Exploring The Precision and Practicality of 6DOF Pose Estimation Approaches Using RGB(D) Data Through Deep Learning-Based and Hybrid Approaches

Amin Heyrani Nobari
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, USA
ahnobari@mit.edu

Abstract*—Estimating the pose of objects of interest in 3D space is fundamental for robotics manipulation and enables robots to perceive and interact with the world around them and the objects in their surroundings. With the advent of depth cameras and their wide availability, this goal has come closer to being realized than ever before, and pose estimation using RGBD images has become the main approach for pose estimation. This problem however is still a rather difficult one in its most general form given the fact a large variety of scenes and objects have to be perceived and their pose estimated while the scene may be noisy and cluttered with obstacles occluding objects in a scene. In this work, we explore a few possible approaches to handle the aforementioned problem and develop a couple of methods for handling this task for two separate datasets. We view the problem as having two primary sub-problems. The first is identifying objects of interest in a scene the second is estimating the pose of each instance of objects of interest. As such we use Mask R-CNN for the first step and for the second step we explore conventional approaches such as the RANSAC algorithm, Fast Global Registration (FGR), and Super4PCS, while also developing a full deep learning approach as well. We conduct experiments to demonstrate the capabilities of each approach as well as compare their performance to the state-of-the-art models in current literature. We observe that our purposed deep-learning approach is able to perform on par with current state-of-the-art models and surpass most of them on the LINEMOD-Occlusion dataset. We also identify key challenges that the LINEMOD-Occlusion dataset fails to encompass and generate a new synthetic dataset which proves a much more difficult challenge to all learning-based approaches, proving the validity of the motivating hypothesis for creating a new dataset. Finally, we see that conventional approaches fail to predict poses with reasonable accuracy with the exception of Super4PCS which performs better (although not at all on par with learning-based methods), while also being painfully slow in comparison to GPU-accelerated deep learning models. Given, this we believe deep learning-based approaches such as the one we propose are the most promising approaches.

I. Introduction

6D object pose estimation using RGB or RGBD images is fundamental to many different areas of research and real-world engineering applications. These applications range from autonomous driving and navigation [1]–[3] to augmented reality [4], [5] and most notably robotics manipulation [6]. This makes the need for robust algorithms and methods for this task crucial and of significant importance. However, this is in no way a trivial task, the problem poses great challenges which stand in the way of developing robust and generalizable approaches for this purpose. One of these challenges is the fact that a variety of objects may be of interest and they will likely have fundamentally different geometry and shape as well as color and texture which makes it difficult to handle a large variety of objects simultaneously. Another important issue is that in the real world, objects are likely to be occluded by other objects and obstacles which will make it harder to rely on a set of predetermined features to make pose estimations as such features may not be visible at all times.

A large portion of methods in the past has focused on using only RGB color information to make pose estimations. This kind of approach is conventionally done using feature detection methods whereby an algorithm identifies key features of an object within an image and uses this information as well as the camera intrinsic parameters to make a prediction on the pose of the object [7]–[9]. Although these approaches are brilliant conventional computer vision algorithms they are highly sensitive to occlusion as well as being highly reliant on notable features within objects of interest, which makes them limited to objects which have distinct recognizable features and incapable of handling smooth and uniformly textured objects. Regardless, there is no longer a significant emphasis on using only RGB information as depth cameras have become widely available and affordable in recent years, and given their popularity in many robotics applications it is more sensible to focus our efforts on performing 6D pose estimation using RGBD information. This allowed for further developments of feature-based object detection and pose estimation approaches which are no longer limited by textures and visual features and can perform pose estimation using 3D features captured by the depth sensor of cameras [10]–[13]. However, still, these methods are susceptible to occlusion as the key features of interest may not be visible

at all times, as well as variations in lighting and shadows. Finally, when it comes to conventional approaches using depth information it is noteworthy to mention a lot of times local and global optimization-based approaches can be used to obtain pose estimations. This can be done by first identifying an object of interest within an RGBD image then extracting only the depth information relating to the pixels which include the object as point clouds and using global registration methods such as RANSAC-based approaches [14], or the fast global registration algorithm proposed by Zhou et al. [15], or the Super4PCS algorithm [16] to estimate the pose which can also further be improved through local registration approaches such as iterative closest point (ICP) algorithm [17].

Given the limitations of conventional approaches and the advent of learning-based methods in the past decades, many researchers have focused their attention on developing robust 6D pose estimation models using data-driven machine learning approaches and have shown that these methods provide a more robust and generalizable solution to the problem in comparison to most conventional methods that existed prior to the development of such models [18]–[24]. One important thing to note here is that many of the best-performing models see significant improvements in their performance when combined with local registration algorithms such as the aforementioned ICP [18], [19]. Although a valid approach for pose estimation using ICP or other conventional iterative methods can be significantly slower compared to using only learning-based methods which can often run on GPUs very efficiently.

In this work, we explore two different avenues for 6D pose estimation and introduce a new framework for 6D pose estimation using deep learning approaches and a combined deep learning and conventional hybrid approach and compare the performance of our model with some of the state-of-the-art approaches in the literature. We show that our approaches perform on par with the current state-of-art models for pose prediction both in accuracy and speed.

Beyond the main contribution, we identify an important concern when it comes to these learning-based approaches which is that all of the methods developed are often tested and trained on a limited number of specific datasets, precisely the LINEMOD and YCB-video datasets, which means that these methods are often not stress tested against any different kind of datasets which could present a different challenge which may be uniquely hard to deal with. Given this important issue, in this work, we introduce a new dataset that is synthetically generated and introduces some different challenges.

II. Background and Related Works

In this section, we will briefly discuss the problem of 6D pose estimation and some of the related works which serve as inspiration for our approach. First, we will briefly go over the problem itself and how it can be formulated, then we will discuss relevant methods for global registration, and finally, talk about some of the learning-based approaches which we take inspiration from for our model. Finally, we will briefly mention the common datasets used to benchmark and train pose estimation models and frame our motivation for developing a new dataset.

A. The Problem: 6D Pose Estimation Using RGBD Images

For this project, the primary goal is to predict the pose of objects of interest in 3D using RGBD images. This means that we aim to develop a model that takes as input an image containing RGB color information and depth images contain depth information, which can be represented as point clouds by converting the depth values to positions in 3D using the camera intrinsic parameters and predict the pose of all objects of interest in the said image relative to the camera. To represent the pose of any object in 3D we must know the object's translation in space relative to the camera frame which in 3D would be the position of the object centroid across the x,y, and z axes of the camera frame and the rotation of an object along the same three axes, totaling 6 continuous parameters, hence the 6D pose, for each object.

However, the problem isn't simply looking at an image of an object and determining its pose relative to the camera. The problem we aim to address is a more generalized one that involves an image with an arbitrary number of objects in it some of which are objects of interest that must be identified and their pose must be estimated. This means that the problem we aim to solve in this project involves fundamentally two aspects. The first is to identify instances of objects of interest within images, and the second is to predict the pose of all objects of interest in the image relative to the camera (Fig.1). Given this realization for our method, we split the problem into two parts and develop our approach around addressing each of the problems in separate stages.

B. Mask R-CNN: Instance Segmentation

As discussed prior one of the main aspects of the task at hand is to identify objects of interest and the pixels within the image that are associated with each instance of an object. This is a problem typically referred to as the instance segmentation problem. Several models have been developed for this purpose using deep convolutional neural networks (CNN) [25]–[32]. Despite these various different architectures developed by researchers in the past few years a specific kind of approach called Mask R-CNN [32] has been the most successful and later models are often based around improving this architecture or introducing new features to this architecture [33], [34]. Given the popularity and success of Mask R-CNN, we adopt this architecture for our methods and use this architecture for the first stage of our approach in both fully learning-based approaches and hybrid approaches. Here we go over this approach briefly, however, for a more detailed discussion of the model interested readers are referred to [32].

The Mask R-CNN model is built around a variant of an instance classification model (i.e., a model that only classifies objects within an image and proposes a bounding box for each instance of an object) called faster R-CNN [35], which itself is a variant of a model called fast R-CNN [36]. The faster R-CNN model starts with a CNN backbone, often a backbone

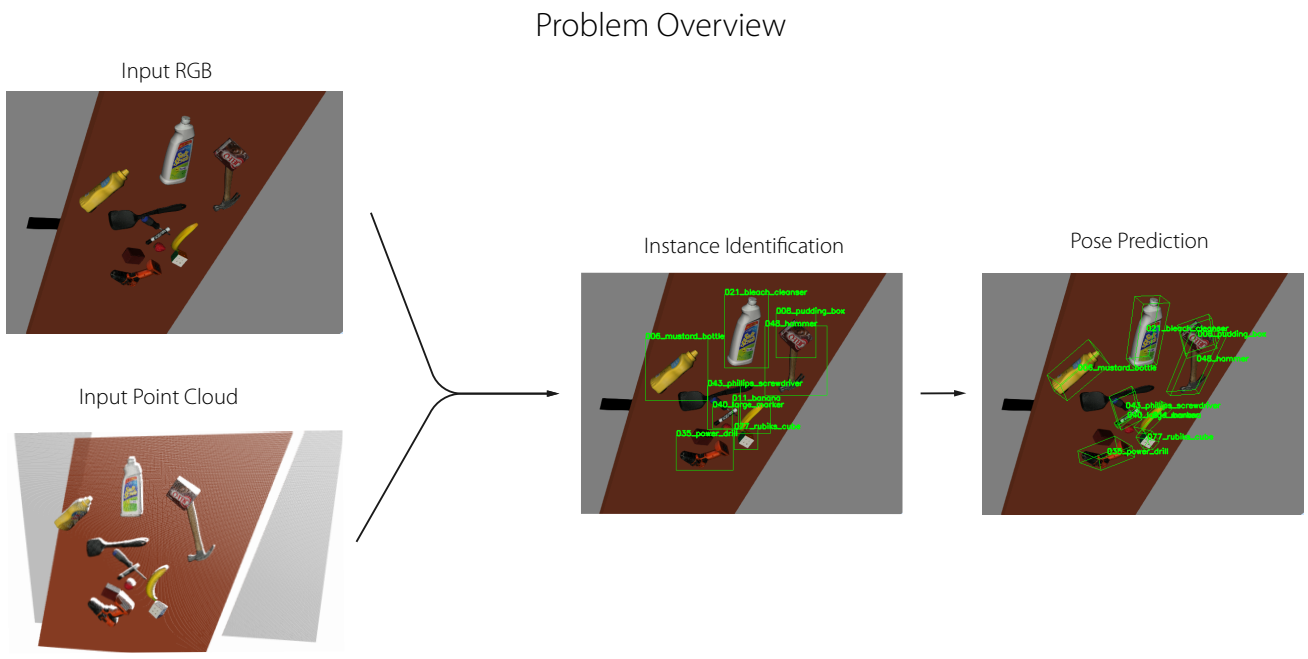

Fig. 1. Overview of the problem of 6D pose generation using RGB-D images.

based around the Res-Net [37] which generates feature maps across an image. Then the faster R-CNN model deploys a model called regional proposal network (RPN) which goes over the features maps from the backbone to identify regions of interest or anchor boxes for each region of interest. This network essentially learns to identify if a given region is background or an object, while simultaneously learning to fit anchor boxes to the objects. After this, each bounding box is classified separately by performing region of interest pooling (RoI Pooling) on each bounding box and classifying each region to identify what type of object it is. Mask R-CNN basically takes this idea one step further by not just classifying the objects within each region of interest but also predicting the binary segmentation mask which for each region identifies exactly which pixels include each given object. In our work, we use Mask R-CNN with a ResNet 50 [37] backbone as described in the original work [32]. It is important to note, however, that this is by no means the optimal approach to take as more efficient and accurate models with lower computational cost have been since developed which would be a more ideal solution for this problem, a notable example is Efficient Det [38]. However, given the limited time for the project and the familiarity of the author with Mask R-CNN, this model was chosen as the instance segmentation model for the project as the smaller variant of this model was found to be fairly fast and practical for this application as most of the datasets for 6D pose estimation do not pose a particularly difficult challenge when it comes to instance segmentation.

C. 6D Pose Estimation Using Data-Driven Approaches

Learning-based approaches have been widely applied to the pose estimation problem. Here we will briefly go over a rough timeline of different architectures and the primary principles behind the success of these models and take these principles and apply them to our method with our own implementation.

One of the first successful models that applied data-driven approaches to 6D pose estimation was PoseCNN [18]. PoseCNN uses an architecture mostly involving CNNs which performs the task of segmentation and pose prediction simultaneously. In PoseCNN separate parts of the model predict rotation and translation. One of the noteworthy parts of this model which increases its capabilities compared to models prior to PoseCNN is the fact that PoseCNN does not predict the translation part of the pose as a direct 3D continuous vector, rather PoseCNN predicts the center of the mode (i.e., the translation relative to the camera) by identifying which pixel in the image corresponds to the center of a given object and what depth the center has from the camera. Given these predictions, PoseCNN can deduce the translation using the camera's intrinsic parameters using the pinhole camera model [18]. This is of particular importance when only RGB information is utilized for predictions and depth information is not used in point cloud form. This is because without having depth information it is not an easy task for CNNs to understand and deduce the camera parameters on their own and having that information be directly implemented in the model improves their accuracy significantly. Furthermore, in PoseCNN the authors also input depth information to their

model as a 4 channel in the input image, which improves their results, however, it is important to note that without camera parameters CNNs will have a harder time interpreting these results in a meaningful way as it was evident in the author's concern with translation prediction. The primary takeaway from this model is that if depth information is not being processed as point clouds translation prediction is best done through center pixel and depth prediction rather than a direct regression. This principle has been adopted by most models developed for pose prediction using images (focused mostly on RGB) since PoseCNN and is often the preferred method of translation prediction for a lot of models that have much stronger results compared to PoseCNN and have come a long way from that point [21]. Another important point that is noted by the authors of PoseCNN is that the predictions made by PoseCNN although much better than conventional approaches still lack very high precision. Given this, the authors purpose an additional step of ICP on top of the predictions of PoseCNN (which is done using the point cloud representation of the depth image), which shows results in significant improvement in the results [18]. ICP or other iterative conventional local registration algorithms can be very slow in comparison to GPU-accelerated deep networks, which means that this final refinement step comes with a large computational cost.

Seeing these limitations of PoseCNN and other CNN-based approaches another important work in the literature that introduced some new concepts and principles to the field was the work called DenseFusion [20]. In this work, the authors introduce some novel approaches to their model which are noteworthy. The first is that they acknowledge that the depth information is being utilized poorly if the depth image is processed by CNNs and they suggest using the depth image in point cloud form which allows them to establish a new state of the art at the time the paper was published [20], while also allowing them to directly predict the translation without resorting to center pixel and depth prediction as we saw in PoseCNN. They do this by using the PointNet architecture [39] to process the point clouds. Another important fact that is acknowledged by the authors is the fact that most of the prior methods see significant improvement when applying ICP or other computationally expensive post-processing algorithms. However, these post-processing steps add significant computational cost to the overall approach and slow the overall process. Given this, the authors aim to replace this step with a learning-based iterative post-processing approach which comes at a much lower time cost and gives the authors up to 10% improvement without resorting to ICP [20]. This kind of iterative learning-based model has since become a staple in pose prediction models and is being applied in most models being published these days [21], [40]. In fact, some works like the RNNPose [40] take this approach and apply it to the maximum extent and design their entire model around the concept of iterative refinement.

From our review of the literature, we identify a few key principles which we adopt in our work. The first is that depth information includes rich context which is best utilized in point cloud form, therefore we build our model around working with the point cloud representation of depth information while also taking into account the visual image information. The other key takeaway is that some form of post-processing is highly beneficial in most cases, given this, we also develop an iterative refinement approach to post-process our model's prediction for added accuracy.

D. Pose Estimation Datasets

Fig. 2. 9 randomly selected images from the LINEMOD-occlusion dataset show the consistency in how objects are positioned in the dataset and the fact that the camera orientation and position do not vary widely making the dataset less diverse.

The last topic of discussion relevant to our work is the common datasets that are used for training and testing the pose prediction models. There are several different datasets that are often used to train and test models that are developed for pose estimation, however going over all of the existing datasets in this section would be outside the scope of this work, therefore, we will discuss the two most popular and tested datasets which are often used to establish the state of the art. These datasets are the YCB-Video dataset introduced in the PoseCNN paper [18] and the LINEMOD [10] and LINEMOD-Occlusion datasets [41]. Of these datasets, most models perform exceptionally well on the YCB-Video dataset with close to perfect accuracy [18], [20], [38], [40]. On the other hand, the LINEMOD-Occlusion dataset has presented a major challenge for most models with performance accuracy in the 60% range for even the best models [40]. Given this, we choose the LINEMOD-Occlusion dataset for our testing as well. However, a few important details must be mentioned about these datasets. Despite presenting a challenging problem, the LINEMOD-Occlusion dataset fails to encompass a more generalized problem, which in the opinion of the author should be the ultimate goal of pose prediction models. We identify a few key limitations we observe in the

datasets currently used to benchmark pose estimation models and develop a new dataset that we synthetically produce to address these limitations. The first thing we note in these datasets is that objects are always placed within a consistent set with respect to the camera (See Fig.2), for example, in these datasets objects are always within a relatively consistent distance to the camera and the camera is always looking at the scene from a certain side and we do not see images from behind the same scene or from the sides, furthermore, in these images, the camera is always looking at the scene in an upright orientation (that is images are always oriented straight and not at an angle). The second limitation that is evident is that in these datasets only one instance of every object is present in each image, however, a more general problem would be to have multiple instances of the same object in the same scene to make the task more generalizable. Moreover, in the LINEMOD-occlusion dataset, we only have 8 total objects which need to be classified, which is not a very large array of possible objects, this means that it is hard to say how these models would generalize in a more difficult and generalized case where a large number of objects need to be handled. Finally, we see that objects in these datasets are always oriented in a handful of specific orientations (See Fig.2), for example, the driller in the LINEMOD dataset is always oriented upright on the table with its battery side on the table, this makes the pose somewhat consistent at least across one axis of rotation which creates a bias in the trained models, which is not very good for the generalizability of these models. Given these limitations, we introduce a synthetic dataset that specifically addresses these challenges and introduces a more difficult problem for these models and a more generalized challenge in the hopes of establishing the next step towards more general models for pose estimation.

III. Introducing A New Challenge: Creating A New Dataset

As mentioned in the prior section, we create a new dataset to specifically introduce a few new challenges for pose prediction models. The new challenging aspects of our dataset compared to existing ones are as follows (See Fig.3):

- The camera position and orientation change significantly in each image and the camera not consistently pointing at the objects from one direction rather the camera points at the objects from random directions.
- Objects are positioned completely randomly and are not consistently positioned across any axes of rotation.
- Rather than having only up to one instance of each object in the frames each frame may include an arbitrary number of instances of each object.
- Rather than having objects always positioned within a roughly consistent distance from the camera objects are sometimes placed in unusual locations to introduce more stochasticity and increase the difficulty of the task at hand.

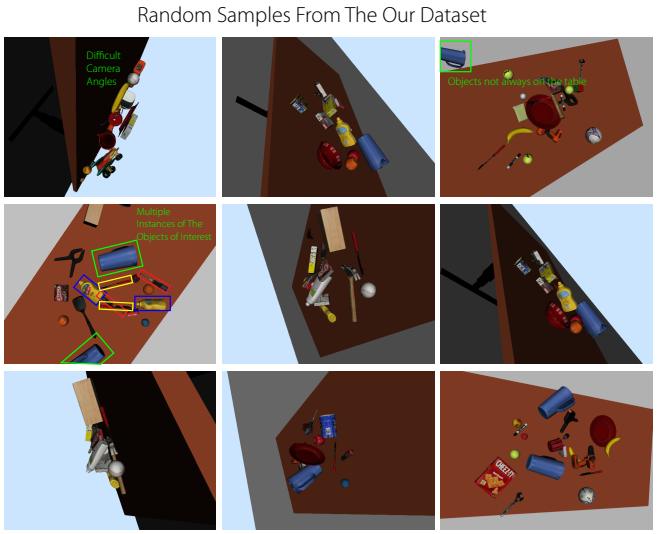

Fig. 3. 9 randomly selected images from our dataset demonstrate some of the features of the new dataset.

- To further increase the difficulty of the challenge we include 18 objects of interest in our dataset (compared to the 8 object types in the LINEMOD-occlusion dataset).

By doing this we create a much more challenging and generalized task of pose prediction. We demonstrate the increased difficulty of this dataset in later sections by comparing the performance of some models in the LINEMOD-Occlusion dataset as well as our dataset.

A. Dataset Generation Pipeline

B. The Problem: 6D Pose Estimation Using RGBD Images

Now that we have established our overall goals with the dataset, we will discuss how we synthesized our dataset and in later sections, we will discuss the details of the dataset and the statistics of the dataset. An overview of the approach we take is demonstrated in Fig.4. To generate a new pose prediction dataset we use a subset of the YCB models [42] as target objects and the rest of the YCB-models as noise objects in our dataset. Specifically, we use the following models as target objects:

- 055_baseball
- 056_tennis_ball*
- 072-a_toy_airplane
- 019_pitcher_base
- 040_large_marker
- 021_bleach_cleanser
- 077_rubiks_cube
- 048_hammer
- 008_pudding_box
- 053_mini_soccer_ball*
- 011_banana
- 006_mustard_bottle
- 013_apple*
- 029_plate*

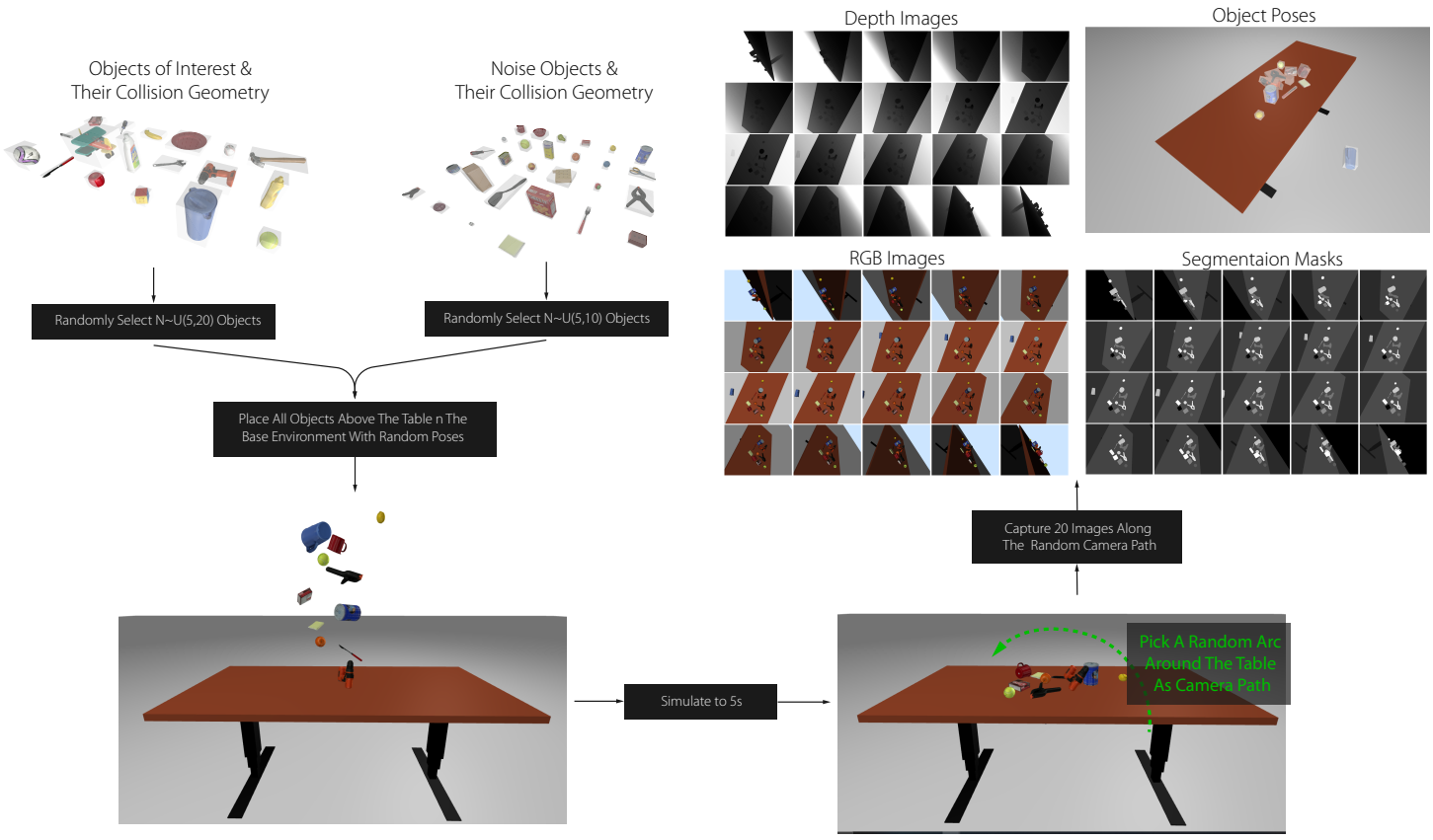

Fig. 4. Overview of the dataset generation approach employed in this work.

- 035_power_drill
- 043_phillips_screwdriver
- 032_knife
- 042_adjustable_wrench

The items with * are considered symmetric objects.

To generate our dataset using these models we use DRAKE [43] for physics simulation and capturing labeled RGBD images. To do this we set up an environment with a table in it and place random objects (target objects and noise objects with repeat objects also being possible) above the table and sometimes intentionally outside the boundary of the table such that they would fall on the ground. Then we simulate the physics of the objects falling on the table for 5 seconds. It is important to note that for the simulation we simplify the collision geometry of the objects by simply modeling their collision with bounding boxes to reduce the computational cost of the simulations and make it practical to generate scenes with large numbers of objects in a reasonable time (see Fig.4). This, however, does not cause any issues down the line as even if the physics of a scene are not realistic pose estimation should be possible and not affected by the realism of the scene. Then to capture images we move an RGBD camera simulated in drake in a random arc (random start and end point across the table) around the table and capture 20 RGBD images of every scene. Then using the tools in drake we capture the relevant binary masks for each object of interest for every one of the 20 images. We also record the ground truth pose of the objects relative to the camera pose which will be used for training the models (see Fig.4).

Finally, we generate colored reference point clouds of the target objects from their meshes using MeshLab. This is necessary for the training of most pose prediction models as they use these reference point clouds to calculate their loss, as do we in our model (This is discussed in later sections).

C. Dataset Content and Statistics

Our dataset is organized into separate folders:

- **RGB**: RGB images captured by the simulated camera in DRAKE in png format with a resolution of 640x480 pixels. File names indicate the image number starting from 0 (#.png).
- **depth**: Depth images (640x480 pixels) in meters captured by the simulated depth camera in drake. These images are saved as NumPy arrays (.npy format) so as to not round values into integers for saving in png format. File names correspond to the RGB images (#.npy).
- **masks**: Inside this folder, there exists a folder for each of the 18 models. For each of these models, the relevant binary masks are provided. File names include the corresponding RGB image number and the instance number to

allow for saving separate masks for each instance of the object in the same RGB image (file names in the format #_#.png).

- **merged_masks**: A single grayscale png image for every RGB image which includes all masks. The values of pixels are coded to allow for distinction between different objects and their instances. Specifically, the following values are set for each object and instance (Note that there is no case where more than 4 instances of the same object are present):
  - 055_baseball: 14 - #instance (starting from 0)
  - 056_tennis_ball*: 28 - #instance
  - 072-a_toy_airplane: 42 - #instance
  - 019_pitcher_base: 56 - #instance
  - 040_large_marker: 70 - #instance
  - 021_bleach_cleanser: 85 - #instance
  - 077_rubiks_cube: 99 - #instance
  - 048_hammer: 113 - #instance
  - 008_pudding_box: 127 - #instance
  - 053_mini_soccer_ball*: 141 - #instance
  - 011_banana: 155 - #instance
  - 006_mustard_bottle: 170 - #instance
  - 013_apple*: 184 - #instance
  - 029_plate*: 198 - #instance
  - 035_power_drill: 212 - #instance
  - 043_phillips_screwdriver: 226 - #instance
  - 032_knife: 240 - #instance
  - 042_adjustable_wrench: 255 - #instance
- **models**: This folder includes reference ply point cloud files for each of the objects. This folder also includes a yml file called models_info.yml which includes the diameter of each model as well as the bounding box information for each model (indicated by the minimum values of the point cloud in x, y, and z as well as the size of the bounding box in each of the x, y, and z directions).
- **gt.yml**: a yml file including the ground truth pose of each instance of the objects. This file is organized with an entry for each file in the RGB folder which includes the rotation matrix and translation vector for each instance of the objects within the corresponding RGB image as well as the camera intrinsic matrix for each image (this value is repeated for every image even though it does not change).

In total 25,000 images are produced with 5 to 20 objects of interest and 5 to 10 random noise objects with over 10,000 instances of each object of interest. This, however, is not the maximum number of points that can be generated and this amount was generated in under a day using an Intel i9-12900k processor. This leaves room for significantly more data to be produced for the training of very large models if that is necessary. The code and data are made publicly available for anyone to access and generate a dataset. The code can be found at https://github.com/ahnobari/6.4212-6D-Pose-Prediction.

IV. Proposed Methods

In our implementation, we see the problem as having two stages and this overall view of the problem and our overall approach to the problem is demonstrated in Fig.5. As we discussed in the problem description the task at hand involves first identifying and classifying each instance of the objects of interest, then, using that information to estimate the pose of each object. Therefore, we approach the problem in two separate parts. The first is instance segmentation and the second is pose estimation and we approach these two separately (See Fig.5) by first training an instance segmentation model on both our dataset as well as the LINEMOD-Occlusion dataset. Then we look at two different approaches for pose estimation. The first is using conventional global registration methods to calculate the pose of different objects, what we call the hybrid methods, and the second is developing a learning-based pose estimation model for predicting the pose of each identified object. We will discuss the details of our approach for each of these approaches in the following sections.

A. Instance Segmentation

As mentioned prior for the instance segmentation part of our approach we use the Mask R-CNN model with a ResNet-50 backbone as described in the original paper [32]. When training we use Image-Net pre-trained weights for the ResNet backbone but do not freeze any layers or parameters for training. We then train the model for 100 epochs using the Adam optimizer with a learning rate of 10^{-3} which decays by a factor of 10 every 30 epochs. For training of the LINEMOD-Occlusion dataset, we use the training and test split used in PoseCNN [18], as this is common practice for most works and allows for easier comparison of metrics later on. For our own dataset, we use a random 90%-10% split for training and testing respectively.

B. Hybrid Approach

Once instance segmentation is taken care of we use the masks generated by the segmentation model to extract the pixels in the image that correspond to a specific object. Then we take those pixels from the depth images and create a masked point cloud that in theory would include only points that relate to the target object (See Fig.5). After this, we use conventional global registration algorithms on the scene point cloud and the reference point clouds provided in the dataset (See Fig.5). For this hybrid approach, we explore several different algorithms and report the results for all of the algorithms tested. The first algorithm we use is the RANSAC-based [44] global registration. For this approach, we use the entire point cloud from the image and a voxel downsampled point cloud from the reference point clouds (with a voxel size of 2mm). Then we extract the Fast Point Feature Histograms (FPFH) [45] from both point clouds and use the RANSAC algorithm with these extracted features to estimate the pose of the object. The second algorithm we use is the fast global registration algorithm (FGR) [46]. Similar to the RANSAC approach for this algorithm we only downsample the reference point cloud

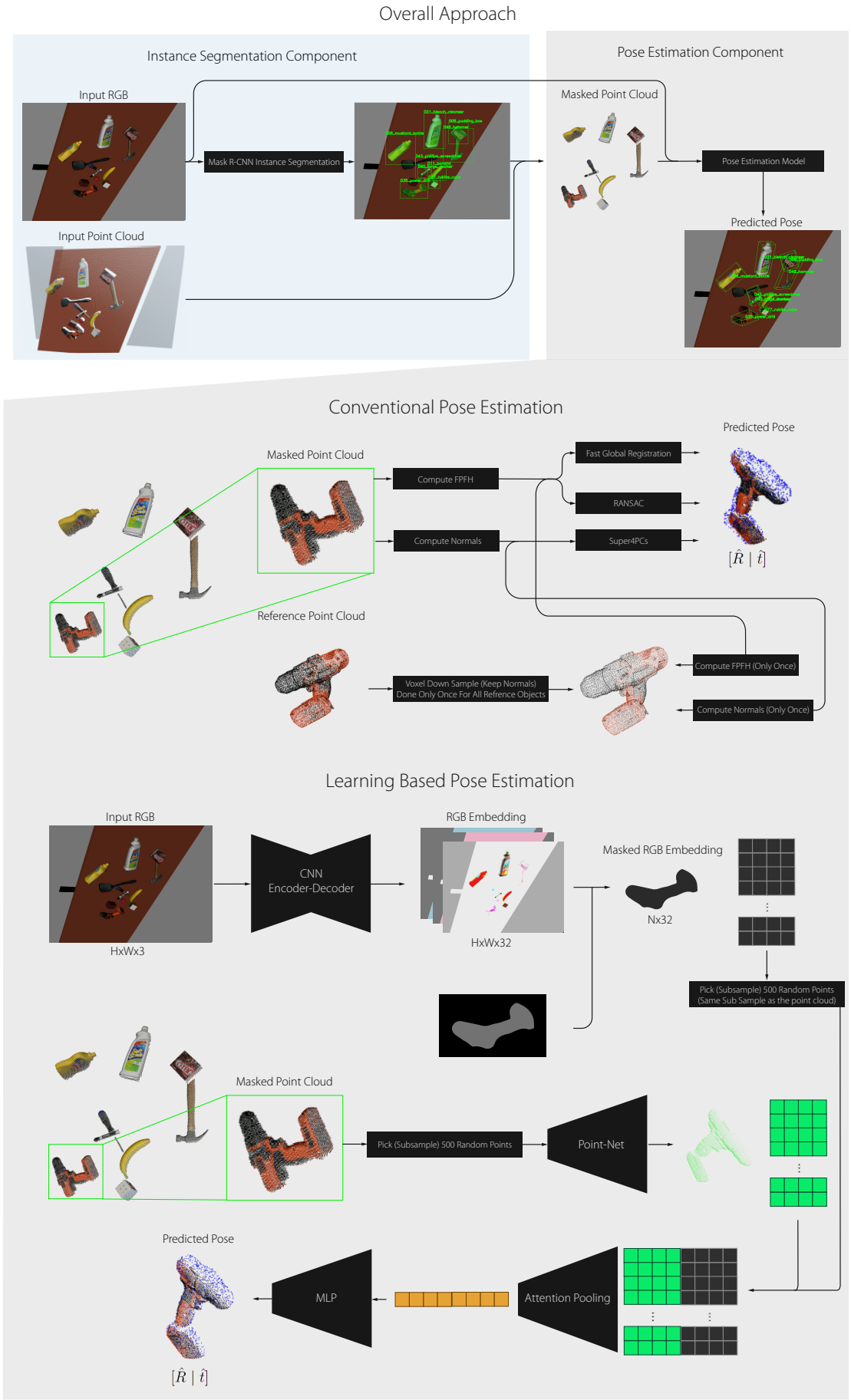

Fig. 5. Overview of our proposed approach and detailed diagram of the architecture of both conventional approaches and our learning-based approach.

and use the full masked point cloud from the image. We also use the same FPFH features in this algorithm as well. Our implementation for both RANSAC and FGR is done using the implementations provided by the Open3D library [47]. Finally, we also explore the 4-Points Congruent Sets (4PCs) algorithm [48] and the faster variant Super4PCS [16] algorithm for global registration. We found that the Super4PCS and 4PCs algorithms work best when rather than just point clouds they have access to the point cloud normals. For the reference data, this is already available on the models provided by each dataset, however, for the scene point clouds, we have to calculate the normals before sending them through the algorithms. For normals calculation, we use plane fitting on the 10 nearest neighbors for each of the points in the point cloud, and to remove ambiguity on the normal direction we always keep the z direction of the normals negative (i.e., facing the camera) as this is the correct way to look at the normals on a given point cloud. Then we pass these point clouds to the Super4PCS algorithm and set the overlap parameter of the algorithm to 0.7 and the registration accuracy (or delta) is set to 1cm and the number of samples used is set to 200. Using these settings for each algorithm we report the results of pose estimation using these methods in the results section.

C. Full Learning-Based Approach

The other approach for pose estimation that we investigate is the approach of learning to estimate the pose of each object using a data-driven approach. For this purpose, we use a mixture of CNN and Point-Net in our architecture (See Fig.5), similar to the ideas brought forth by the DenseFusion model [20].

1) Overall Approach: Since we already have an instance segmentation model we can use the masks predicted by this model to extract the pixels from the depth image that are predicted to be part of the target object. That leads to a point cloud of the scene, which can be processed by a Point-Net-like model. Similarly, we can extract the features from the visual signal from the image using the same mask. In our approach, we follow a similar overall approach to the DenseFusion [20] approach, where we extract features from both the point cloud as well as the image and fuse the features to finally obtain a pose prediction for every instance of the objects. In this approach, we have a few main components. The first is a CNN that processes the RGB information and creates feature maps for each pixel in the image. The second component of the model is a PointNet-based [39] model that processes the masked 3D point cloud from the depth image to generate depth embeddings for each pixel. Finally, the third component is the fusion and prediction of the pose of an object which is done using an attention mechanism.

2) Feature Extraction: As mentioned before, it has been shown that treating depth information as images and applying CNNs to the depth data is sub-optimal. Given this, we perform feature extraction for color and depth information separately.

PointNet [39] was introduced to handle point cloud data and it achieves this through the use of symmetric functions (such as max-pooling) to achieve permutation invariance in processing unordered point sets. Point-Net takes an unordered point set and learns features that are shown to be useful in shape classification and segmentation [39] as well as pose prediction [20]. In our approach, we use this architecture to encode per-point features from each point in the point cloud.

Now that per-pixel features have been extracted from the point cloud we must extract features from the RGB image as well. The idea behind this is to extract per-pixel features from the image such that we can correlate them to the extracted features from the point cloud. to do this, an image embedding CNN is used. In our approach, we input the RGB image to an encoder-decoder CNN which will yield an output image of the same size but with extracted features in the output channels rather than the 3 RGB channels.

3) Feature Fusion: At this point, we have extracted features from both the image and the point cloud. Now we must combine them such that we can make precise pose predictions. The naive approach would be to combine the features into a global feature. This, however, has been shown to be problematic when there is significant occlusion and mask prediction error in the instance segmentation step as the set of features from the feature extraction step may involve pixels that are not actually associated with the target object [20]. Similar to DenseFusion [20] we also fuse the features of every pixel from the CNN and point-net by concatenating them and combining a global representation with them (see Fig.). However, unlike DenseFusion rather than making per-pixel predictions with associated confidence predictions we make pose predictions through the use of self-attention pooling. To do this we train a neural network to predict the attention that should be paid to each pixel's features and apply that as a weighted average to the input set of fused features. In this way, we mitigate the need for per-pixel predictions as the attention mechanism should take care of combining the information in a useful manner. However, an important deviation we take from the typical implementation of attention pooling is that we remove the normalization layers that are typically applied in most implementations. We find that normalizing the depth information makes it very difficult for the model to learn translation values and renders the entire model ineffective, therefore, we remove all normalization layers in our model entirely. One more important thing to note is that during training a subset of 500 pixels is randomly picked from the full predicted mask of a given object (In the case in which fewer than 500 pixels are identified as belonging to an object all the points are used) and these pixels are used for making the final prediction, therefore the total input length for the model never exceeds 500.

4) Pose Prediction: Having discussed the overall approach, we must now describe the learning objective which enables our model to perform its predictions effectively. For this purpose, we use the commonly employed pose estimation loss [20] which measures the distance between the points from the reference model transformed to the ground truth pose and corresponding points of the same reference model transformed

by the predicted pose. Precisely, the loss that is minimized during training is as follows:

$$L_p = \frac{1}{M} \sum_j \|(Rx_j + t) - (\hat{R}x_j + \hat{t})\| \quad (1)$$

where x_j is the j^{th} point from the M randomly sampled points of the reference point cloud from the object's reference model, $p = [R \mid t]$ is the ground truth pose, and $\hat{p} = [\hat{R}|\hat{t}]$ is the predicted pose from our model. This loss function is only well-defined for asymmetric objects, where the object has a unique canonical frame. Symmetric objects can have possibly an infinite number of frames, which renders the above loss function useless. Therefore, for symmetric objects, we instead minimize the single-directional chamfer distance from the predicted transformation of the reference points and the ground truth transformation of the reference points:

$$L_p = \frac{1}{M} \sum_j \min_{0<k<M} \left\|(Rx_j + t) - \left(\hat{R}x_k + \hat{t}\right)\right\| \quad (2)$$

5) Iterative Refinement: As we discussed before, it is a common approach to introduce an iterative refinement component to most learning-based models. We also saw that using ICP or other conventional approaches can make things significantly slower and reduce some of the benefits of GPU-accelerated learning-based models. Given this, we adopt the approach proposed by DenseFusion [20] and apply it to our model's predictions. The overview of our refinement approach can be seen in Fig.6.

To refine existing predictions the refinement network has to see prior predictions and make corrections to these predictions. To do this we train a separate Point-Net which takes as input RGB embeddings from the main model and transformed point clouds (See Fig.6). This transformed point cloud is essentially the original masked point cloud from the depth image, however, it is transformed in the opposite way of the predicted pose (See Fig.6). In this way, the transformed point cloud is in the residual relative frame with respect to the original point cloud and the predicted transformation. Given this, we train the refinement model to predict the residual needed to correct the current pose prediction and train it using the same objective as before but this time we add the predicted residuals to the prior prediction for loss calculation.

$$L_p = \frac{1}{M} \sum_j \|(Rx_j + t) - ((\hat{R} + \hat{\Delta R})x_j + \hat{t} + \hat{\Delta t})\| \quad (3)$$

where $\Delta\hat{p} = [\hat{\Delta R} \mid \hat{\Delta t}]$ is the predicted corrections. In our approach we only apply two iterations of refinement, however, more iterations are possible and not nearly as slow as ICP or other conventional approaches. Finally, a note on the training of the refinement network is that as evident this network relies on the RGB embeddings of the main network, and this information at the beginning of the training is not very useful, therefore we train the refinement model separately after the main model has been trained. Furthermore, at that point the pose predictions are all accurate on the training data, therefore to train the model instead of just using the pose predictions from the model we add noise to the predicted pose by the model to make it vary in accuracy and train the refinement model more effectively. We find that this noise specifically makes the refinement model significantly more effective, however, this is not yet provably demonstrated by ablation studies and is based purely on empirical observation.

V. Experimental Results

Now that we have established a few different approaches we run experiments on these approaches on both the LINEMOD-Occlusion dataset as well as our own synthetic dataset. Before we discuss the results however we must establish our metrics for measuring the performance of each approach.

A. Metrics

We measure the performance of each model with respect to accuracy as well as time efficiency. We measure everything on the same computer with an RTX3090Ti GPU and an Intel i9-12900k CPU to make things consistent with respect to time measurements. We report the average frames per second for each of the approaches for time efficiency. It is important to note we include all of the overhead for each method in our time calculations. For example, when applying Super4PCS normals are calculated for each point cloud and that requires time which will be added to the overall time for Super4PCS. Similarly, all overheads are captured for each of the approaches tested. However, the overhead regarding any read/write operations involving the hard drive is isolated and removed from our measurements.

For accuracy, we measure the common ADD(-S) metric for both datasets with respect to each method [21]. This metric is identical to the loss we described in the previous sections:

$$\text{ADD}(-S) = \begin{cases} \text{if asymmetric :} \\ \frac{1}{M}\sum_j \|(Rx_j + t) - (\hat{R}x_j + \hat{t})\| \\ \\ \text{if symmetric:} \\ \frac{1}{M}\sum_j \min_{0<k<M} \left\|(Rx_j + t) - \left(\hat{R}x_k + \hat{t}\right)\right\| \end{cases} \quad (4)$$

however, rather than just reporting the value, accuracy is reported. An estimated 6D pose is considered correct if the ADD(-S) is smaller than 10% of the object's diameter.

B. Implementation Details

Going over every minute detail of the implementation in this section would be impractical and not productive, therefore, for the sake of brevity in this report here we refer the readers to our publicly available code which can be found at https://github.com/ahnobari/6.4212-6D-Pose-Prediction.

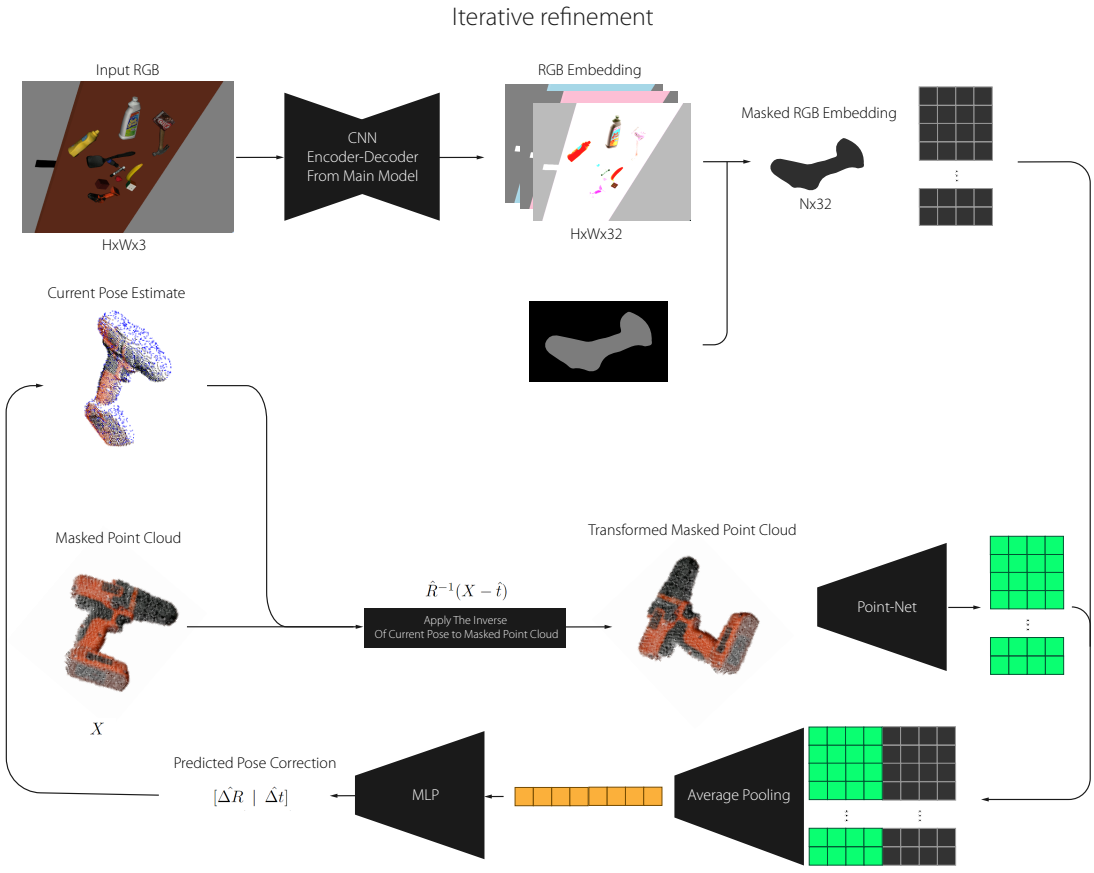

Fig. 6. Details of the refinement component of our model.

C. LINEMOD-Occlusion Results

For the LINEMOD-Occlusion we report results on the validation data and we use the split that was used in PoseCNN [18]. Although it would be best practice to run multiple training runs and report average performance the time available for this project was limited therefore, we only have run the training once, which means that the results are not fully verified at this point, however, in changing the architecture and exploring different approaches results were similar throughout different variation of the current architecture. We also trained models from some of the recently published works on both datasets and report the results of those experiments here as well. Fig.7 qualitatively demonstrates the accuracy of our full learning-based model. It is clear that the model struggles most with symmetric objects, specifically the model has a tendency of predicting the pose in an upsidedown configuration. This is a common issue with these models as they notoriously struggle most with symmetric objects [18].

The accuracy results of our experiments are displayed in Table III. As evident FGR and RANSAC approaches struggle to perform pose estimation very well. This is likely because the depth information is very noisy and the masks themselves are not perfect. This demonstrates the need for learning-based approaches for this task as even when paired with learning

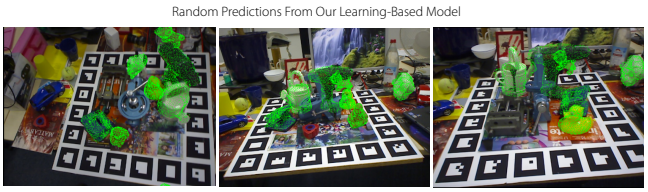

Fig. 7. Sample predictions of our model with refinement. These images qualitatively show the accuracy of the predicted poses.

the limitations of conventional approaches are rather serious. Having said that the Super4PCS approach when paired with a learning-based instance segmentation model performs well and in fact outperforms PoseCNN when only RGB information is used or no ICP is applied.

When it comes to data-driven approaches we see that our approach without the need for ICP significantly out-performs most state-of-the-art methods, and despite not out-performing all methods mentioned here the model performs on par with them when iterative refinement is applied, and with more time and refinement of the architecture, it is clear that this kind of approach can outperform or match other state-of-the-art methods.

We also explore the practicality of each of the approaches by measuring the average frame rate that these models can

TABLE I
EXPERIMENTAL RESULTS ON THE ADD(-S) ACCURACY OF EACH APPROACH ON THE LINEMOD-OCCLUSION DATASET. FOR EFFICIENTNET WE TRAIN THE MODEL FROM SCRATCH USING THE CODE PROVIDED BY THE AUTHORS AND REPORT THE RESULTS OF OUR TRAINING RUN (IN PARENTHESES) AS WELL AS THE AUTHOR'S ORIGINAL EXPERIMENTAL RESULTS.

Model	ADD(-S)
PoseCNN [18]	24.9
PoseCNN+ICP [18]	78.0
PVNet [49]	40.8
HybridPose [24]	47.5
EfficientPose ($\phi = 0$) [21]	79.04 (76.14)
EfficientPose ($\phi = 3$) [21]	83.98 (84.05)
RNNPose [40]	60.65
Ours Hybrid(RANSAC + ICP)	9.38
Ours Hybrid(FGR + ICP)	7.21
Ours Hybrid(Super4PCS)	40.33
Ours Full Learning (No Refinement)	53.02
Ours Full Learning (Iterative)	80.67

TABLE II
EXPERIMENTAL RESULTS ON THE AVERAGE FPS OF EACH APPROACH ON THE LINEMOD-OCCLUSION DATASET USING AN INTEL I9-12900K AND RTX 3090TI

Model	Average FPS
EfficientPose ($\phi = 0$) [21]	29.06
EfficientPose ($\phi = 3$) [21]	15.57
Ours Hybrid(RANSAC + ICP)	0.88
Ours Hybrid(FGR)	0.97
Ours Hybrid(Super4PCS)	0.08
Ours Full Learning (No Refinement)	16.23
Ours Full Learning	14.97

handle, which we present in Table II. These timing values are measured measure such that they include all the overhead for each method. However, it is important to note that since the compiled version of the Super4PCS algorithm was used we could not isolate the hard drive read/write timing of the compiled binaries and could only eliminate this overhead from our side of the experiments, therefore, the FPS values reported for the Super4PCS is slightly under-reported. Regardless Super4PCS despite being the best-performing hybrid approach is by far the slowest approach among the approaches we tested. Following Super4PCS are of course the other hybrid approaches with RANSAC + ICP having the slowest performance after Super4PCS and FGR + ICP following it. These results demonstrate how real-time pose prediction is realistically currently only possible through the use of GPU-accelerated deep learning models.

Amongst the learning-based approaches, we see that amongst the models we have tested EfficientPose with $\phi = 0$ is the fastest model, however, it is not the most accurate model and EfficientPose with $\phi = 3$ has speeds similar to our approach. We see that our approach is capable of making pose estimations at 14 FPS on an RTX 3090Ti. This speed is high enough for real-time predictions which demonstrates that our model and other learning-based models are clearly the superior choices for pose prediction when possible. The only limitation of learning approaches, however, is the fact that we always do not have the necessary data to train them, and in a lot of circumstances, this is the main barrier to the deployment of learning-based models. We will discuss possible future approaches to address this issue and make such models much more generalizable. However, it is clear that Super4PCS provides a strong conventional path to making pose predictions when such data is not available.

D. New Dataset Results

Given that most other models are not able to deal with multiple instances of objects, we create a subset of the dataset without multiple instances to be able to train other models on our dataset as well, therefore the results presented here do not provide a full picture of the capabilities of our dataset. This subset is exactly as described prior with the only difference being that multiple-instance of each object is not present in any of the samples used for this part of the experiments.

TABLE III
EXPERIMENTAL RESULTS ON THE ADD(-S) ACCURACY OF EACH APPROACH ON OUR DATASET.

Model	ADD(-S)
EfficientPose ($\phi = 0$) [21]	33.68
EfficientPose ($\phi = 3$) [21]	32.85
Ours Hybrid(RANSAC + ICP)	53.55
Ours Hybrid(FGR + ICP)	40.69
Ours Hybrid(Super4PCS)	57.32
Ours Full Learning (No Refinement)	18.58
Ours Full Learning (Iterative)	26.87

Since, prior works have not had access to this data to compare our methods to other state-of-the-art methods we use the best-performing model from the listed models for LINEMOD-Occlusion, namely EfficientPose, as a state-of-the-art model to gauge the baseline. The tareftab:customac shows the accuracy performance of each model on the dataset. Somewhat surprisingly, hybrid approaches perform better than learning approaches on this data. This may seem surprising at first but once we acknowledge the fact that this dataset is generated synthetically with perfect accuracy on the depth images as well as the fact that the exact mesh that is captured in the simulation is used for pose predictions we easily see the flaw in taking these results at face value. In fact, if we constructed this environment in reality and deployed an actual depth camera with the official YCB objects we would likely see these methods fail similarly to how they fail with the LINEMOD dataset. Moving past this observation we see that both our learning-based method as well as EfficientPose perform significantly worse on this new dataset compared to the LINEMOD dataset. This confirms our hypothesis that certain features in the LINEMOD dataset make the task slightly easier for learning-based models. We, therefore, show that by systematically adding more complexity to the dataset, despite having more accurate data that is noise free, we have

created a much more difficult challenge for learning-based models. The hope is that this may point out the potential limitations of these approaches and serve as motivation for developing more generalizable models that can overcome these limitations. Finally, it is important to mention that the results for the average FPS of each model are not reported here as it follows a near identical trend as before with slightly different values resulting from more objects being present in each scene which reduces the FPS of all methods by a small amount.

VI. Conclusion and Future Work

In this project, we explored the efficacy of both conventional algorithms as well as purely learning-based algorithms for 6D object pose detection. We also developed a new dataset that poses a more difficult challenge to learning-based methods, with most learning-based methods performing significantly worse when applied to this dataset compared to the LINEMOD-Occlusion dataset. This evidence demonstrates the validity of our initial hypothesis regarding some of the limitations of the LINEMOD-Occlusion dataset by demonstrably making it harder for learning-based algorithms to cope with the task. However, we do see a limitation in our dataset when it comes to hybrid models with learning combined with conventional approaches. We saw that the lack of noise in the data and the precision of the depth images in the data makes the task of pose estimation using conventional approaches much easier, with conventional approaches outperforming some of the best learning-based approaches, confirming that the synthetic nature of the data leads to very precise measurements which are not realistic. We also established that learning-based methods with GPU acceleration provide the best path for enabling real-time pose estimation, given that conventional approaches can be very inaccurate when dealing with real-world data and approaches such as Super4PCS which achieve relatively decent performance are extremely slow.

Despite the clear advantage of deep learning-based methods, it is important to note that in a general manipulation task data may not be easily available, which would render these models in capable of being utilized. Therefore, future research should focus on developing generalized models which can adapt to different environments and target objects. The ultimate goal in fact should not be to perform best on any given dataset, but rather to generalize to zero-shot and few-shot learning methods, where no data is needed for the model to be capable of making accurate predictions, rather given a reference point cloud or mesh of a target object the models should be able to without retrain be able to identify that object in a scene and estimate its pose. This is essentially a model that can be conditioned on any arbitrary target object and is not limited to a pre-defined set of objects that the model can train for. Therefore, the author believes that the most valuable direction for future work is to adapt these models to such a generalized task which can ultimately be used as an off-the-shelf pose estimation model without the need for training or time-consuming and often expensive data gathering.

References

[1] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[2] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A survey on 3d object detection methods for autonomous driving applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3782–3795, 2019.

[3] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: Deep sensor fusion for 3d bounding box estimation," 2017. [Online]. Available: https://arxiv.org/abs/1711.10871

[4] E. Marchand, H. Uchiyama, and F. Spindler, "Pose estimation for augmented reality: A hands-on survey," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 12, pp. 2633–2651, 2016.

[5] W. Lee, N. Park, and W. Woo, "Depth-assisted real-time 3d object detection for augmented reality," in *ICAT*, vol. 11, no. 2, 2011, pp. 126–132.

[6] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," 2018. [Online]. Available: https://arxiv.org/abs/1809.10790

[7] A. Collet, M. Martinez, and S. S. Srinivasa, "The moped framework: Object recognition and pose estimation for manipulation," *The international journal of robotics research*, vol. 30, no. 10, pp. 1284–1306, 2011.

[8] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce, "3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints," *International Journal of Computer Vision*, vol. 66, no. 3, p. 231–259, 2006.

[9] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 1150–1157 vol.2.

[10] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *2011 International Conference on Computer Vision*, 2011, pp. 858–865.

[11] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-class hough forests for 3d object detection and pose estimation," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 462–477.

[12] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Computer Vision – ACCV 2012*, K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 548–562.

[13] U. Asif, M. Bennamoun, and F. A. Sohel, "Rgb-d object recognition and grasp detection using hierarchical cascaded forests," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 547–564, 2017.

[14] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5556–5565.

[15] Q.-Y. Zhou and V. Koltun, "Color map optimization for 3d reconstruction with consumer depth cameras," *ACM Trans. Graph.*, vol. 33, no. 4, jul 2014. [Online]. Available: https://doi.org/10.1145/2601097.2601134

[16] N. Mellado, D. Aiger, and N. J. Mitra, "Super 4pcs fast global pointcloud registration via smart indexing," *Computer Graphics Forum*, vol. 33, no. 5, pp. 205–215, 2014. [Online]. Available: http://dx.doi.org/10.1111/cgf.12446

[17] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[18] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," 2017. [Online]. Available: https://arxiv.org/abs/1711.00199

[19] C. Li, J. Bai, and G. D. Hager, "A unified framework for multi-view multi-class object pose estimation," in *Proceedings of the european conference on computer vision (eccv)*, 2018, pp. 254–269.

[20] C. Wang, D. Xu, Y. Zhu, R. Martin-Martin, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6d object pose estimation by iterative dense fusion," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE

Computer Society, jun 2019, pp. 3338–3347. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00346
[21] Y. Bukschat and M. Vetter, "Efficientpose: An efficient, accurate and scalable end-to-end 6d multi object pose estimation approach," 2020. [Online]. Available: https://arxiv.org/abs/2011.04307
[22] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "DeepIM: Deep iterative matching for 6d pose estimation," *International Journal of Computer Vision*, vol. 128, no. 3, pp. 657–678, nov 2019. [Online]. Available: https://doi.org/10.1007%2Fs11263-019-01250-9
[23] K. Park, T. Patten, and M. Vincze, "Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.
[24] C. Song, J. Song, and Q. Huang, "Hybridpose: 6d object pose estimation under hybrid representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
[25] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.
[26] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *European conference on computer vision*. Springer, 2014, pp. 297–312.
[27] ——, "Hypercolumns for object segmentation and fine-grained localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 447–456.
[28] J. Dai, K. He, and J. Sun, "Convolutional feature masking for joint object and stuff segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3992–4000.
[29] P. O. O Pinheiro, R. Collobert, and P. Dollár, "Learning to segment object candidates," *Advances in neural information processing systems*, vol. 28, 2015.
[30] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár, "Learning to refine object segments," in *European conference on computer vision*. Springer, 2016, pp. 75–91.
[31] J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3150–3158.
[32] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.
[33] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 10 012–10 022.
[34] G. Zhan, W. Xie, and A. Zisserman, "A tri-layer plugin to improve occluded detection," 2022. [Online]. Available: https://arxiv.org/abs/2210.10046
[35] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf
[36] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385
[38] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," 2019. [Online]. Available: https://arxiv.org/abs/1911.09070
[39] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
[40] Y. Xu, K.-Y. Lin, G. Zhang, X. Wang, and H. Li, "Rnnpose: Recurrent 6-dof object pose refinement with robust correspondence field estimation and pose optimization," 2022. [Online]. Available: https://arxiv.org/abs/2203.12870
[41] A. Krull, E. Brachmann, F. Michel, M. Y. Yang, S. Gumhold, and C. Rother, "Learning analysis-by-synthesis for 6d pose estimation in rgb-d images," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.
[42] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.
[43] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: https://drake.mit.edu
[44] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, p. 381–395, jun 1981. [Online]. Available: https://doi.org/10.1145/358669.358692
[45] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 3212–3217.
[46] Q.-Y. Zhou, J. Park, and V. Koltun, "Fast global registration," in *European conference on computer vision*. Springer, 2016, pp. 766–782.
[47] ——, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.
[48] D. Aiger, N. J. Mitra, and D. Cohen-Or, "4-points congruent sets for robust surface registration," *ACM Transactions on Graphics*, vol. 27, no. 3, pp. #85, 1–10, 2008.
[49] S. Peng, Y. Liu, Q. Huang, H. Bao, and X. Zhou, "Pvnet: Pixel-wise voting network for 6dof pose estimation," 2018. [Online]. Available: https://arxiv.org/abs/1812.11788