

PARTS-BASED 3D OBJECT POSE ESTIMATION

Ziad Ben Hadj-Alouane

A THESIS

in

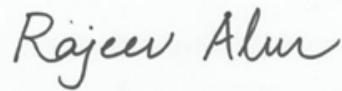
Computer and Information Science

Presented to the Faculties of the University of Pennsylvania in Partial
Fulfillment of the Requirements for the Degree of of Master of Science in Engineering

2020



Professor Kostas Daniilidis
Supervisor of Thesis Signature



Professor Rajeev Alur
Graduate Group Chairperson Signature

PARTS-BASED 3D OBJECT POSE ESTIMATION

© COPYRIGHT

2020

Ziad Ben Hadj-Alouane

This work is licensed under the
Creative Commons Attribution
NonCommercial-ShareAlike 3.0
License

To view a copy of this license, visit

<http://creativecommons.org/licenses/by-nc-sa/3.0/>

Dedicated to my loving parents Nejib and Atidel, and my close friend and brother Houssem. Without your support, I would have not been able to learn and to grow as much as I have.

Thank you from the bottom of my heart.

ACKNOWLEDGEMENT

My special thanks to my advisor Professor Kostas Daniilidis for his guidance and support. I am grateful for the resources he provided me, including extensive expert knowledge on the subject matter, as well as access to the essential infrastructure (GRASP Lab) needed to do my research.

Additionally, I am grateful for my friend Zhengyi (Zen) Luo for helping me with conducting experiments, for reviewing my work, as well as for the long and fruitful discussions about the relevant academic literature.

Alongside Zen, many other friends helped me throughout my college journey, and I wanted to also thank them for their continuous support. They are too numerous to name individually, but I'm sure they know who they are.

I would like to also thank my mentors Karl Shmeckpeper and Bernadette Bucher for helping me navigate and utilize the resources at the GRASP Lab. Many other wonderful people I met at the University of Pennsylvania helped me achieve my goals, and I would like to thank them as well.

Last – but certainly not least – I would like to express my gratitude towards my parents and brother. Without their unyielding love, unconditional support, and doubtless trust, I would not have been able to achieve this work. Thank you for always guiding me through the happiest and toughest times.

ABSTRACT

PARTS-BASED 3D OBJECT POSE ESTIMATION

Ziad Ben Hadj-Alouane

Professor Kostas Daniilidis

The task of 3D object pose estimation consists of locating and orienting an object in 3D space. Many solutions to this problem make use of a complex representation of the object, such as 3D CAD models or point clouds. Unfortunately, this can prove to be unmanageable in real-world settings due to the lack of such high-fidelity representations or due to the growing size of the object catalog. Inspired by recent advancements in 3D object decomposition, we present a method for 3D object pose estimation that instead uses a compact parametric representation. Using this simple representation as a prerequisite, our method first predicts the pose of the parts of the object, then combines them into a final pose estimation. We demonstrate the success of our parts-based method by comparing its performance to that of a standard baseline method.

TABLE OF CONTENTS

ACKNOWLEDGEMENT	iii
ABSTRACT	iv
LIST OF TABLES	vii
LIST OF ILLUSTRATIONS	ix
CHAPTER 1 : Introduction	1
CHAPTER 2 : Background and Related Works	3
2.1 3D Object Pose Estimation	3
2.2 3D Parametric Object Decomposition	24
CHAPTER 3 : Proposed Methodology	31
3.1 Data Generation	31
3.2 Model Training	34
3.3 Model Evaluation	36
CHAPTER 4 : Experiments	37
4.1 Baseline: Direct Pose Regression	37
4.2 Our Method: Parts-Based Pose Regression	38
4.3 Comparison	40
4.4 Quaternions vs. 6D Rotation Representation	43
CHAPTER 5 : Conclusion	46

APPENDIX 47

BIBLIOGRAPHY 47

LIST OF TABLES

TABLE 1 : Experiment Summary (Quaternion Rotation)	40
TABLE 2 : Experiment Summary (6D Rotation)	43

LIST OF ILLUSTRATIONS

FIGURE 1 : Pose Estimation Input-Output	3
FIGURE 2 : Object Detection vs. Pose Estimation	4
FIGURE 3 : Pose Estimation Input Priors	5
FIGURE 4 : Viewpoint Sampling in LINEMOD	8
FIGURE 5 : LINEMOD Multimodalities	9
FIGURE 6 : Scale Invariant Feature Transform (SIFT)	11
FIGURE 7 : MOPED Framework Pipeline	11
FIGURE 8 : Energy Components in Dense Object Coordinate Method .	12
FIGURE 9 : PoseCNN Architecture	14
FIGURE 10 : Hough Voting for Object Centers	15
FIGURE 11 : Pose Ambiguities Due to Object Symmetry	16
FIGURE 12 : DenseFusion’s Iterative Pose Refinement	18
FIGURE 13 : Bin and Delta Model Architecture	19
FIGURE 14 : Toy Example of the Geodesic Bin and Delta Model	21
FIGURE 15 : Stacked-Hourglass CNN	22
FIGURE 16 : Semantic Keypoints Method Outputs	23
FIGURE 17 : Cuboid Parametric Representation	24
FIGURE 18 : Cuboid Training Architecture	27
FIGURE 19 : Superquadric Shapes	28
FIGURE 20 : Comparing Cuboids and Superquadrics	28
FIGURE 21 : ShapeNet Chair Category	32
FIGURE 22 : Samples of Generated Data	32
FIGURE 23 : Chair Instances used in Experiments	33

FIGURE 24 : Example of Superquadric Decomposition of a Chair	34
FIGURE 25 : Parts-Based Pose Regression Architecture	35
FIGURE 26 : Baseline Loss Curves - Instance #1	37
FIGURE 27 : Baseline ADD Curve - All Instances	38
FIGURE 28 : Our Method's Loss Curves - Instance #1	39
FIGURE 29 : Our Method's ADD Curve - All Instances	39
FIGURE 30 : ADD Comparison	41
FIGURE 31 : Chair Instance #3: Baseline vs. Our Method	42
FIGURE 32 : 6D Rotation - Our Method	44
FIGURE 33 : 6D Rotation - Baseline	45
FIGURE 34 : Baseline Loss Curves	47
FIGURE 35 : Our Method's Loss Curves	48
FIGURE 36 : Chair Instance #1: Baseline vs. Our Method	49
FIGURE 37 : Chair Instance #2: Baseline vs. Our Method	50

1. Introduction

Estimating the orientation (pose) of an object in 3D space generally requires a representation of the said object. Complex and rich representations such as 3D computer-aided design (CAD) models or 3D point clouds can help increase the accuracy of predictive models due to the high-fidelity information they pack. Unfortunately, these representations come at a cost: (1) they are hard to acquire, (2) are non-negligible in size, and (3) their numbers tend to grow as the set of objects expands. As such, many solutions to 3D object pose estimation try to either not use a complex representation or attempt to approximate it through other means.

However, with the field of 3D object decomposition seeing a reignited interest, it is likely that future solutions to 3D pose estimation can make use of a rich yet compact representation. The ability to accurately decompose an object into a small set of parts may allow complex representation to be directly mapped to "compressed" counterparts. As a consequence, many solutions to 3D pose estimation may beneficially relax their requirements on the choice of object representation.

Inspired by this guiding insight, we propose a method that combines recent successes in 3D object pose estimation and 3D parametric object decomposition. The contributions of this thesis can be summarized as follows:

- A data generation pipeline that augments a single object pose into a set of pose labels, one per object part.
- An architecture for predicting the pose of instances of objects using a parts-based paradigm.

This thesis is organized as follows:

- **Background and Related Works** section, in which we describe different state-of-the-art methodologies in 3D object pose estimation and 3D parametric object decomposition.
- **Propose Methodology** section, in which we describe our method from the data generation pipeline, to the model training and evaluation procedures.
- **Experiments** section, in which we analyze, compare, and discuss the various results obtained from evaluating the pose predictors.

2. Background and Related Works

Our work builds upon two main fields: 3D object pose estimation and 3D parametric object decomposition.

2.1. 3D Object Pose Estimation

2.1.1. Problem Definition and Importance

Given an input image, the problem of object pose estimation consists of determining where an object of interest is in space and how it is oriented. Concretely, the pose of an object is defined by a 3D orientation (rotation) and a 3D translation comprising six degrees of freedom (6DOF). Both the 3D rotation and the translation are often expressed relative to the camera coordinate system. The camera intrinsics are assumed to be known (since these are readily available). A simple illustration of the problem is shown in Figure 1.

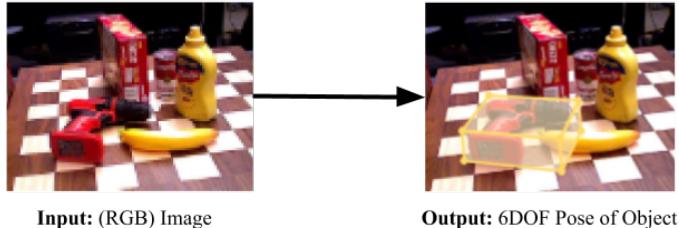


Figure 1: Typically, a pose estimator predicts the 6DOF pose of an object given a colored image containing the object of interest. In this case, the object of interest is the red drill.¹

Pose estimation has been of primary importance for a variety of computer vision tasks. Most notably, it is crucial to be able to accurately and efficiently estimate the pose of objects for robotic manipulation or for any application that involves interacting with real world objects. In augmented reality, 6DOF pose estimation enables virtual

¹Image credit Wang et al. (2019)

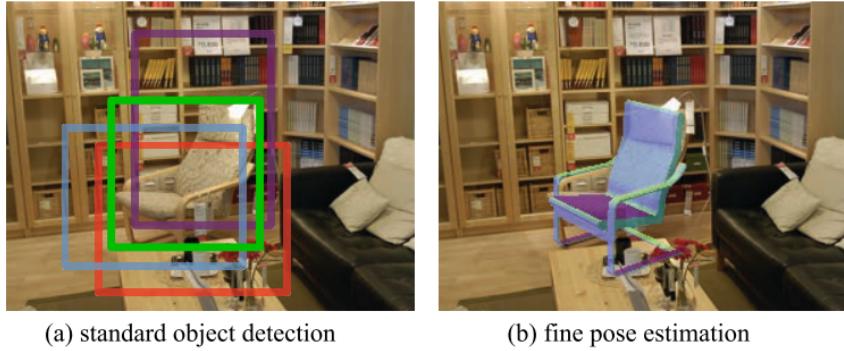


Figure 2: Illustration depicting the differences between object detection (left) and pose estimation (right). An agent attempting to sit on a chair based on correct detections by an object detector could end up sitting anywhere in the vicinity. Pose estimation may offer a more precise solution.²

interaction and re-rendering of real world objects. As such, pose estimation has been the focus of intense research in recent years, and is still considered a non-trivial problem in many contexts.

It is important to distinguish between pose estimation and object detection. Lim et al. (2014) describe the problem of an autonomous agent attempting to find a chair to illustrate this difference. Using a state-of-the-art object detection system, the agent finds a number of correct detections of chairs as shown in Figure 2. However, with the estimated set of correct detections, the agent could end up sitting anywhere from the bookshelf, to the floor. In contrast, estimating the pose of detected objects may provide a better understanding of the environment, thus enabling more precise interactions.

2.1.2. Input Assumptions

In its basic form, pose estimation is performed on inputs in the form of RGB colored images (i.e., red, green, and blue color channels). In some models, some input priors

²Image credit Lim et al. (2014)

³Image credit Wang et al. (2019)

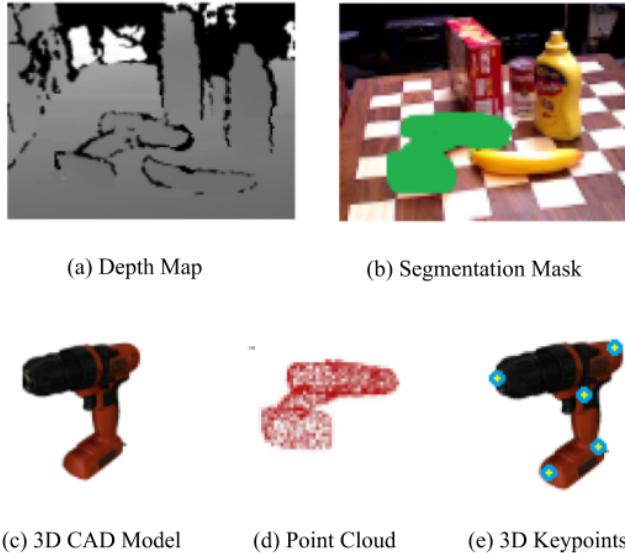


Figure 3: Common examples of input priors used in state-of-the-art 3D object pose estimation. Note that it is possible to infer some of these priors using a combination of them (e.g., a segmentation mask and a depth map could be used to generate a point cloud).³

may be used to add useful information. For example, some methods pre-process the input images through a segmentation pipeline to generate a binary mask around the object of interest. Such pre-processing steps are considered to be fairly accurate and efficient due to the proliferation of fast and successful visual recognition models.

Other priors are more difficult to obtain through simple image pre-processing. For example, some models require the availability of a 3D CAD model for each object. It is also typical to see models requiring the availability of depth information (i.e., RGB-D images) which has been made easier to obtain due to commercial depth cameras (e.g., Microsoft’s Kinect depth camera). In Section 2.1.4, we discuss methods that use a combination of the above restrictions to illustrate the variety of modern solutions. Additionally, a common list of input priors is shown in Figure 3 for clarity.

Finally, assumptions can be made about the objects of interest to increase or decrease

the difficulty of the task. For example, certain ambitious models tackle class-based (e.g., all chairs) or cross-category (e.g., any chair, cup, car etc.) pose estimation. Others relax the prediction space and focus only on solving instance-based (e.g., a specific chair) pose estimation.

2.1.3. Challenges

The challenges of pose estimation can be categorized into data availability and data quality issues.

Data Availability

It turns out that many methods require high fidelity 3D models of the objects of interest to be available. Although modern 3D reconstruction and scanning techniques can generate 3D models of objects, they typically require significant effort. It is easy to see how building a 3D model for every object is an infeasible task.

Additionally, RGB-D (color and depth) based methods cannot be made real-time if deployed on most mobile phones and tablets due to the unavailability of depth data. This is typical for mobile augmented reality applications. As such, substantial research is dedicated to estimating poses of known objects using RGB images only.

Furthermore, pose data is difficult to collect efficiently. A large number of images need to be taken, and generally need associated ground-truth pose label to be measured accurately. Automating this process is not trivial and typically requires careful human manipulation.

Data Quality

As we will elaborate on in Section 2.1.4, many classical approaches rely on extracting object features from RGB-D data and performing 3D-2D correspondence grouping. These methodologies heavily rely on handcrafted features and template matching

procedures that are empirically shown to have limited performance in presence of heavy occlusion and lighting variation. This issue is accentuated when considering the diversity of backgrounds that appear in real images. Indeed, many real world scenes are cluttered, and consequently difficult to deal with.

Finally, methods that train a single network for multiple objects (class-based or category-based) typically suffer from drops in accuracy as number of objects increases. This is due to large variation of object appearances depending on the pose, lighting conditions, and occlusion.

2.1.4. Methodologies

With the recent proliferation of Graphics Processing Units (GPUs), many publicly available contributions to object pose estimation shifted from hand-engineered classical methods towards deep-learning (DL) based approaches. DL approaches, in the context of 3D object pose estimation, leverage the efficacy of Convolutional Neural Networks (CNN). These types of networks are known to automatically learn hierarchical features of image data, as demonstrated in massive successes in the areas of image recognition and image segmentation. As such, we decided to split the works in the literature into two types: classical methods, and deep-learning methods.

Classical Methods

Classical methods can be split into three main areas: template search-and-match methods, keypoint correspondence methods, and dense feature methods. We will present an overview of a past notable solution in each of these three categories.

TEMPLATE SEARCH-AND-MATCH: Template methods typically use RGB-D images to identify global (e.g., appearance, 3D shape etc.) features of the object. Once the features are identified, the input is compared to a set of template images that

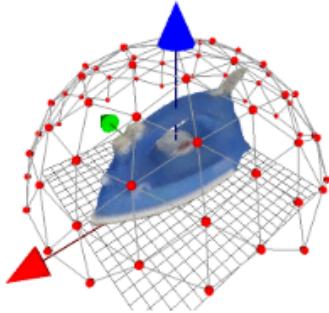


Figure 4: Red vertices represent the virtual camera centers used to generate templates. The camera centers are uniformly sampled⁴.

are recorded using different viewpoints. Since each viewpoint is associated with a 3D pose, matching the input against a template essentially provides a coarse estimate of the pose of the object.

In Hinterstoisser et al. (2013), the authors build the template database offline by rendering the CAD model of the object of interest. The rendered viewpoints are sampled uniformly around an icosahedron, as shown in Figure 4.

The authors mention that this offline sampling approach has two main advantages over other online sampling approaches:

1. Online template building requires careful human (or robot) interaction with the camera and the environment. It is thus time consuming to generate templates and learn features from them as the system is running.
2. It takes an educated user to be able to collect a well-sampled training set of templates, let alone to recover the pose of these templates accurately.

Once the templates are collected, the system learns the following features: color gradients (computed from the template RGB image), and associated surface normals

⁴Image credit Hinterstoisser et al. (2013)

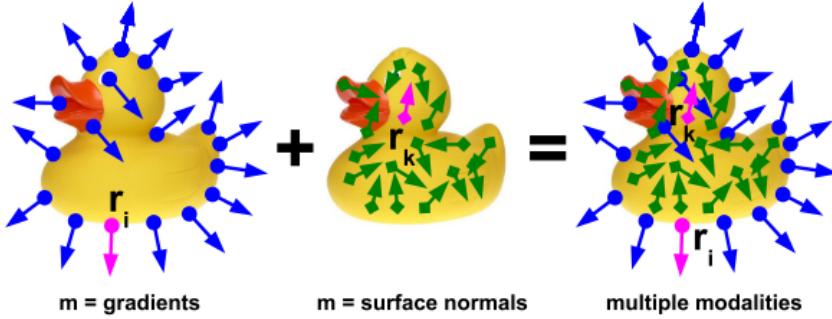


Figure 5: A duck model with two discretized modalities: color gradients at the contour (left), surface normals in the interior (middle), both modalities at the same time (right).⁵

(computed from the CAD model). To be specific, only color gradients around the contour of the object silhouette are kept, since the authors assume textureless objects that typically exhibit little to no texture variation in the interior of the silhouette. In contrast, surface normals are only considered at the interior of the object. The authors argue that the surface normals at the borders of the projected 3D objects are seldom reliably estimated. Figure 5 illustrates these collected features.

Template matching is done following the LINEMOD technique presented in Hinterstoisser et al. (2011). In short, LINEMOD computes similarities between database templates and patches of the input image by considering multiple modalities (e.g., color, depth). In the case of color gradient and surface normal features, the dot product of the features at the template location and the input image patch is sufficient. Finally, estimating the pose using detected templates is done by using the Iterative Closest Point (ICP) algorithm. The crux of the algorithm is to align two input point clouds of an object. In the case of Hinterstoisser et al. (2011), the input point clouds are (1) the ground truth CAD model associated with the template and (2) the input depth map segmented around the object of interest.

⁵Image credit Hinterstoisser et al. (2013)

KEYPOINT CORRESPONDENCE: Keypoint correspondence methods typically have three stages:

1. *Keypoint Extraction*, which intuitively refers to identifying local points of interest around the object. Typically, a keypoint will have an associated pose as well as a descriptor that generally describes the local region around it. Descriptors usually contain summarized local information such as color gradients. Note that a keypoint can be defined in 2D space (e.g., in the input image) or in 3D space (e.g., typically the CAD model of the object).
2. *Keypoint 2D-3D Correspondence*, which refers to matching the 2D keypoints to corresponding 3D keypoints in a database of processed 3D CAD models.
3. *Pose Estimation*, which is where most contributions in this space vary. In short, the previously estimated 2D-to-3D correspondences are used to make a pose hypothesis. Solutions to the *Perspective-n-Point* (PnP) problem such as EPnP (Lepetit et al. (2009)) are often used to extract such a hypothesis.

An example of a keypoint method is the MOPED framework by Collet and Martinez. MOPED builds a database of 3D keypoint features of 91 CAD models by running rendered viewpoints through SIFT. SIFT was developed by Low (2004), and is a state-of-the-art keypoint extraction method that is known to be scale and rotation invariant, as well as robust to a wide range of noise transformations, illumination changes, and 3D viewpoint alterations. Figure 6 illustrates the result of running SIFT on an input image.

At inference time, MOPED extracts the 2D features of the input images which are

⁶Image credit Low (2004)

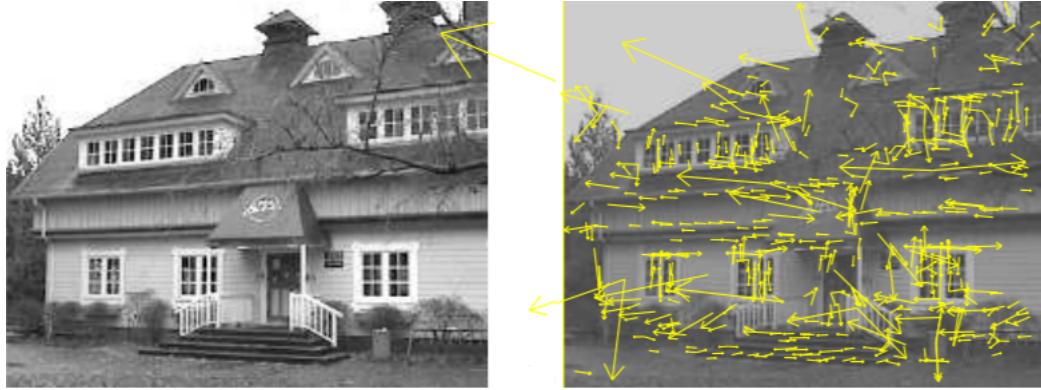


Figure 6: Illustration of the output of SIFT’s keypoint extraction. Each keypoint is represented by a vector to visualize its scale, rotation, and position.⁶

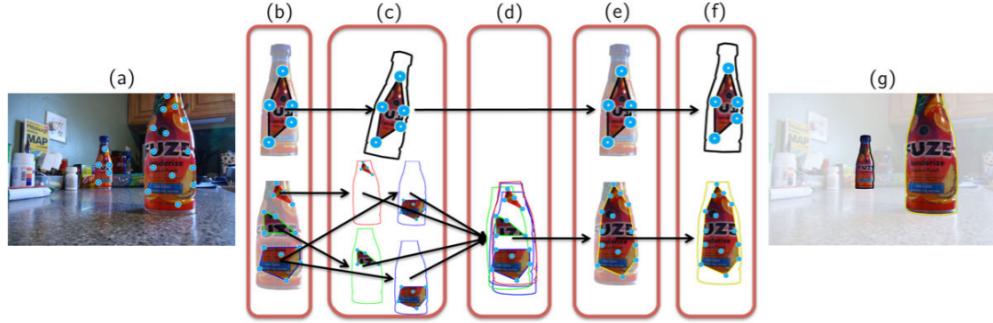


Figure 7: Illustration of the MOPED pose estimation pipeline. (a) keypoint extraction with SIFT, (b) keypoint matching, (c) keypoint clustering by object ID, (d-e) additional clustering of clusters, (f) pose hypothesis refinement, (g) final pose output.⁷

then matched against 3D object features stored in the database. Features are then clustered by object ID, which intuitively means that spatially-close features generally belong to the same object. At the final stages of MOPED, RANSAC-like outlier detection is used to filter the clusters and generate pose hypotheses for each cluster. A pose refinement and merging step is then applied to all the clusters to generate object-level hypotheses for each detected object in the input image. Figure 7 illustrates the MOPED framework in detail.

⁷Image credit Collet and Martinez

DENSE FEATURE METHODS: Brachmann et al. (2014) describe the primary shortcoming of template-based methods as the reliance on matching the complete template to a target image. This essentially means that the object is encoded in a particular pose with one “global” feature. This empirically leads to downgraded performance when the object is occluded in the scene. The authors recommend encoding the object with dense “local” features instead. They propose a hierarchical matching system based on a random forest. The model first predicts dense (local) object coordinates, and uses them to perform dense correspondences to recover the object pose.

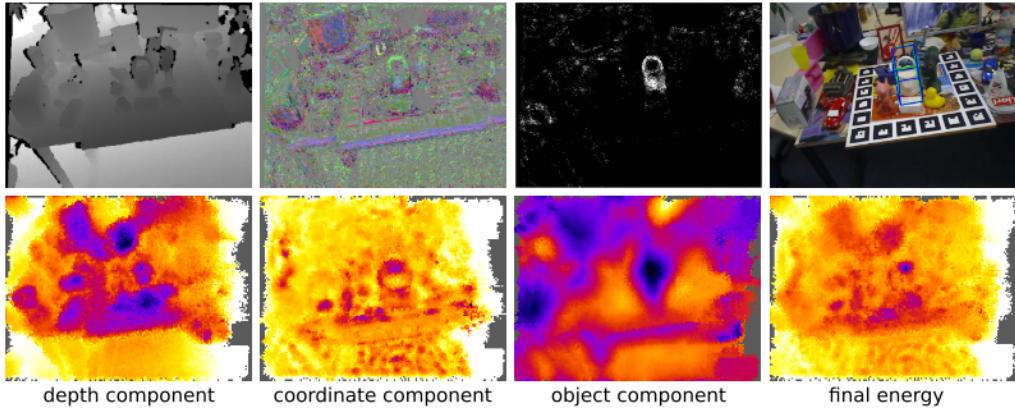


Figure 8: The energies were calculated for different poses and projected into image space using minimum projection. White stands for high energy dark blue for low energy.⁸

The random forest they use is comprised of a set of decision trees. Each pixel of the input RGB-D image is classified by each tree, and end up in one of the tree’s leaves. This hierarchical structure is trained in a way that allows (1) gaining information about which object a pixel might belong to and (2) the approximate position of the object. Predicting the pose of the object is done after training the forest. The authors formulate the pose estimation as an energy optimization problem comprised by three components: a depth energy component (dense features), an object seg-

⁸Image credit Brachmann et al. (2014)

mentation component (sparse global features), and a coordinate component (dense features). Computing the energy requires comparing synthetic images rendered using the hypothesized pose with the observed depth values and the results of the forest. Figure 8 illustrates the different energy components.

Deep-Learning Methods

In computer vision, most deep-learning methods employ some convolutional component in the network architecture. In the case of object detection and recognition, Convolutional Neural Networks (CNNs) have seen wide successes. CNNs are preferred in this space because of their ability to automatically learn features from raw image data as well as for representing images in an abstract and hierarchical fashion. This is done by successively stacking convolutional and pooling layers. Once a suitable network architecture is defined and the corresponding model is trained, CNNs can cope with a large variety of object appearances and classes.

However, in the context of 3D object pose estimation, convolving over the input images is just a component. The innovation in this space comes from defining input constraints (e.g., RGB, RGB-D, 3D CAD models, etc.), supervision constraints (e.g., supervision with pose labels, no supervision, etc.) as well as careful loss function engineering (e.g., reconstruction loss, intermediate representation losses, direct pose regression loss etc.). In the following section, we will explore notable works that use such deep-learning methods.

RGB IMAGE + POSE LABELS + CAD MODEL: Xiang et al. (2018) propose a generic end-to-end object detection and 3D object pose prediction framework named PoseCNN. The authors designed a CNN that takes as input a single RGB image and predicts the classes and poses of objects captured in the image. Training is

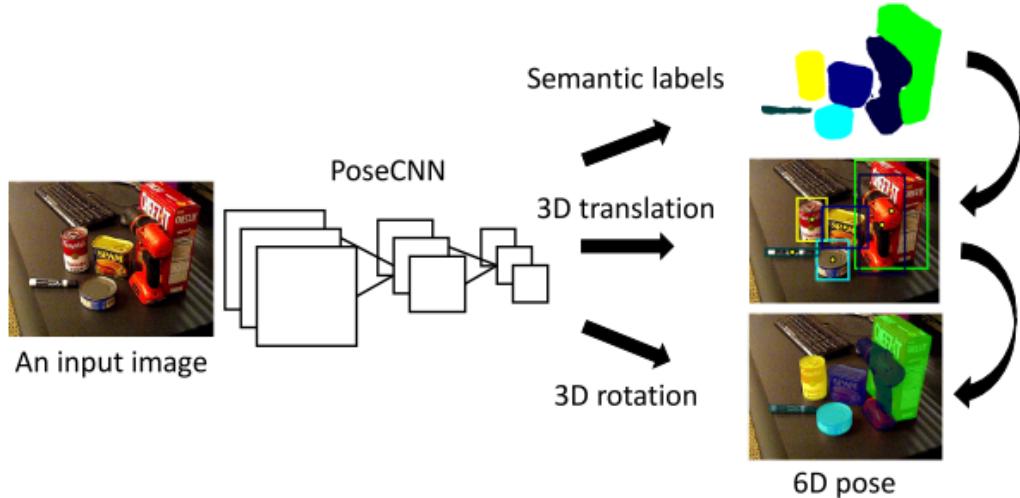


Figure 9: PoseCNN runs first predict semantic labels, which are used to predict object locations. The combination of these outputs is used to finally predict the rotation of each object.⁹

supervised by providing ground-truth pose labels with respect to the 3D model space of the object in consideration. As such, a 3D CAD model of every object must be provided in training time.

The CNN runs in three stages as illustrated by Figure 9 and as detailed below:

1. *Semantic Labeling* stage, wherein the RGB input image is convolved down to a dimension of 64 (with additional pooling and summing of feature maps), then deconvolved up to the original size of the input image. These embedded features are then passed to a fully-connected (FC) layer of size n , where n is the number of semantic classes (i.e., object types). The goal of this stage is to assign for each pixel a probability of belonging to an object class. Softmax cross-entropy loss is used to predict a probability distribution for each pixel.
2. *3D Translation Estimation* stage, wherein the network predicts the 2D object

⁹Image credit Xiang et al. (2018)

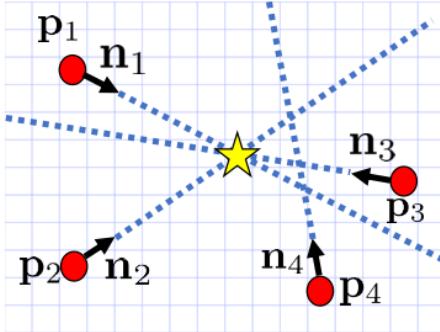


Figure 10: Each pixel casts a vote in the form of a unit vector pointing towards its prediction of the center of the object. ¹⁰

center in the image as well as 3D depth value for that center. Specifically, for each pixel the network predicts a unit vector that points towards the predicted center, as well as a depth value for that center. Finally, each pixel’s center predictions are weighted by its semantic labeling, and fed into a voting layer to decide on a final object center prediction. Figure 10, illustrates the voting process. The final 2D object center prediction (and the depth value) is used to recover the 3D object center assuming a pinhole camera model.

3. *3D Rotation Estimation* stage, wherein the semantic labels and center predictions as well as the convolved image features are pooled together to generate a rotation prediction for each object. The authors represent a rotation by a quaternion, which is a widely used 4D vector representation of rotations. In this stage of training, the network requires the presence of a 3D CAD model for each object. This is because the network’s rotation loss function requires sampling 3D points on a CAD model representing the object. Formally, the loss function is defined by Equation 2.1, where \mathcal{M} is the set of 3D model points, m is the number of points, and \mathbf{q} and $\hat{\mathbf{q}}$ are the ground-truth and predicted

¹⁰Image credit Xiang et al. (2018)

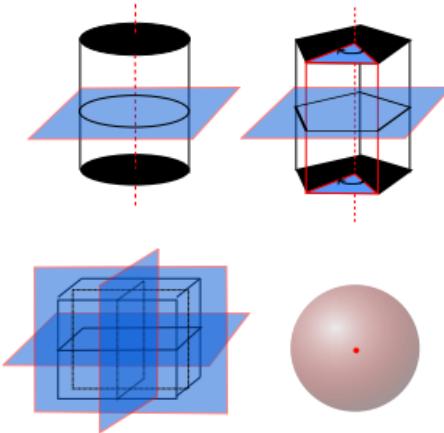


Figure 11: A subset of symmetry cases that give rise to pose ambiguities. For example, a sphere has the same rotation regardless of the viewpoint, even though we might arbitrarily assign to a view a specific rotation (i.e., a one-to-many mapping from image to pose)¹¹

quaternions respectivley. We note that this loss measures the average squared distance between every point on the predicted rotated model and the *closest* point on the ground-truth rotated model. This design avoids penalizing pose predictions that are deemed incorrect due to symmetries in the model. Figure 11 describes a subset of these symmetric cases.

$$\text{SLoss}(\mathbf{q}, \hat{\mathbf{q}}) = \frac{1}{2m} \sum_{\mathbf{x}_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \|R(\hat{\mathbf{q}})\mathbf{x}_1 - R(\mathbf{q})\mathbf{x}_2\|^2 \quad (2.1)$$

RGB-D IMAGE + POSE LABELS + CAD MODEL: In the previous PoseCNN method, we note that the predicted poses can be refined using Iterative Closest Point (ICP) if depth data is available. In Wang et al. (2019), the authors suggest that decoupling color and depth information in such a way is costly (i.e., the extra refinement and post-processing steps are not ideal). Instead, the authors propose DenseFusion, a generic framework for estimating poses of known objects that fuses RGB and depth

¹¹Image credit Sundermeyer et al. (2018)

data in a novel way.

DenseFusion begins by segmenting the input RGB image into an $N + 1$ -channelled semantic segmentation mask to predict the existence of up to N different objects in the scene. The authors reuse the same segmentation network used by PoseCNN. Then, DenseFusion extracts information from the color and depth channels separately. Intuitively, the authors argue that color embeddings and geometric (depth) embeddings reside in different spaces, requiring separate processing pipelines. The following is an overview of these embeddings:

1. *Depth Embeddings*: The network first converts the segmented depth pixels into point clouds using the known camera intrinsics. Thus, if the segmentation network detected k semantic segments, then k point clouds will be generated. Each point cloud is then processed individually to generate embeddings using network that uses symmetric reduction modules (e.g., max-pooling, average-pooling, etc.), pioneered by Qi et al. (2017)'s PointNet. The basic idea is that the unordered-set format of point clouds should have no impact on the learned features from said point clouds.
2. *Color Embeddings*: The network is a CNN-based encoder-decoder that maps each pixel in the segmented region into a higher dimension embedding that encodes appearance information at that input location.

Thus far, the network extracted two heterogeneous pieces of information: depth embeddings encoding geometric information at each point in the point cloud, and color embeddings encoding appearance information at each pixel in the image. DenseFusion fuses these two embeddings by creating pixel-wise dense intermediate vectors. This is done by mapping each point in the point cloud to its corresponding pixel location,

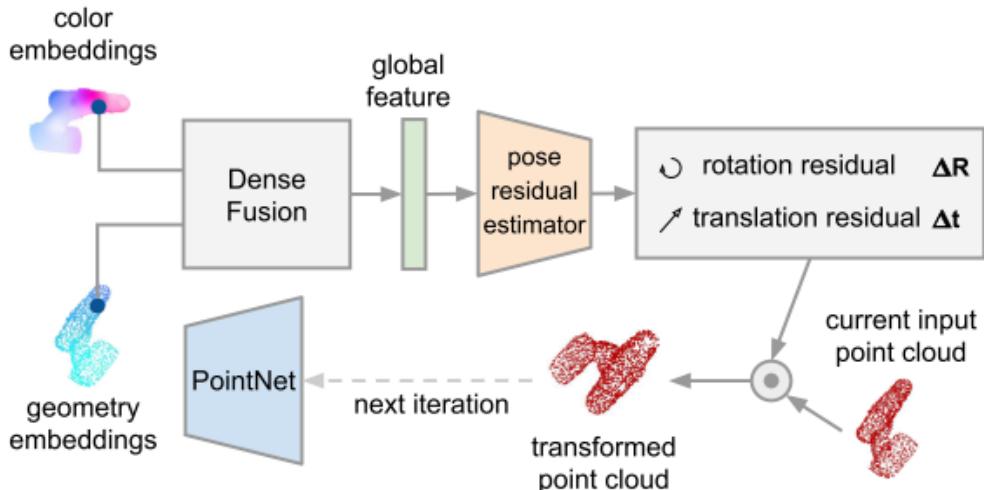


Figure 12: The predicted poses are used to transform the input point cloud, which is then used to iteratively predict the next residual pose. This differentiable iterative refinement speeds up the inference process as ICP is not needed anymore.¹²

and concatenating the color embedding with the depth embedding. Therefore, the network generates a pixel-wise dense vector for each point on the point cloud. Finally, each per-pixel dense vector is fed into a simple regression network that predicts the pose of its associated object. This network uses the same loss function defined by Equation 2.1.

Instead of using ICP to refine the predicted poses, DenseFusion makes use of the available depth channel to refine the pose as the network trains (i.e., pose refinement is built into the network). The network considers the previously predicted pose as an estimate of the canonical frame of the target object, and transforms the input point cloud obtained by the depth channel into this estimated frame. Intuitively, the transformed point cloud more accurately depicts the final target pose. The network thus continues to refine the pose by predicting residual poses in an iterative manner.

Figure 12 illustrates this process.

¹²Image credit Wang et al. (2019)

RGB IMAGE + POSE LABELS: In Mahendran et al. (2018), the authors tackle the problem of 3D object pose prediction from a single RGB image without the use of a 3D CAD model. This methodology uses a CNN to perform a mix of classification and regression, as the authors note that using one or the other alone comes with disadvantages:

1. *Classification Only:* Discretizing the orientation space into bins can lead to coarse (thus inaccurate) pose predictions. Many such networks will use a cross-entropy loss function to perform standard classification, which neglects the Riemannian attributes of the orientation space (as opposed to simpler Euclidean attributes iconic of flat spaces).
2. *Regression Only:* These overcome the descretization issues that arise from classification by predicting continuous poses. However, the authors note that current regression methods still employ Euclidean distance measures to compare poses, which again neglects Riemannian attributes. Additionally, regression methods do not fully deal with symmetry issues that are displayed in Figure 11, as they generally predict a simple pose without a confidence score.

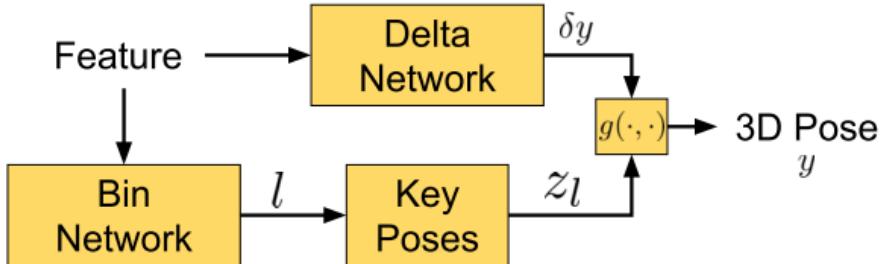


Figure 13: The feature network (ResNet-50) output is shared by both the Bin and the Delta networks. The predictions of the latter networks are combined with a combination function $g(.,.)$. In the case of the Geodesic Bin & Delta model, $g(.,.)$ is simply the addition function. ¹³

This mixed method is dubbed Bin & Delta, wherein a coarse pose is first predicted using the classifier, and a refinement is applied on the prediction using the regressor. The paper showcases a variety of Bin & Delta model variations (e.g., geodesic, Riemannian, probabilistic, relaxed, etc.). We chose to present the geodesic Bin & Delta model for its simplicity and interpretability. A summary of the network is provided in Figure 13.

Euclidian Loss vs. Geodesic Loss: Assuming the use of a quaternion representation, the standard Euclidean loss between a ground-truth rotation \mathbf{q} and a predicted rotation $\hat{\mathbf{q}}$ is defined by Equation 2.2.

$$\text{ELoss}(\mathbf{q}, \hat{\mathbf{q}}) = \|\mathbf{q} - \hat{\mathbf{q}}\|^2 \quad (2.2)$$

In contrast, the geodesic loss that takes into account the Riemannian attributes of the orientation space is defined by Equation 2.3. The authors remark that such distance (loss) function computes the shortest distance between two points along the Riemannian manifold, which outperforms Euclidean distance as measure of the “closeness” in the context of orientation representations. This is the loss adopted in the Bin & Delta models.

$$\text{GLoss}(\mathbf{q}, \hat{\mathbf{q}}) = 2 \cos^{-1}(|\langle \mathbf{q}, \hat{\mathbf{q}} \rangle|) \quad (2.3)$$

Feature Network: Components in the overall network described by Figure 13 share a common feature network used to extract features from input RGB images. Common to transfer-learning approaches, the authors use the ResNet-50 by He et al. (2016)

¹³Image credit Mahendran et al. (2018)

minus the last output layer.

Bin Network (Classifier): The pose space is first discretized using K-Means: the training data is used to split the pose labels into K similar categories. The output features of the feature network are then fed into the bin network to classify the pose of the object of interest. Standard cross-entropy loss is used, and the output of the bin network is a pose label that falls in one of the K clusters. The predicted pose label has a corresponding pose \mathbf{q} mapped to it (e.g., the mean pose of the K-means cluster).

Delta Network (Regressor): The same features generated by the feature network are finally fed into the delta network to predict a residual pose $\delta\mathbf{q}$. This residual pose can be thought of as an "additional" pose applied to the coarsely predicted pose in the previous stage. The loss used is the geodesic loss previously defined, with inputs being the ground-truth pose \mathbf{q}^* and the combined pose $\frac{\mathbf{q}+\delta\mathbf{q}}{\|\mathbf{q}+\delta\mathbf{q}\|}$ (the normalization is applied assuming quaternions are used). The process is nicely interpretable as shown in Figure 14.

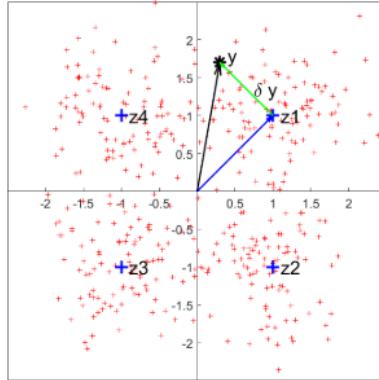


Figure 14: Points on the plane (red) represent various poses. The space is split into four clusters, with each cluster corresponding to a coarse pose (blue z points). The bin network predicts the coarse pose (blue vector), and the delta network refines the prediction (green vector). The overall pose is noted as y . ¹⁴

RGB IMAGE + CAD MODEL: The previous methods we have explored used ground-truth pose labels. In reality, it is challenging to collect accurate and robust pose data. Pose has to be defined with respect to a specific frame of reference, and most likely with a uniform camera model. Additionally, pose can be represented in many ways (e.g., quaternions, axis-angles, Euler angles, transform matrix, etc.). Thus, there is a general interest in predicting poses without providing the learning model with ground-truth pose labels (i.e., pose-unsupervised). Pavlakos et al. (2017)’s work circumvents this issue by training a model that is pose-unsupervised. The core idea is to use keypoint labels instead of pose labels to predict the 6DOF object pose.

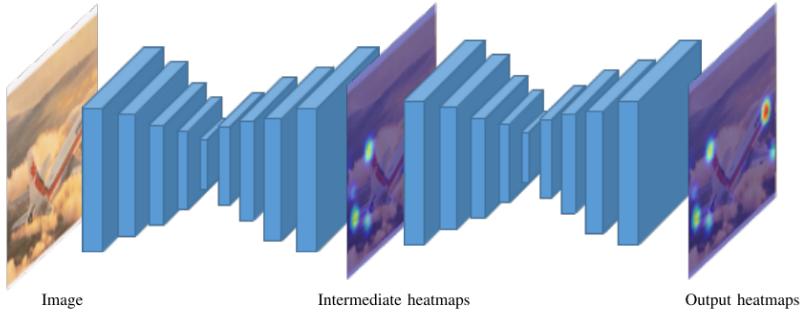


Figure 15: The stacked-hourglass model used to output semantic keypoints of an object given an RGB image.¹⁵

Assuming that objects of interests are given as an input image with associated bounding boxes obtained from off-the-shelf object detectors (e.g., Ren et al. (2017)’s Faster-RCNN), the following two steps are executed:

1. *Keypoint Localization:* This step consists of a CNN that takes as input a single RGB image and produces a set of heatmaps. Each heatmap corresponds to a keypoint detection, and the corresponding heatmap intensity is proportional to

¹⁴Image credit Mahendran et al. (2018)

¹⁵Image credit Pavlakos et al. (2017)

the network’s detection confidence. The network consists of two stacked hour-glass modules. Each module essentially downsamples the input down to 4×4 feature maps, then upsamples those back to the original input size. The first module produces coarse heatmaps that can be refined with intermediate supervision, while the last module produces the network’s final answer. Supervision is done on ground-truth heatmaps generated by centering a gaussian distribution around each keypoint, with a standard deviation set to one. The network architecture is shown in Figure 15.

2. *Pose Estimation*: This step seeks to fit the predicted 2D keypoints from the previous step to 3D keypoints. The authors suggest two approaches: (1) use PnP just like in typical keypoint correspondence methods or (2) fit a deformable shape model to the 2D keypoints. The former requires a CAD model per instance, while the latter only requires a deformable model per class of objects. A visualization of the process outputs is shown in Figure 16.

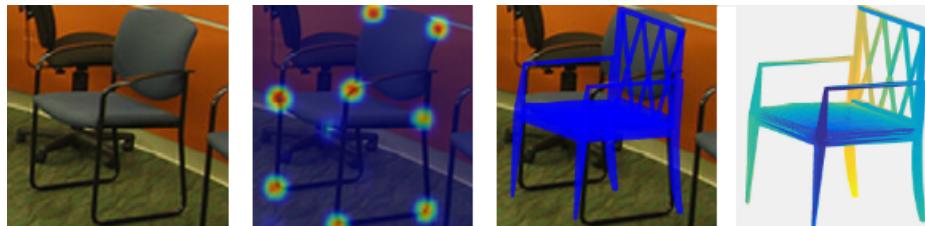


Figure 16: From left to right: input RGB image, semantic keypoints from keypoint localization, deformable shape (chair) overlaid over the input image, 3D model with predicted pose.¹⁶

¹⁶Image credit Pavlakos et al. (2017)

2.2. 3D Parametric Object Decomposition

2.2.1. Problem Definition and Importance

In modern solutions to various computer vision problems (including 3D object pose estimation), the use of high-dimensional and complex 3D representations is prevalent. Examples of such representations are the familiar point clouds and 3D CAD models. They typically provide a high level of fidelity, enabling capturing finer shape and appearance intricacies. However, due to the large amount of information they typically encode, the use of such complex representations been recognized as a bottleneck in some areas of computer vision. For example, Park et al. (2019a) mention that it is unfeasible to expect the availability of a 3D CAD model for every object in 3D object pose estimation.

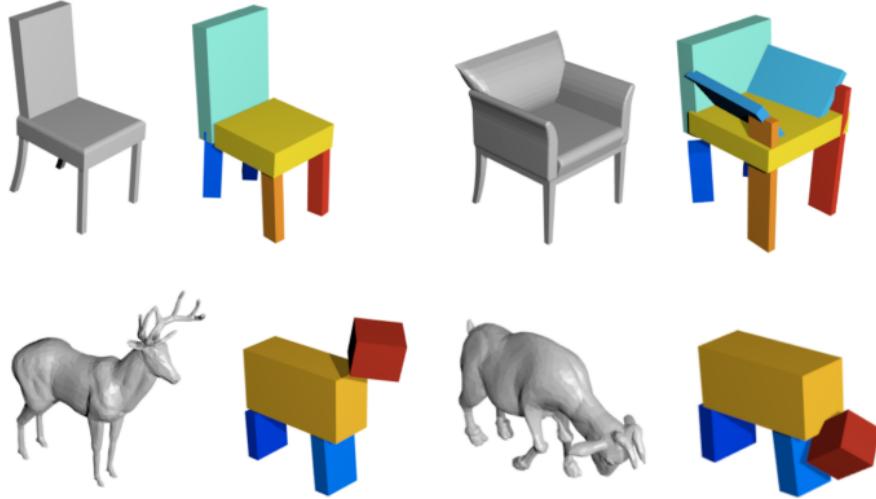


Figure 17: Illustrations of complex meshes (gray) and their corresponding cuboid volumetric representations (colored).¹⁷

As such, the community has seen a reignited interest in representing 3D objects in a compact manner. Specifically, recent works investigated the problem of object decom-

¹⁷Image credit Tulsiani et al. (2017)

position. That is, given an input object O with an associated high-dimensional input I (e.g., 3D CAD model, point cloud, etc.), the goal is to predict up to M distinct volumetric parts that form O . An example of volumetric primitives would be a generalized cylinder as introduced by Thomas Binford in 1971. The key intuition is that the general shape, structure, and hierarchy of the 3D object is preserved even when represented by simpler primitive shapes. Examples of such compact representations can be seen in Figure 17.

2.2.2. Methodologies

Just like in 3D pose estimations, the availability of commercial GPUs and the advances in deep-learning (e.g., CNNs) allowed for more innovation in this space. To be specific, Tulsiani et al. (2017) explored a data-driven approach to represent objects as a collection of cuboids. Paschalidou et al. (2019) then followed-up with an approach that uses a more refined representation that uses superquadrics. We provide an overview of both of these recent works in the following subsections.

Cuboids

Tulsiani et al. (2017) use the simplest volumetric primitives – rigidly transformed cuboids – to represent objects as seen in Figure 17. The authors emphasize that while non-data-driven approaches had successes in solving instance-based representation using handcrafted cues, this data-driven approach provides a consistent representation across instances: the output is an *indexed* set of primitives instead of an *unordered* set of primitives.

Tulsiani et al. (2017)'s approach uses a CNN to predict these primitives in an unsupervised way. The authors argue that even without direct labels of primitives, it is possible to engineer a loss function that measures whether the predicted clusters of primitives matches the target object.

REPRESENTING CUBOIDS: Each predicted primitive is encoded as a triple $(\mathbf{z}, \mathbf{q}, \mathbf{t})$ where \mathbf{z} represents the parametric shape of the *untransformed* primitive, and (\mathbf{q}, \mathbf{t}) represent its pose (rotation, translation). Intuitively, \mathbf{z} is the ‘what’ and (\mathbf{q}, \mathbf{t}) are the ‘where’ of the primitive. In particular, we assume an origin-centered cuboid with $\mathbf{z} = (w, h, d)$, its extents in the three dimensions. The authors make use of the (squared) distance field function defined by Equation (2.7), where $\mathbf{p} = (p_x, p_y, p_z)$ is an arbitrary 3D point. This function computes the distance between \mathbf{p} and the closest point on the cuboid. It is clear that if \mathbf{p} is on the surface/inside the cuboid, the distance field will output 0.

$$\mathcal{C}_{cub}(\mathbf{p}; \mathbf{z})^2 = (|p_x| - w)_+^2 + (|p_y| - h)_+^2 + (|p_z| - d)_+^2 \quad (2.4)$$

Formally, the distance field of the *transformed* cuboid parametrized by \mathbf{z} and computed on a point \mathbf{p} is denoted by $\mathcal{C}_{cub}(\mathbf{p}'; \mathbf{z}, \mathbf{q}, \mathbf{t})$, where $\mathbf{p}' = R^{-1}(\mathbf{p} - \mathbf{t})$ and R^{-1} refers to the inverse application of the quaternion \mathbf{q} .

COVERAGE AND CONSISTENCY LOSSES: Comparing the target object O and a predicted transformed cuboid $P_m = (\mathbf{z}_m, \mathbf{q}_m, \mathbf{t}_m)$ requires a notion of sampling points on either shapes. We denote this sampling process with $\mathbf{p} \sim S(O)$ (or $\mathbf{p} \sim S(P_m)$). Concretely, the authors define a coverage loss and a consistency loss. The former refers to how well the predicted shape covers the target object. Equation 2.5 describes this loss, as it essentially computes the minimum distance between a point sampled on the object O and the surface of any of the predicted shapes.

$$L_{cov}(P_m, O) = \mathbb{E}_{\mathbf{p} \sim S(O)} \left\| \min_m \mathcal{C}_{cub}(\mathbf{p}'; P_m) \right\|^2 \quad (2.5)$$

The latter refers to how well the target object covers the predicted shape. Equation 2.6 describes this loss in a similar manner to the coverage loss.

$$L_{cons}(P_m, O) = \sum_m \mathbb{E}_{\mathbf{p} \sim S(P_m)} \|\mathcal{C}_{obj}(\mathbf{p}'; O)\|^2 \quad (2.6)$$

In this case, we use the object distance field, which is $\mathcal{C}_{obj}(\mathbf{p}; O) = \min_{\mathbf{p}' \in O} \|\mathbf{p}' - \mathbf{p}\|^2$. During training, both of these losses weighted together as they essentially compete against each other: coverage favors maximal representation, and consistency favors compactness. A clear overview of the training process is shown in Figure 18.

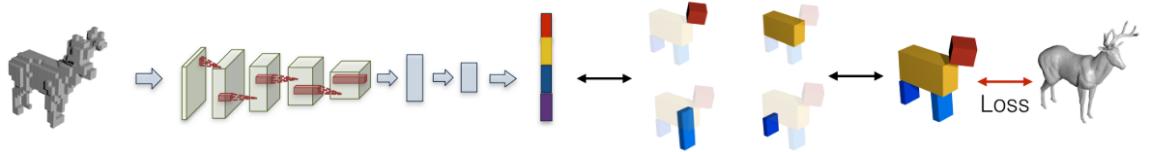


Figure 18: The input object is discretized using an occupancy grid and is fed into a 3D CNN. The predicted shapes are then compared to the original object using the coverage and consistency losses. ¹⁸

Superquadratics

While Tulsiani et al. (2017)'s method has seen great success in showcasing patterns in shape categories as well as shape manipulation, it still lacks in complexity as cuboids do not capture more complex shapes (e.g., curved objects, spheres). Paschalidou et al. (2019) circumvent this issue by considering a diverse shape vocabulary – superquadratics (Figure 19) – leading to more expressive scene abstractions. Figure 20 illustrates this difference.

¹⁸Image credit Tulsiani et al. (2017)

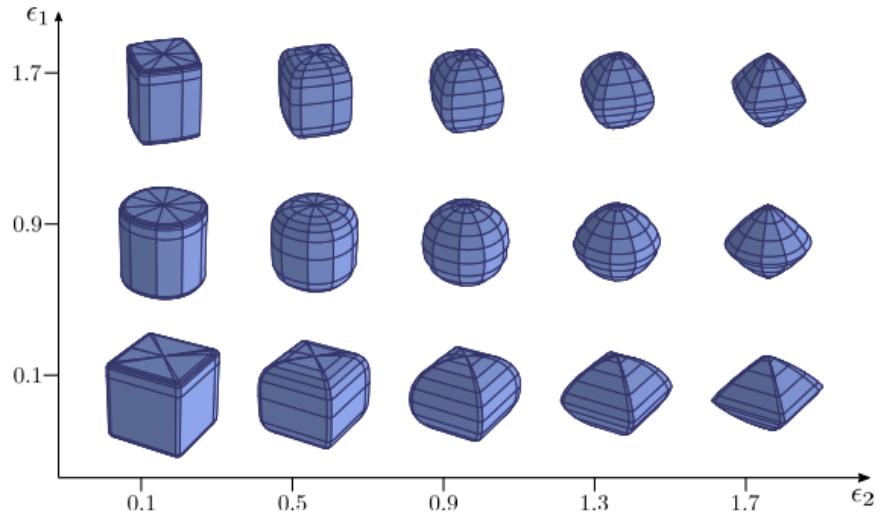


Figure 19: The superquadric shape vocabulary is more complex and can represent curvatures well. ¹⁹

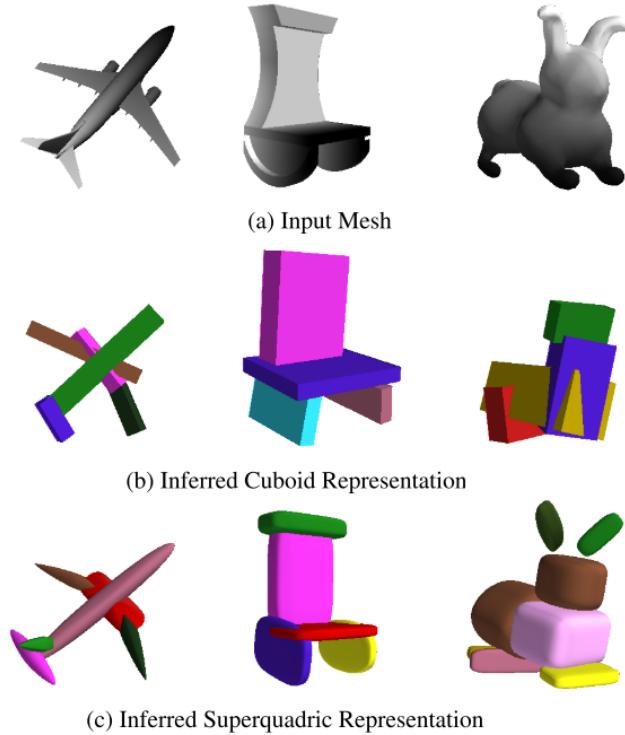


Figure 20: Tulsiani et al. (2017)'s cuboid method compared to Paschalidou et al. (2019)'s superquadric method. ²⁰

¹⁹Image credit Paschalidou et al. (2019)

Paschalidou et al. (2019) borrow the same CNN architecture used in Tulsiani et al. (2017), except that the prediction layers that output M cuboids now output M superquadrics. We explain the superquadric representation below:

REPRESENTING SUPERQUADRICS: A superquadric is parametrized by shape parameters $(\alpha, \epsilon, \eta, \omega)$ and the usual pose parameters (\mathbf{q}, \mathbf{t}) that define the rotation and translation. Formally, the surface vector \mathbf{r} of a superquadric is given by Equation 2.7. ϵ values are bound between $[0.1, 1.9]$ to prevent non-convex shapes, as the authors argue the rarity of occurrence of such shapes.

$$\mathbf{r}(\eta, \omega) = \begin{bmatrix} \alpha_1 \cos^{\epsilon_1} \eta \cos^{\epsilon_2} \omega \\ \alpha_2 \cos^{\epsilon_1} \eta \sin^{\epsilon_2} \omega \\ \alpha_3 \sin^{\epsilon_1} \eta \end{bmatrix} \quad (2.7)$$

POINT SAMPLING: Defining a distance field for every possible superquadric surface is difficult. The authors resort to a point sampling strategy to compute the losses defined below. Concretely, the network samples N points on the target object O , and K points on the surface of each predicted superquadric S_m . The authors refer to the superquadric sampling technique presented in Pilu and Fisher (1995) that selects η and ω in a way that allows for uniform point sampling. We denote the sampled object points by \mathbf{x}_i , and the sampled superquadric points on the m th primitive by \mathbf{y}_j^m . All following loss computations are done in the local space of a given superquadric. We thus denote by \mathbf{x}_i^m the transformed \mathbf{x}_i point in the m th primitive's space.

RECONSTRUCTION LOSS: Much like in Tulsiani et al. (2017), Paschalidou et al. (2019) employ a bi-directional reconstruction loss that measures coverage and consis-

²⁰Image credit Paschalidou et al. (2019)

tency. Given the sampled points, the network computes a coverage loss by computing the closest point on the target object for every sampled superquadric surface point. The resulting coverage loss is described by Equation 2.8.

$$L_{cov}(S_m, O) = \frac{1}{K} \sum_{k=1}^K \min_{i=1,\dots,N} \|\mathbf{x}_i^m - \mathbf{y}_k^m\|^2 \quad (2.8)$$

Similarly, the network computes a consistency loss by computing the closest superquadric surface point for every sampled object point. This is described by Equation 2.9.

$$L_{cons}(S_m, O) = \sum_{\mathbf{x}_i} \min_m \left[\min_{k=1,\dots,K} \|\mathbf{x}_i^m - \mathbf{y}_k^m\|^2 \right] \quad (2.9)$$

We note that the loss functions look very similar, except that the former minimizes the distance over all object points, while the latter minimizes the distance over all superquadric points. During training, both losses are weighted with 1.2 and 0.8 respectively, leading to good empirical results showcased by Figure 20.

3. Proposed Methodology

Ultimately, our work aims to combine recent successes in 3D object pose estimation with insights from object decomposition methods such as Paschalidou et al. (2019)’s superquadric decomposition. We thus propose the following method for 3D object pose estimation. Our approach is similar in the work described in Mahendran et al. (2018), wherein the pose is directly regressed from a single RGB image. However, we require additional pre-processing steps, namely the availability of a decomposition of the object into parts. Additionally, to focus the scope of this work, we have opted to only consider instance-based object pose estimation.

3.1. Data Generation

Our method predicts the pose of an object given an RGB image. We require pose supervision data to be available during training. This section outlines our data generation pipeline used to (1) construct a dataset of input RGB images and (2) associate pose labels to each image.

3.1.1. *Object-level Data*

We use data from the publicly available ShapeNet repository, which is a 3D CAD model dataset organized in categories. Our work uses 3D CAD models from ShapeNet’s chair category to conduct experiments. Since we focus on instance-based pose prediction, each trained model is only applicable to its associated chair instance. Samples from the chair category are shown in Figure 21.

To generate RGB image data, we first load the model in a 3D model editor. We then center a sphere around the center of the model, fix the virtual camera on the sphere’s surface uniformly at random, and render images of fixed dimensions from that angle. This directly gives us the associated object pose, since we know the camera’s relative



Figure 21: Samples of 3D CAD models within the chair category of the ShapeNet dataset.



Figure 22: Samples of renderings of a specific instance from the chair category of ShapeNet. Each image has an associated object pose inferred from the camera's position at rendering time.

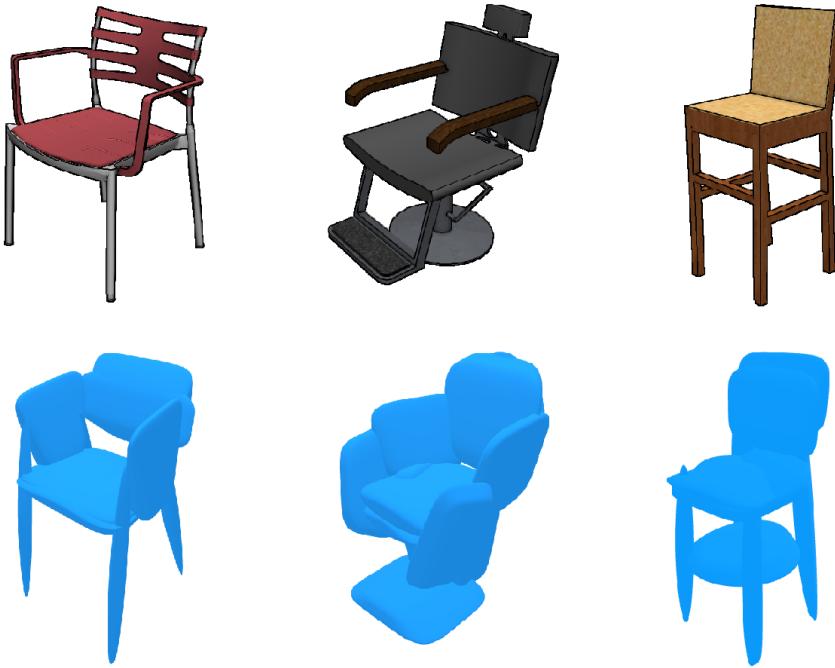


Figure 23: Images of the three chairs used in experiments. Upper row is the original 3D model, bottom row is the corresponding superquadric decomposition. The instances were picked with the intent of varying the shape and topology of the chairs. From left to right: instance #1, instance #2, instance #3.

position to the target object. Examples of generated data are shown in Figure 22.

As will be elaborated in Section 4, we generated data for three instances visualized in Figure 23. Concretely, we generated a total of 30,000 images: 10,000 per instance, with the a 70-30% train-test split.

3.1.2. Parts-level Data

Now that we have RGB images with associated object poses, we further decompose the object into M volumetric primitives and retrieve each of their poses. The goal is to obtain $M + 1$ pose labels for every RGB input image: one label for the object pose, and M labels for the object’s parts. As will be described in Section 3.2, the aim is to first predict M parts poses, and use these predictions to get a combined object pose prediction.



Figure 24: On the left is the superquadric decomposition of the original 3D model displayed on the right. Each of the 11 parts is uniquely colored.

We use Paschalidou et al. (2019)' superquadric decomposition to decompose the chair instance 3D CAD model into $M = 11$ primitives as shown in Figure 24. As previously explained, the result of the decomposition comes with the local pose of each primitive. Finally, we apply the object pose obtained from the previous section to transform the primitives from their local pose space to the object pose space.

3.2. Model Training

3.2.1. Architecture

Much like in Mahendran et al. (2018), the network architecture we used is composed of a feature network and a regressor network. The feature network is a pre-trained ResNet-50 by He et al. (2016) that extracts image features from the RGB input images. The regressor network takes as input the image features and first predicts M poses, corresponding to the parts of the object of interest. These M poses are then fed into an object pose retrieval module (i.e., a multi-layer perceptron) that infers the overall object pose given the predicted parts poses. Using this sequential prediction structure forces the network to *first* predict good M object poses in order to *then* obtain a good overall pose prediction. The architecture is shown in Figure 25.

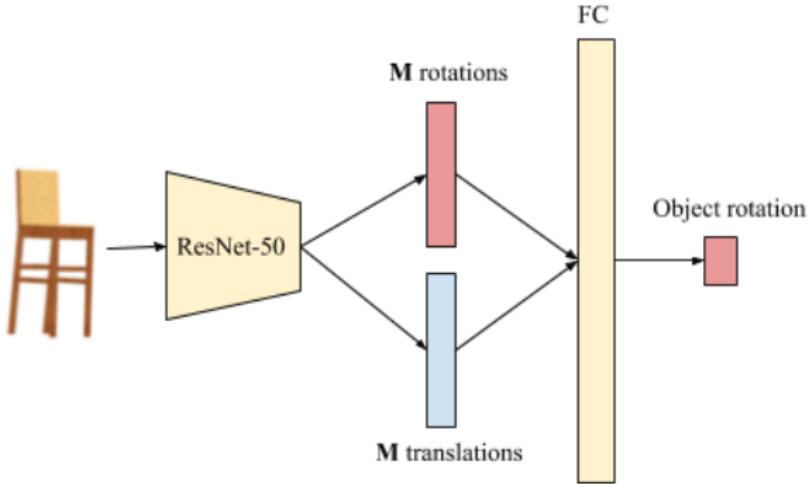


Figure 25: The RGB input image is passed through a ResNet-50 from which image features are extracted. The features are individually used to regress M rotations (red) and M translations (blue), which is one pose per object part. The M part poses are then passed through a fully-connected (FC) layer, which then outputs a single object rotation (red).

3.2.2. Loss Functions

Since the objective of our work is to compare parts-based regression against non-parts-based pose regression, we decided to use a simple L2 distance loss (i.e., Euclidean metric) to compute losses for both rotations and translations. Our intuition is that as long as we compare the models consistently, then using a Riemannian distance metric or a Euclidean distance metric should not be impactful for the comparison’s sake.

Using the L2 distance described by Equation 2.2, we compute M losses, one for every part. We then average these M losses to obtain a parts loss L_{parts} . We do the same operation for the overall object pose to get L_{obj} . The final (differentiable) loss value is $L_{parts} + L_{obj}$.

3.3. Model Evaluation

The collected images and their associated poses have been split into train and test sets. Evaluation is done solely on the test set. We used a standard pose evaluation metric called the Average Distance (ADD) shown in Equation 3.1. Formally, given the ground-truth rotation \mathbf{R} and translation \mathbf{T} , and the predicted rotation $\hat{\mathbf{R}}$ and translation $\hat{\mathbf{T}}$, the ADD computes the average pairwise distance between points on the 3D model \mathcal{M} transformed by the ground-truth pose (\mathbf{R}, \mathbf{T}) and points on the 3D model transformed by the predicted pose $(\hat{\mathbf{R}}, \hat{\mathbf{T}})$. Intuitively, ADD measures how well the predicted rotated 3D model is overlaid on top of the ground-truth rotated model. Thus, this metric measures the accuracy of our object pose prediction.

$$\text{ADD} = \frac{1}{|\mathcal{M}|} \sum_{\mathbf{x} \in \mathcal{M}} \|(\mathbf{Rx} + \mathbf{T}) - (\hat{\mathbf{R}}\mathbf{x} + \hat{\mathbf{T}})\| \quad (3.1)$$

4. Experiments

We first present results for the baseline method that uses direct pose regression. We then show our method’s results: parts-based pose regression. We follow by a discussion comparing both methodologies by contrasting their Average Distance (ADD) curves. Finally, we show results obtained by training the models using a rotation representation different from quaternions: 6D rotation representation.

4.1. Baseline: Direct Pose Regression

For direct regression, we follow the methodology of Mahendran et al. (2018) except that we do not use an intermediate coarse classifier. This is to maintain consistency for the sake of comparison. That is, the baseline network is a ResNet-50 passed to a multi-layer perceptron that predicts the overall object pose.

The network is trained for 100 epochs with a learning rate $\mu = 0.001$. We plot the L2 loss after each epoch, as shown in Figure 26. Every 10 epochs, we evaluate the model’s performance on the entire test set using the ADD metric. The plot in Figure 27 showcases this performance evaluated on each of the three chair instances.



Figure 26: Loss curves from training the baseline model on the first chair instance.

Note that this performance need not be monotonically decreasing as the model trains more. The model may overfit on the training data, and as such starts to generalize worse on the test set. To illustrate, the best model for instance #2 is the one trained until epoch 80 (lowest ADD), while the best models for the other two instances are those trained until epoch 100.

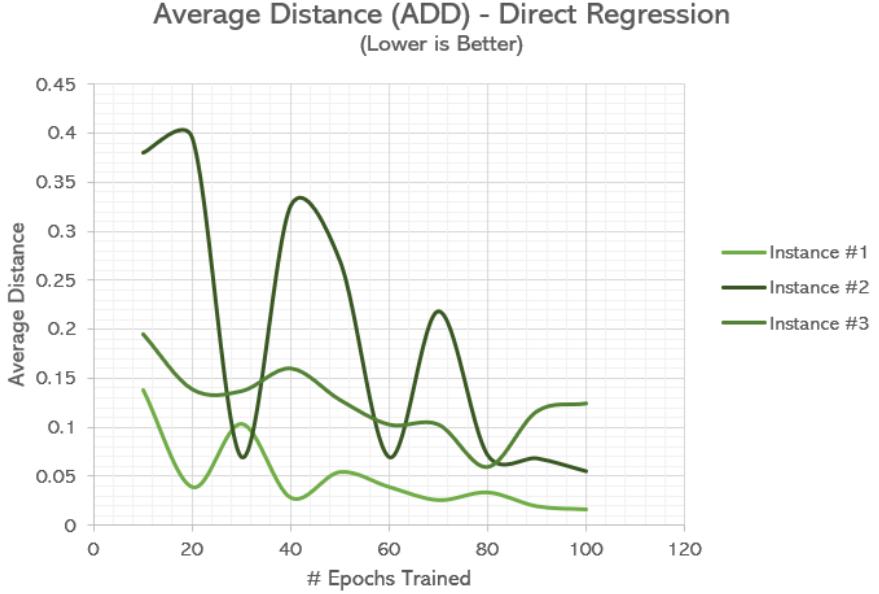


Figure 27: Average Distance (ADD) curves obtained using the baseline method. Each curve represents a single model trained on a specific chair instance.

4.2. Our Method: Parts-Based Pose Regression

Similar to the baseline model, we train our network for 100 epochs with a learning rate $\mu = 0.001$. The L2 loss plots are shown in Figure 28. The performance evaluation is shown in Figure 29. We note that like the baseline, our method’s performance varies in performance depending on the chair instance. However, in all three cases, the ADD curves exhibit a ”valley” shape, suggesting some level of stability during training.

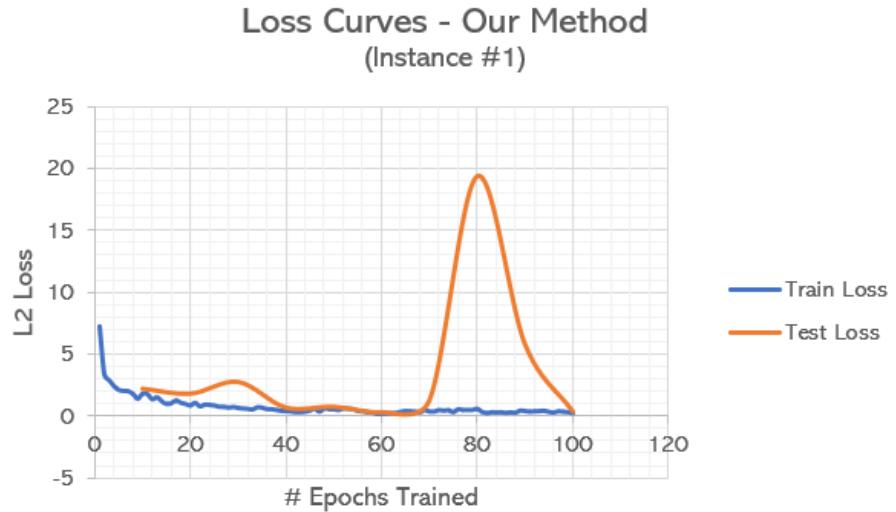


Figure 28: Loss curves from training a model according to our method on the first chair instance.

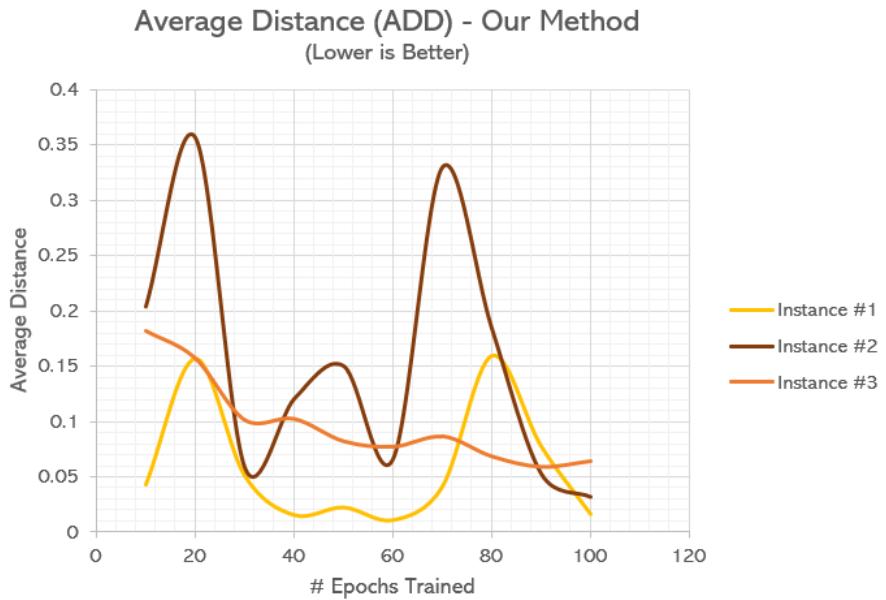


Figure 29: Average Distance (ADD) curves obtained using our method. Each curve represents a single model trained on a specific chair instance.

4.3. Comparison

From the ADD Figures 27 and 29, we notice that models trained using our methods exhibit smoother performance compared to the baseline. For example, the baseline seems to overfit quite often, as it has trouble finding a stable and generalizing model between epochs 30 and 60 for instance #2, while our method has less bumpy ADD values in the same range.

Figure 30 showcases ADD comparisons directly. Each figure compares the ADD values for a specific instance using a model trained with the baseline method and another model trained with our methodology. In all cases, our method outperforms the baseline. Table 1 summarizes these results by showing the minimum and average ADD values per instance.

Chair Instance	Instance #1		Instance #2		Instance #3	
Average Distance	Best	Average	Best	Average	Best	Average
Baseline (quat)	0.0155	0.0490	0.0557	0.1926	0.0596	0.1267
Our Method (quat)	0.0108	0.0593	0.0319	0.1555	0.0592	0.0979

Table 1: Average Distance (ADD) measurements for the trained models using quaternion rotation. Bolded numbers represent the better measurement in the respective column. The title "best" refers to the minimum ADD out of all epochs, and "average" denotes the mean ADD within all epochs.

We additionally provide visual comparison figures. For each chair instance, we rendered its corresponding 3D CAD model using the ground-truth pose in red, and contrasted the predicted poses by rendering an transparent model using the predictions. The visual results show that our method generally does a better job compared to the baseline. The renders for a specific instance are shown in Figure 31. The rest of the instances' renders are showcased in the Appendix, Figures 36 and 37.

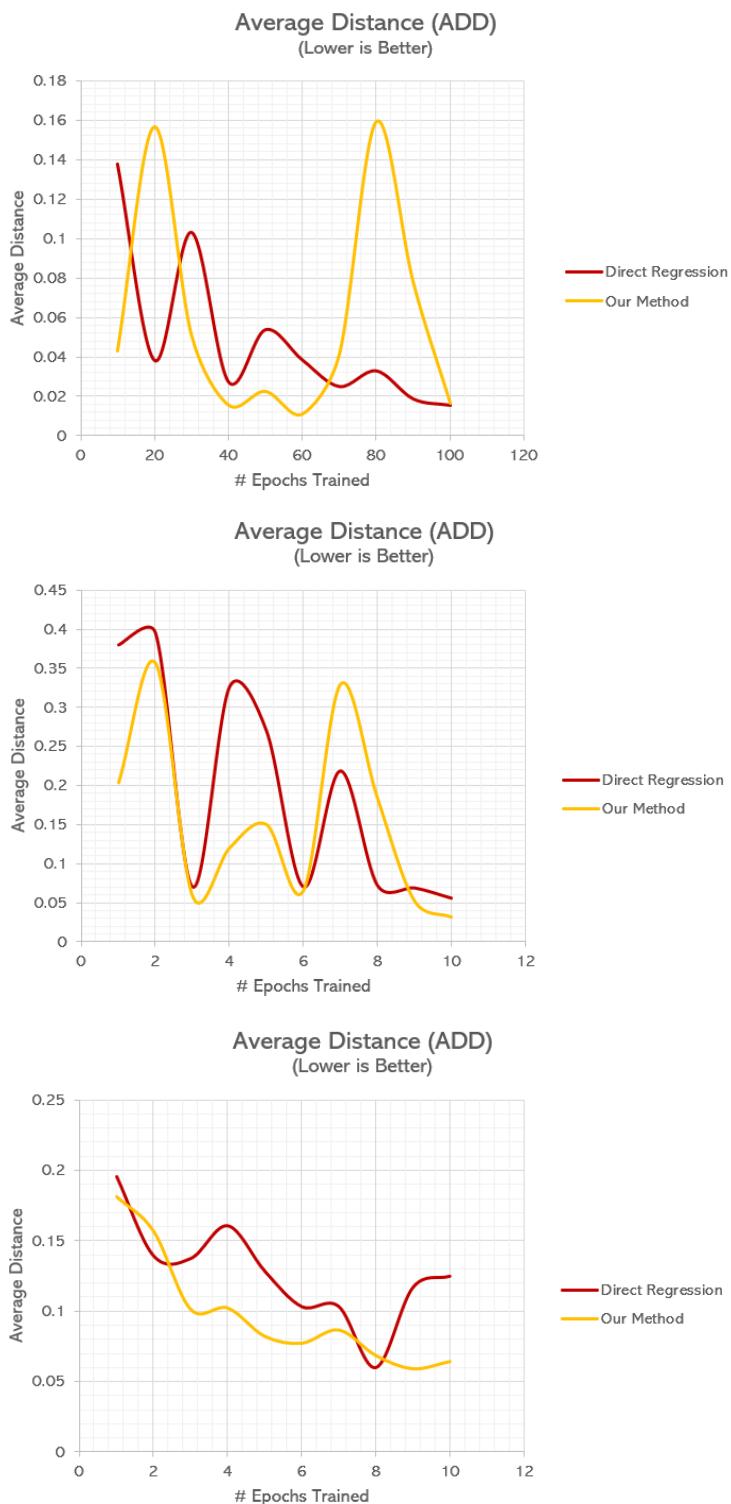


Figure 30: ADD curves comparing the baseline and our method. From top to bottom: instance #1, instance #2, instance #3.

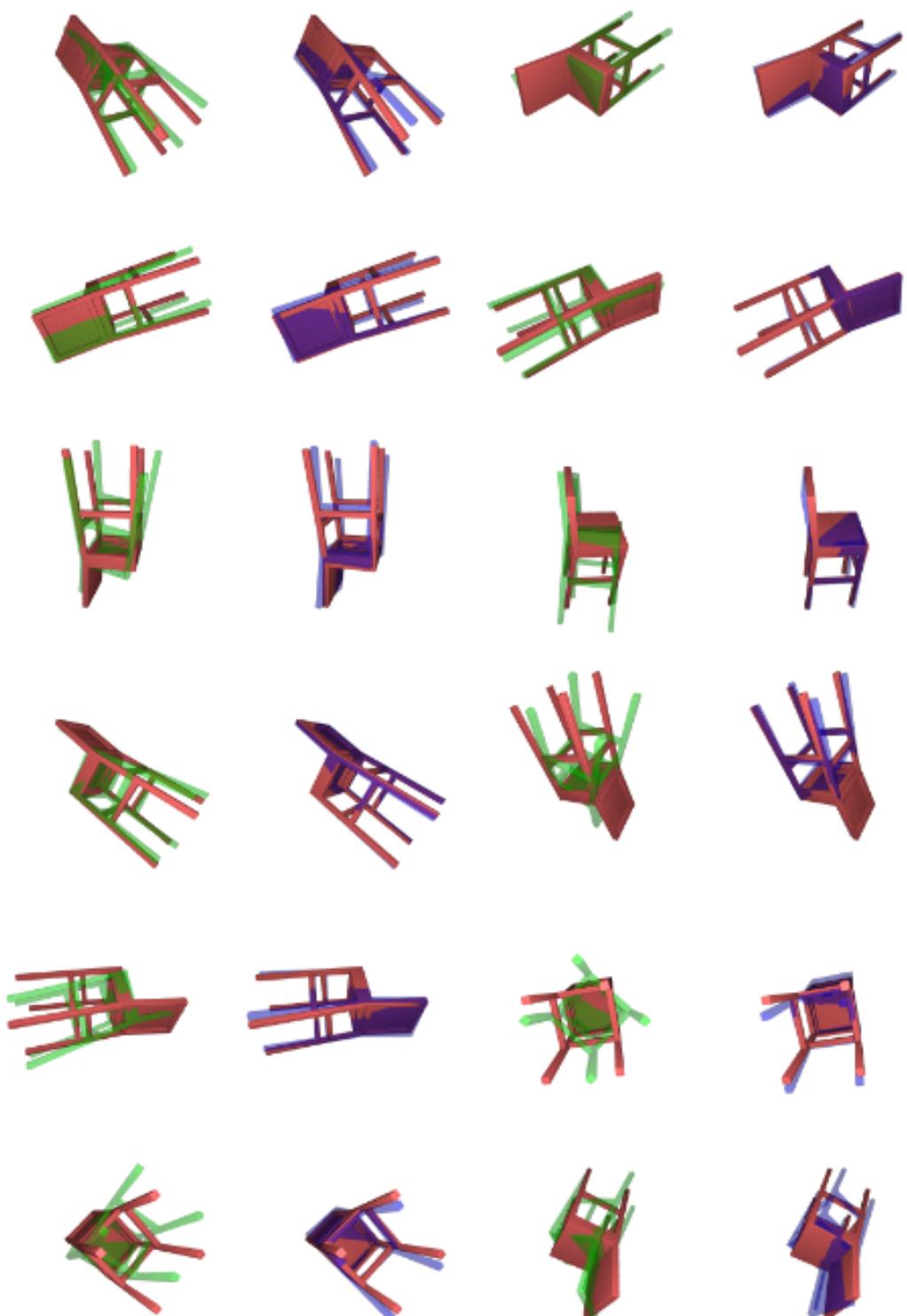


Figure 31: Chair instance #3 results. The true model is displayed in red. The baseline prediction is in green. Our model’s prediction is in blue.

4.4. Quaternions vs. 6D Rotation Representation

Zhou et al. (2019) demonstrate that the widely used 3D and 4D representations of rotations, such as Euler angles and quaternions, are discontinuous and are thus difficult for neural networks to learn. To circumvent this issue, they propose the usage of the 6D representation of rotation, which is proven to be continuous and thus easier to learn. We trained both the baseline network and our network using this 6D representation, and compared it to the models trained using quaternions. Figures 33 and 32 showcase the results on an instance-by-instance basis.

Chair Instance	Instance #1		Instance #2		Instance #3	
Average Distance	Best	Average	Best	Average	Best	Average
<i>Baseline (quat)</i>	0.0155	0.0490	0.0557	0.1926	0.0596	0.1267
<i>Baseline (6d)</i>	0.0134	0.0700	0.0480	0.2231	0.0759	0.1585
<i>Our Method (quat)</i>	0.0108	0.0593	0.0319	0.1555	0.0592	0.0979
<i>Our Method (6d)</i>	0.0083	0.0793	0.0340	0.2809	0.0563	0.0943

Table 2: Average Distance (ADD) measurements for the trained models using 6D rotation.

We observe that using 6D rotation seems to result in noisier ADD measurements in the majority of the cases. For example, both methods suffer from wavy ADD curves when using 6D rotation for the first chair instance. However, our method seems to be robust to this representation change when it comes to the third chair instance: the results are better, and the ADD curve remains smooth as shown in Figure 32. In contrast, direct regression has better but noisier results as shown in Figure 33. Table 2 provides a comprehensive summary of the results, which suggest that it is *inconclusive* whether using 6D rotation over quaternions is better for our purposes.

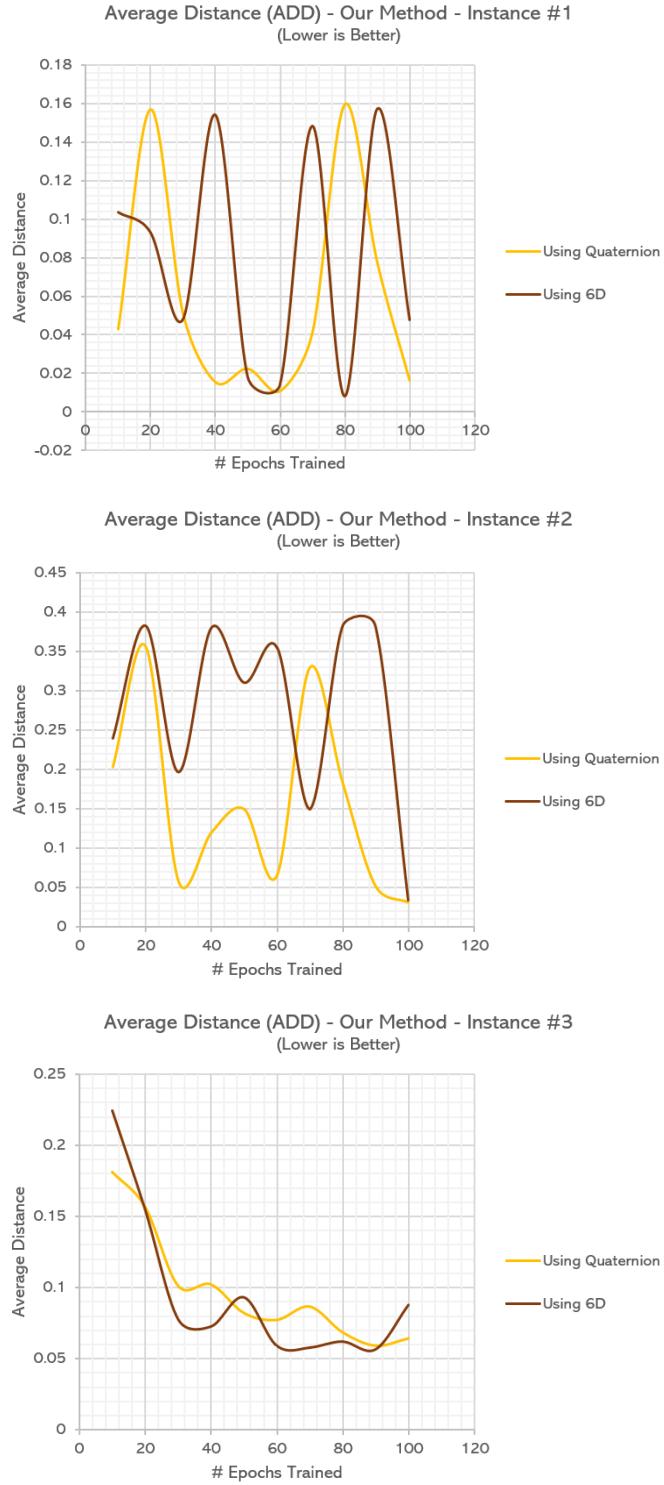


Figure 32: ADD curves comparing our method using quaternions and using 6D rotation. From top to bottom: instance #1, instance #2, instance #3.

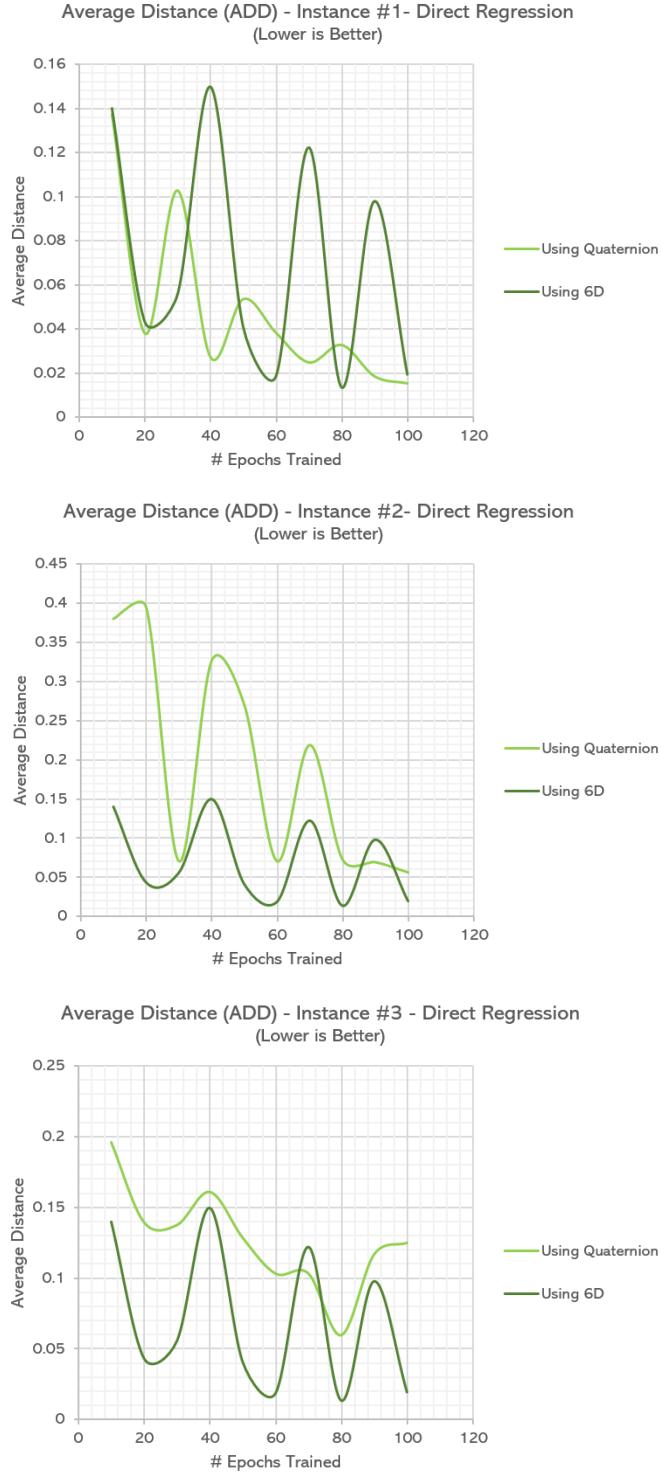


Figure 33: ADD curves comparing the baseline using quaternions and using 6D rotation. From top to bottom: instance #1, instance #2, instance #3.

5. Conclusion

In this work, we present a supervised 3D object pose estimation method inspired by regression-based methodologies. We take advantage of recent advancements in object decomposition and incorporate part-based pose estimation in the training pipeline. The overall result suggests that intermediately learning the pose of the parts allows for better learning of the overall pose of the object. As such, extracting the object pose from the parts' poses can be thought of like a pose refinement process.

While providing better results, the usage of the decomposed parts is still a costly process. Indeed, we assume the availability of a 3D model to learn the decomposition from, in addition to a larger set of pose labels resulting from the pose of the parts. As such, we would like to relax the assumptions made in this work in the future.

For example, attempting parts-based pose prediction using generalized parts learned from a single 3D model category (e.g., all chairs) may allow for category-based pose prediction. This may beneficially lead to only requiring one canonical 3D model per category. Additionally, resorting to an unsupervised superquadric reconstruction of the model from a single image may be a promising way to remove restrictive pose supervision.

APPENDIX

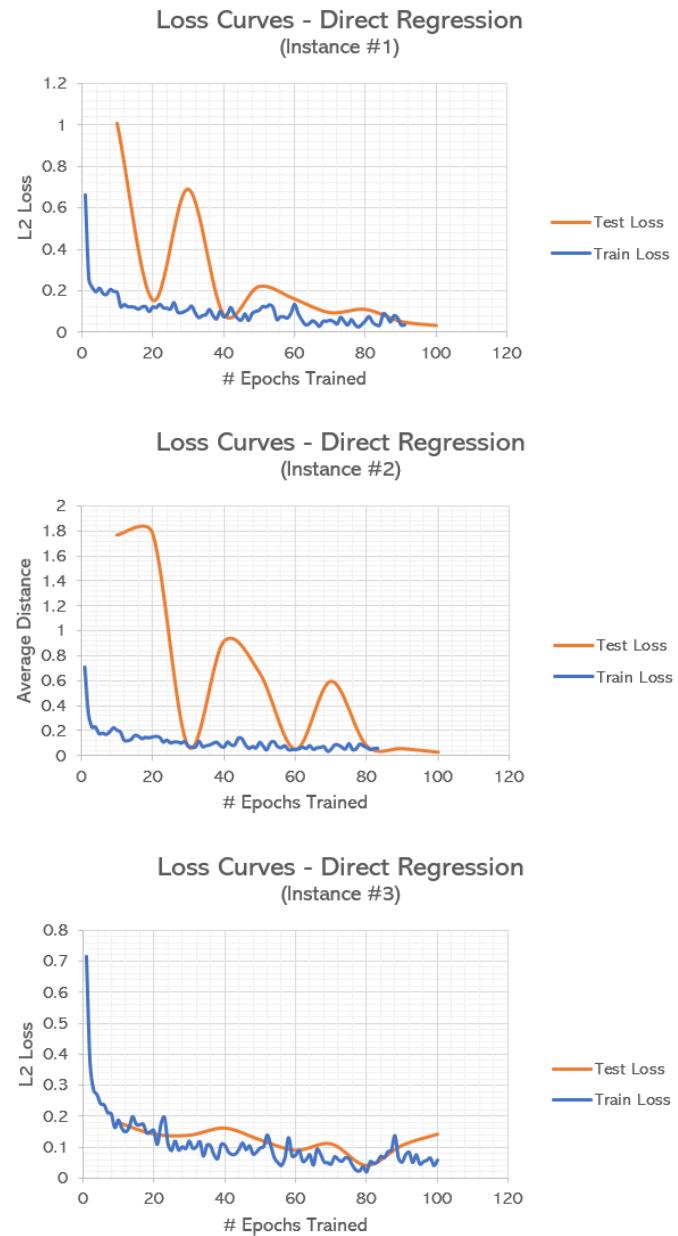


Figure 34: Loss curves from training the baseline model, one model per instance.

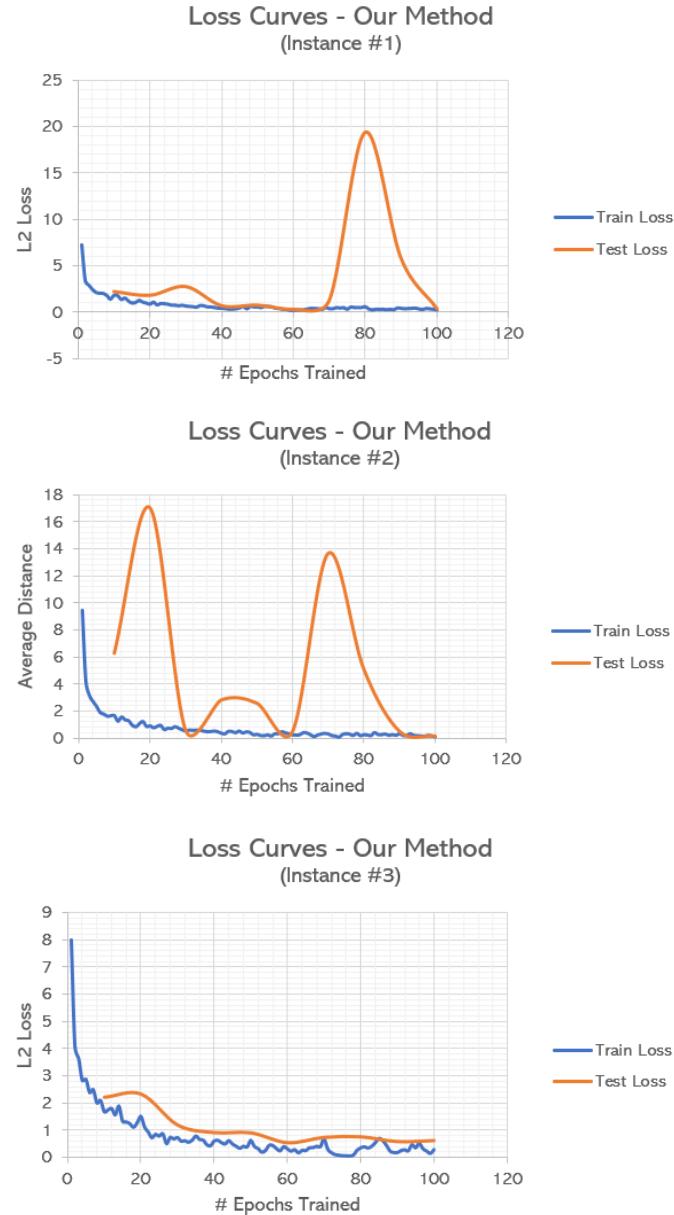


Figure 35: Loss curves from training models according to our method, one model per instance.

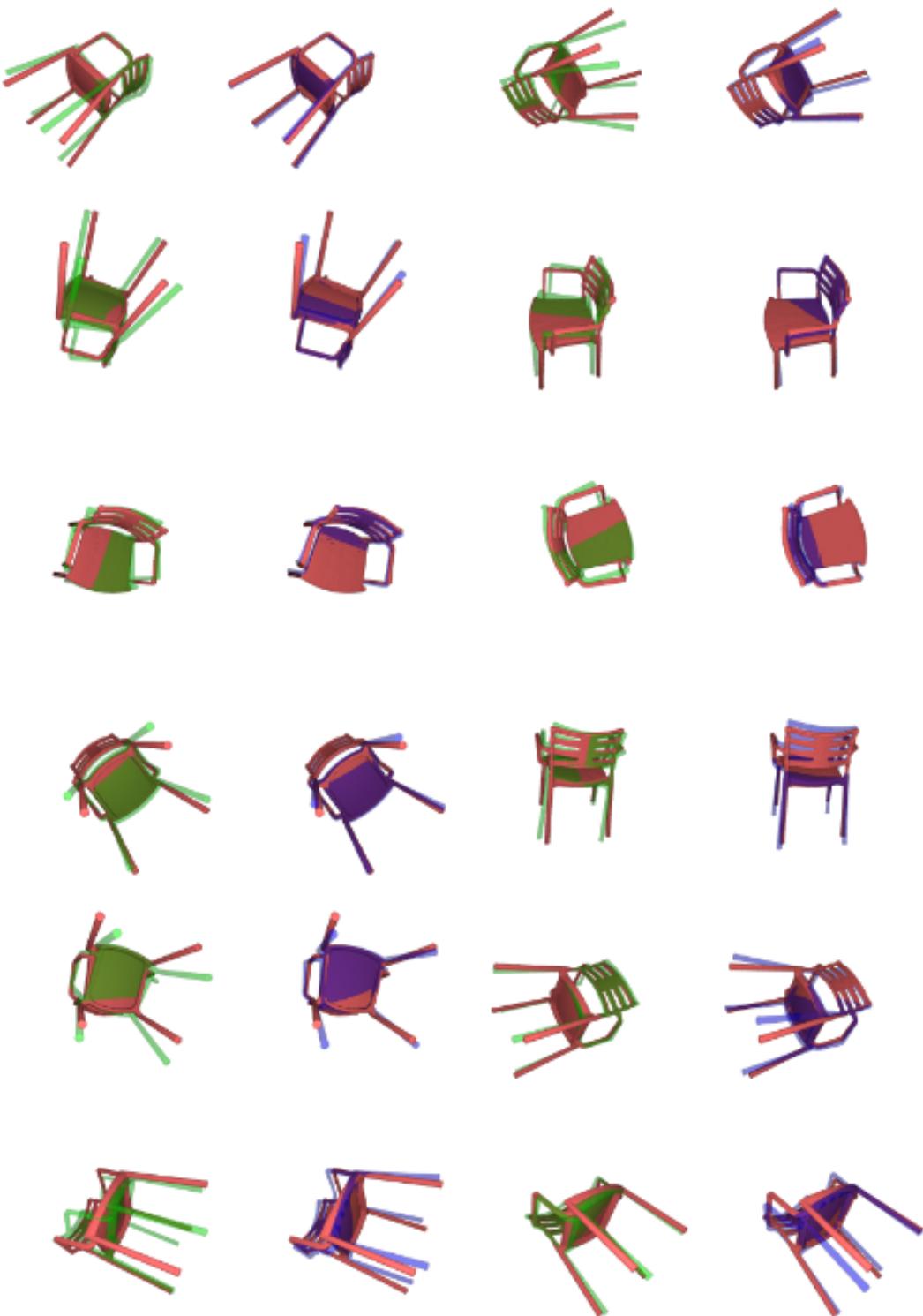


Figure 36: Chair instance #1 results. The true model is displayed in red. The baseline prediction is in green. Our model's prediction is in blue.

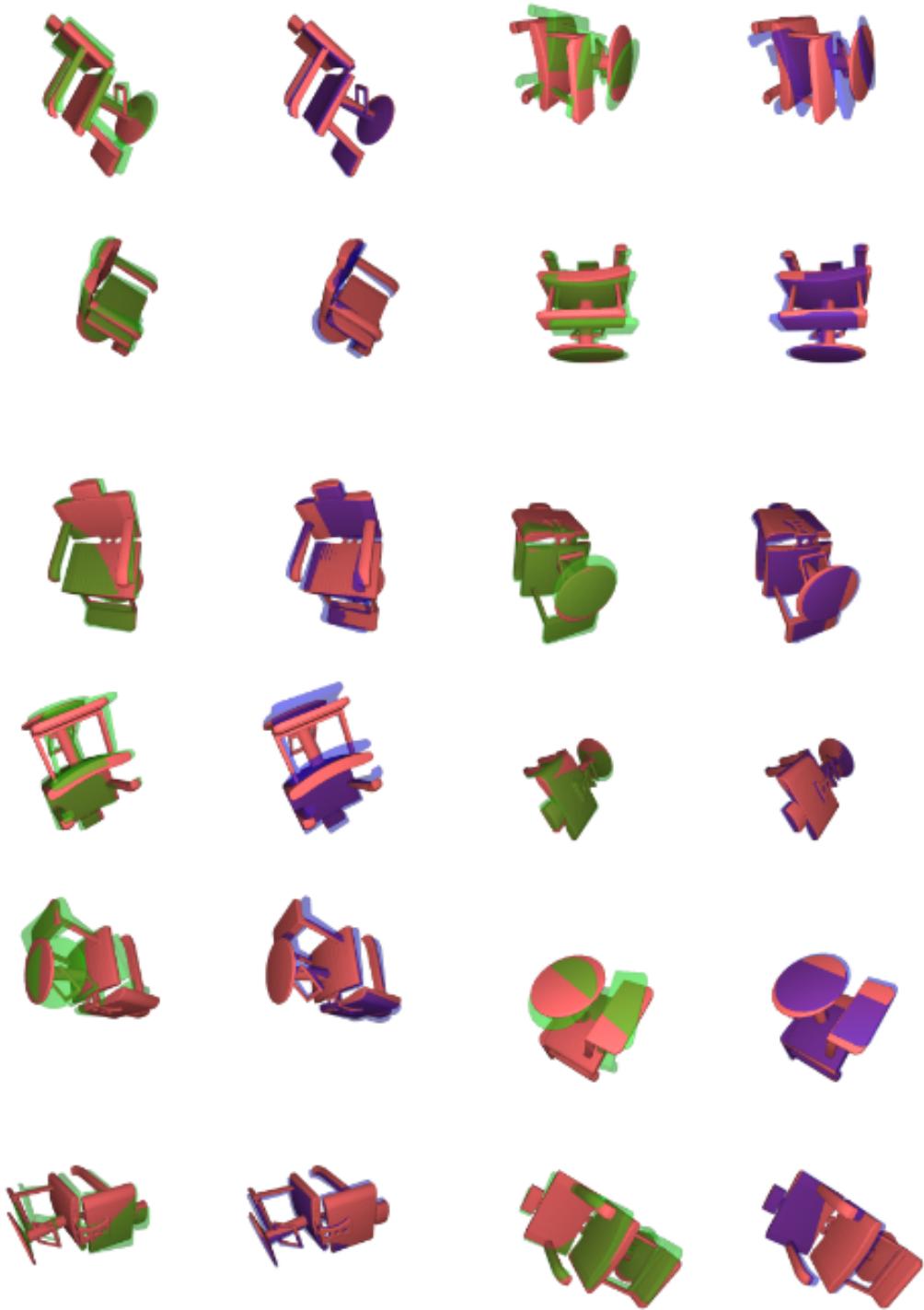


Figure 37: Chair instance #2 results. The true model is displayed in red. The baseline prediction is in green. Our model's prediction is in blue.

BIBLIOGRAPHY

- E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6D object pose estimation using 3D object coordinates. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8690 LNCS(PART 2):536–551, 2014. ISSN 16113349. doi: 10.1007/978-3-319-10605-2_35.
- Z. Cao, Y. Sheikh, and N. K. Banerjee. Real-time scalable 6DOF pose estimation for textureless objects. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:2441–2448, 2016. ISSN 10504729. doi: 10.1109/ICRA.2016.7487396.
- A. Collet and M. Martinez. MOPED: Object Recognition and Pose Estimation for Manipulation. URL <https://personalrobotics.ri.cmu.edu/projects/moped.php>.
- J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2188–2202, 2011. ISSN 01628828. doi: 10.1109/TPAMI.2011.70.
- S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8695 LNCS(PART 7):345–360, 2014. ISSN 16113349. doi: 10.1007/978-3-319-10584-0_23.
- K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:770–778, 2016. ISSN 10636919. doi: 10.1109/CVPR.2016.90.
- 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. *Proceedings of the IEEE International Conference on Computer Vision*, pages 858–865, 2011. doi: 10.1109/ICCV.2011.6126326.
- 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. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7724

LNCS(PART 1):548–562, 2013. ISSN 03029743. doi: 10.1007/978-3-642-37331-2_42.

- W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab. Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9907 LNCS:205–220, 2016. ISSN 16113349. doi: 10.1007/978-3-319-46487-9_13.
- V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate $O(n)$ solution to the PnP problem. *International Journal of Computer Vision*, 81(2):155–166, 2009. ISSN 09205691. doi: 10.1007/s11263-008-0152-6.
- J. J. Lim, A. Khosla, and A. Torralba. FPM: Fine pose parts-based model with 3D CAD models. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8694 LNCS (PART 6):478–493, 2014. ISSN 16113349. doi: 10.1007/978-3-319-10599-4_31.
- D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, pages 91–110, 2004. URL <https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf>.
- S. Mahendran, M. Y. Lu, H. Ali, and R. Vidal. Monocular Object Orientation Estimation using Riemannian Regression and Classification Networks. 2018. URL <http://arxiv.org/abs/1807.07226>.
- C. Niu, J. Li, and K. Xu. Im2Struct: Recovering 3D Shape Structure from a Single RGB Image. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4521–4529, 2018. ISSN 10636919. doi: 10.1109/CVPR.2018.00475.
- K. Park, A. Mousavian, Y. Xiang, and D. Fox. LatentFusion: End-to-End Differentiable Reconstruction and Rendering for Unseen Object Pose Estimation. 2019a. URL <http://arxiv.org/abs/1912.00416>.
- K. Park, T. Patten, and M. Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6D pose estimation. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:7667–7676, 2019b. ISSN 15505499. doi: 10.1109/ICCV.2019.00776.
- D. Paschalidou, A. O. Ulusoy, and A. Geiger. Superquadrics revisited: Learning 3D shape parsing beyond cuboids. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:10336–10345, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.01059.

- G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis. 6-DoF object pose from semantic keypoints. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2011–2018, 2017. ISSN 10504729. doi: 10.1109/ICRA.2017.7989233.
- M. Pilu and R. B. Fisher. Equal-distance sampling of superellipse models. pages 257–266, 1995.
- C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:77–85, 2017. doi: 10.1109/CVPR.2017.16.
- S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. ISSN 01628828. doi: 10.1109/TPAMI.2016.2577031.
- ShapeNet. Shapenet dataset. URL <https://www.shapenet.org/>.
- H. Su, C. R. Qi, Y. Li, and L. J. Guibas. Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3D model views. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter(Sec 2):2686–2694, 2015. ISSN 15505499. doi: 10.1109/ICCV.2015.308.
- M. Sundermeyer, Z. C. Marton, M. Durner, M. Brucker, and R. Triebel. Implicit 3D orientation learning for 6D object detection from RGB images. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11210 LNCS(2010):712–729, 2018. ISSN 16113349. doi: 10.1007/978-3-030-01231-1_43.
- A. Tejani, R. Kouskouridas, A. Doumanoglou, D. Tang, and T. K. Kim. Latent-Class Hough Forests for 6 DoF Object Pose Estimation. *IEEE transactions on pattern analysis and machine intelligence*, 40(1):119–132, 2018. ISSN 19393539. doi: 10.1109/TPAMI.2017.2665623.
- B. Tekin, S. N. Sinha, and P. Fua. Real-Time Seamless Single Shot 6D Object Pose Prediction. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 292–301, 2018. ISSN 10636919. doi: 10.1109/CVPR.2018.00038.
- S. Tulsiani, H. Su, L. J. Guibas, A. A. Efros, and J. Malik. Learning shape abstractions by assembling volumetric primitives. *Proceedings - 30th IEEE Conference on*

Computer Vision and Pattern Recognition, CVPR 2017, 2017-Janua:1466–1474, 2017. doi: 10.1109/CVPR.2017.160.

- 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. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:3338–3347, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.00346.
- P. Wohlhart and V. Lepetit. Learning descriptors for object recognition and 3D pose estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June(1):3109–3118, 2015. ISSN 10636919. doi: 10.1109/CVPR.2015.7298930.
- Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. 2018. doi: 10.15607/rss.2018.xiv.019.
- Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:5738–5746, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.00589.
- M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis. Single image 3D object detection and pose estimation for grasping. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3936–3943, 2014. ISSN 10504729. doi: 10.1109/ICRA.2014.6907430.