To Expand	@May 15, 2021 4:53 PM	

Figure 13 shows a few instances where MaskedFusion fails.

Figure 13. Results on LineMOD where MaskedFusion failed

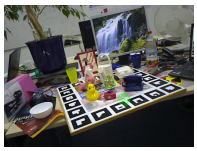
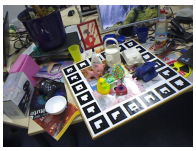
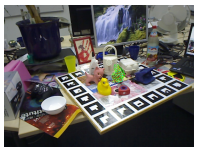
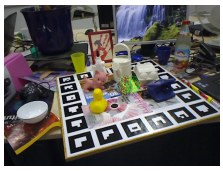
Aa Name	Created	Tags
Click To Expand	@May 15, 2021 4:55 PM	

Figure 14 shows an image that MaskedFusion did not perform well on but ICP was able to correct and give a better result.

Figure 14. Example of MaskedFusion failed case corrected by adding another ICP

The refinement step of MaskedFusion was very robust, this gave us an that it can be used for object tracking where initially the whole pipeline is run and then only refinement step is run for subsequent frames assuming small motion. This resulted in robust accurate object tracking in 3D, and it was be fast since the refinement step is surprisingly fast during testing. The results can be seen in Figure 15. (GIF). The green point cloud represents the final refined pose from previous frame and is fed as initial pose to pose refinement, the refined point cloud is yellow and the ground truth is cyan.

Figure 15. Tracking over multiple images using just the refinement step

Conclusion and Future Work

From the results it can be observed that the refinement step is very important for accurate 6DOF pose estimation. While ICP is generalizable, it is highly dependent on initialization. An end-to-end deep learning algorithm for 6DOF pose estimation which does semantic/instance segmentation, followed by pose estimation and finally an iterative pose refinement has the best results.

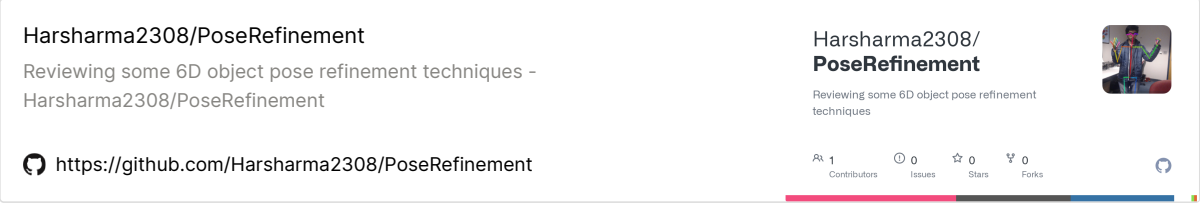
One improvement that can be done is the use of RGB embeddings in pose refinement along with depth. Something that combines DeepIM and MaskedFusion's approach such that refinement works when the point cloud is noisy or when the RGB image has occlusions.

Another improvement can be training the networks on Occlusion LineMOD making the results even more robust.

Video

https://youtu.be/MsowiDfyNdM

Github Link



References

1. Li, Yi, et al. "Deepim: Deep iterative matching for 6d pose estimation." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
2. DeepIM implementation - https://github.com/NVlabs/DeepIM-PyTorch
3. Pereira, Nuno, and Luís A. Alexandre. "Maskedfusion: mask-based 6d object pose estimation." *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2020.
4. Xu, D., Anguelov, D., Jain, A.: Pointfusion: Deep sensor fusion for 3d bounding box estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 244–253 (2018)
5. Wang, Chen, et al. "Densefusion: 6d object pose estimation by iterative dense fusion." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.
6. Peng, S., Liu, Y., Huang, Q., Zhou, X., & Bao, H. (2019). PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation. In CVPR.
7. Kiru Park and Timothy Patten and Markus Vincze (2019). Pix2Pose: Pixel-Wise Coordinate Regression of Objects for 6D Pose Estimation. CoRR, abs/1908.07433.
8. Xingyu Liu and Rico Jonschkowski and Anelia Angelova and Kurt Konolige (2019). KeyPose: Multi-view 3D Labeling and Keypoint Estimation for Transparent Objects. CoRR, abs/1912.02805.
9. Jonathan Tremblay and Thang To and Balakumar Sundaralingam and Yu Xiang and Dieter Fox and Stan Birchfield (2018). Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects. CoRR, abs/1809.10790.
10. Hinterstoisser, S., Holzer, S., Cagniart, C., Ilic, S., Konolige, K., Navab, N., Lepetit, V.: Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In: 2011 international conference on computer vision. pp. 858–865. IEEE (2011)
11. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. arXiv preprint arXiv:1711.00199 (2017)
12. Tremblay, Jonathan, et al. "Deep object pose estimation for semantic robotic grasping of household objects." *arXiv preprint arXiv:1809.10790* (2018).
13. ICP Point to Point: https://github.com/ClayFlannigan/icp
14. Qian-Yi Zhou, Jaesik Park, & Vladlen Koltun (2018). Open3D: A Modern Library for 3D Data Processing. arXiv:1801.09847.

15. Badrinarayanan, Vijay, et al. “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation.” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 12, 2017, pp. 2481–95, arxiv.org/abs/1511.00561.
16. Y. Zheng, Y. Kuang, S. Sugimoto, K. Åström and M. Okutomi, "Revisiting the PnP Problem: A Fast, General and Optimal Solution," 2013 IEEE International Conference on Computer Vision, 2013, pp. 2344-2351, doi: 10.1109/ICCV.2013.291.