Towards Learning Geometric Eigen-Lengths Crucial for Fitting Tasks

Yijia Weng [1]   **Kaichun Mo** [1 2]   **Ruoxi Shi** [3]   **Yanchao Yang** [1 4]   **Leonidas Guibas** [1]

Abstract

Some extremely low-dimensional yet crucial geometric eigen-lengths often determine the success of some geometric tasks. For example, the *height* of an object is important to measure to check if it can fit between the shelves of a cabinet, while the *width* of a couch is crucial when trying to move it through a doorway. Humans have materialized such crucial geometric eigen-lengths in common sense since they are very useful in serving as succinct yet effective, highly interpretable, and universal object representations. However, it remains obscure and underexplored if learning systems can be equipped with similar capabilities of automatically discovering such key geometric quantities from doing tasks. In this work, we therefore for the first time formulate and propose a novel learning problem on this question and set up a benchmark suite including tasks, data, and evaluation metrics for studying the problem. We focus on a family of common fitting tasks as the testbed for the proposed learning problem. We explore potential solutions and demonstrate the feasibility of learning eigen-lengths from simply observing successful and failed fitting trials. We also attempt geometric grounding for more accurate eigen-length measurement and study the reusability of the learned eigen-lengths across multiple tasks. Our work marks the first exploratory step toward learning crucial geometric eigen-lengths and we hope it can inspire future research in tackling this important yet underexplored problem. Project page: https://yijiaweng.github.io/geo-eigen-length.

1. Introduction

Consider a robot tasked with placing many small objects on warehouse shelves, where both the objects and the shelves have diverse geometric configurations. While the robot can simply try to accomplish the task by trial and error, to us as humans, it is clear that certain placements should not be attempted because they will obviously fail. For example, we should not attempt to place a tall object on a shelf whose height is too low. We base this judgment on the estimation of a critical geometric eigen-length or measurement, the *height* of the object and the shelf, whose comparison allows a quick estimate of task feasibility. Such scalar measurements are crucial for downstream tasks. And we call them "eigen" because they are intrinsic properties of the object and usually act as very low-dimensional geometric summaries with respect to many tasks, invariant to the environment it interacts with. For example, to determine whether M objects can be placed on N different shelves, once the height of each object/shelf is extracted, we can compose and compare them arbitrarily without having to exhaustively analyze $N \times M$ pairs of raw geometries.

While object *height* is an example of important eigen-lengths that are crucial for the above shelf placement task, it is not hard to think of many other types of object eigen-lengths for other geometric tasks. Figure 1 presents some example tasks together with the presumed geometric eigen-lengths based on human common sense. For example, the geometric eigen-length *diameter* is important for the task of stacking plates in different sizes (Figure 1, (a)), while the *width* and *length* of an object are crucial geometric eigen-lengths for deciding if one can put it into an open box (Figure 1, (c)).

Having such extremely low-dimensional yet crucial geometric eigen-lengths extracted as the representations for objects is certainly beneficial for designing learning systems aimed at artificial general intelligence. One telling evidence is that we humans have naturally built up the vocabulary of geometric key quantities, such as *height, width,* and *diameter*, when perceiving and modeling everyday objects, and used them to perform various tasks. Besides being succinct yet effective abstractions of objects for quickly estimating the feasibility for the downstream tasks, such crucial geometric eigen-lengths are also highly *interpretable*, which exposes the principled reasoning process behind the feasibility checking, and *universal*, as they are generally applicable to objects with arbitrary shapes and useful across different downstream tasks.

[1]Stanford University [2]NVIDIA Research [3]Shanghai Jiaotong University [4]HKU. Correspondence to: Yijia Weng <yijiaw@stanford.edu>.
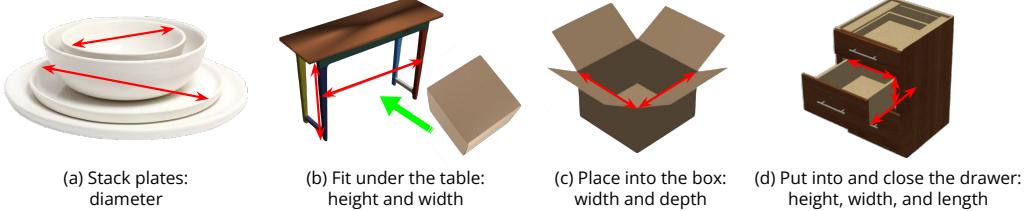
Figure 1. Example tasks and the hypothesized crucial geometric measurements by humans.

Current research in representation learning for computer vision and robotics has mostly been focusing on learning high-dimensional latent codes or heavily injecting human knowledge as inductive biases for learning structured representations. While learning high-dimensional latent codes provides total flexibility in learning any useful feature for mastering the downstream tasks, these latent codes are hard to interpret and may be prone to overfitting the training domain. For structured representations, though researchers have explored using different kinds of object representations, such as bounding boxes (Tulsiani et al., 2017) and key points (Manuelli et al., 2019), to accomplish various downstream tasks in computer vision and robotics, these structure priors are manually specified based on human knowledge about the tasks. In contrast, we aim to explore the automatic discovery of low-dimensional yet crucial geometric quantities for downstream tasks while injecting the minimal human prior knowledge – only assuming that we are measuring some 1D lengths of the input objects.

In this paper, we first propose to study a novel learning problem on discovering low-dimensional geometric eigen-lengths crucial for downstream tasks and set up the benchmark suite for studying the problem. Specifically, we target a family of fitting tasks where the goal is to find a placement/trajectory for an object in an environment, subject to geometric and semantic constraints, e.g. no collision. As illustrated in Figure 2, given a fitting task (putting the bowl inside the drawer of the table) that involves an environment geometry (the table) and an object shape (the bowl), we are interested in predicting whether the object can fit in the scene accomplishing the task or not, via discovering a few crucial geometric eigen-lengths and composing them with a task program which outputs the final task feasibility prediction. To study the problem, we also define a set of commonly seen fitting tasks, generate large-scale data for the training and evaluation on each task, and set up a set of quantitative and qualitative metrics for evaluating and analyzing the method performance and if the emergent geometric eigen-lengths match the desired ones humans usually use.

We also explore potential solutions to the proposed learning problem and present several of our key findings. First of all, we will show that learning such low-dimensional key geo-metric eigen-lengths are achievable from only using weak supervision signals such as the success or failure of training fitting trials. Secondly, the learned crucial geometric eigen-lengths can be more accurately measured if geometric grounding is allowed and attainable for certain fitting tasks. Finally, we make an initial stab at exploring how to share and re-use the learned geometric eigen-lengths across different tasks and even for novel tasks. Marking the first step in defining and approaching this important yet underexplored problem, we hope our work can draw people's attention to this direction and inspire future research.

To summarize, this work makes the following contributions:

- We propose a novel learning problem on discovering low-dimensional geometric eigen-lengths crucial for fitting tasks;
- We set up a benchmark suite for studying the problem, including a set of fitting tasks, the dataset for each task, and a range of quantitative and qualitative metrics for thorough performance evaluation and analysis;
- We explore potential solutions to the proposed learning problem and present some key take-away messages summarizing both the successes and unresolved challenges.

2. Related Work

Learning Geometry Abstraction. A long line of research has focused on learning low-dimensional and compact abstraction for input geometry. Given as input a 2D or 3D shape, past works have studied learning various geometric abstraction as the shape representation, such as bounding boxes (Tulsiani et al., 2017; Sun et al., 2019), convex shapes (Deng et al., 2020), Gaussian mixtures (Genova et al., 2019; 2020), superquadrics (Paschalidou et al., 2019; 2020), parametric curves (Reddy et al., 2021) and surfaces (Sharma et al., 2020; Smirnov et al., 2020), *etc.*. Most of these works use geometry fitting as the primary objective. Our work, however, focuses on discovering geometric abstraction that can help solve the downstream manipulation tasks instead of reconstruction.

There are also previous works exploring ways to learn task-specific geometry representation for manipulation tasks. For example, researchers have tried to learn key points (Manuelli
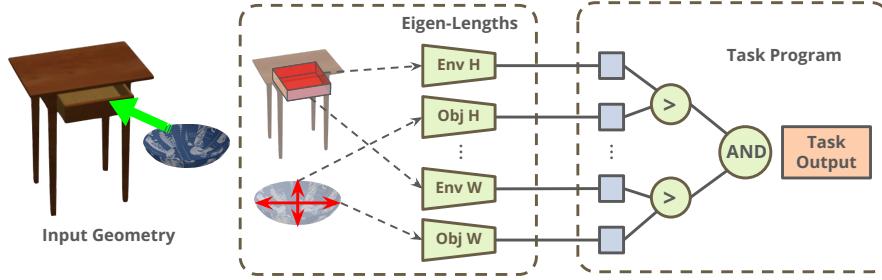
Figure 2. **Proposed Learning Paradigm** where we first predict a set of geometric eigen-lengths from the input geometries, then compose them using a task program to get the final task output.

et al., 2019; Qin et al., 2020; Wang et al., 2020; Chen et al., 2020; Jakab et al., 2021; Chen et al., 2021) and affordance information (Kim & Sukhatme, 2014; Mo et al., 2021a;b; Turpin et al., 2021; Deng et al., 2021) for robotic manipulation tasks. These works mostly pre-define the types of geometry abstraction and the downstream policies to use the extracted shape summaries, and the abstraction is mostly dense or high dimensional. In this paper, we aim for useful geometric eigen-lengths and ways to automatically discover and compose them for solving manipulation tasks.

Disentangled Visual Representation Learning. Another line of work focuses on unsupervised representation learning techniques that pursue disentangled and compositional latent representations for visual concepts. For example, InfoGAN (Chen et al., 2016), beta-VAE (Higgins et al., 2017), and many more works (Higgins et al., 2016; Siddharth et al., 2017; Yang et al., 2020) discover disentangled features, each of which controls a certain aspect of visual attributes, usually with reconstruction as the objective. In contrast to their primary objectives of controllable reconstruction or generation, we explore the problem of learning geometric eigen-lengths driven by the goal of accomplishing downstream fitting tasks. Also, our task involves reasoning over two geometric inputs and comparing the extracted eigen-lengths on both inputs, while these previous works on disentangled visual representation learning factor out visual attributes for a single input datum.

3. Learning Problem Formulation

Given a fitting task $T \in \mathcal{T}$, we aim to learn very few but the crucial geometric eigen-lengths $\mathcal{L}_T$ (*e.g.*, *width, length, height*) of the object shape $O \in \mathcal{O}$ and the environment geometry $E \in \mathcal{E}$ that are useful for checking the feasibility of fitting O into E under the task T. Figure 2 presents an example of the proposed learning problem where the task is to put the bowl (O) inside the drawer of the cabinet (E). In this example, the *width, length, height* of the drawer and the bowl are the crucial desired geometric eigen-lengths ($\mathcal{L}_T$) and we can compose them in a task program to output the final task feasibility prediction.

We define each eigen-length $L \in \mathcal{L}_T$ as a function mapping from the input object shape O or the environment geometry E to a scalar value, which is the eigen-length measurement, *i.e.* $L : \mathcal{O} \cup \mathcal{E} \to \mathbb{R}$. After obtaining the eigen-length measurements for both the object and environment inputs, *i.e.* $\{L(O)|L \in \mathcal{L}_T\}$ and $\{L(E)|L \in \mathcal{L}_T\}$, we perform pairwise comparisons between the corresponding eigen-lengths checking if $L(O) < L(E)$ holds for every $L \in \mathcal{L}_T$. The task of fitting O in E is predicted as successful if all the conditions hold and as failed if any condition does not hold. This format of task program is based on the intuition that in fitting tasks, we require the object to be "smaller" than the parts of the environment affording the action.

During training, the learning systems see many fitting trials over different object and environment configurations together with their ground-truth fitting feasibility, *i.e.* $\{(O_i, E_i, \text{Successful/Failed})|i = 0, 1, 2, \cdots\}$. The goal is to learn eigen-length functions based on which correct prediction of task feasibility given test input (O_{test}, E_{test}) can be made.

4. Can Geometric Eigen-Lengths be Learned from Binary Task Supervision?

In this work, we are interested in learning geometric eigen-lengths that are crucial for downstream tasks. We hope to achieve automatic discovery of these eigen-lengths from doing tasks as it requires the least human prior and allows maximum flexibility. Therefore, we start with the minimum form of supervision and explore the following question: given only binary task success/failure supervision, is it possible to learn geometric eigen-lengths of input geometries that are sufficient for the task?

4.1. Testbed for Eigen-Length Learning

We start by curating a set of tasks as the testbed for the learning problem, as summarized in Fig. 3. For each task, we build a large-scale dataset comprising diverse shapes and configurations.

Task Design Principles We design the tasks to (1) cover a wide range of geometries, including synthetic, simple primitive shapes and more complex ones like ShapeNet objects; (2) facilitate the analysis and interpretation of learned
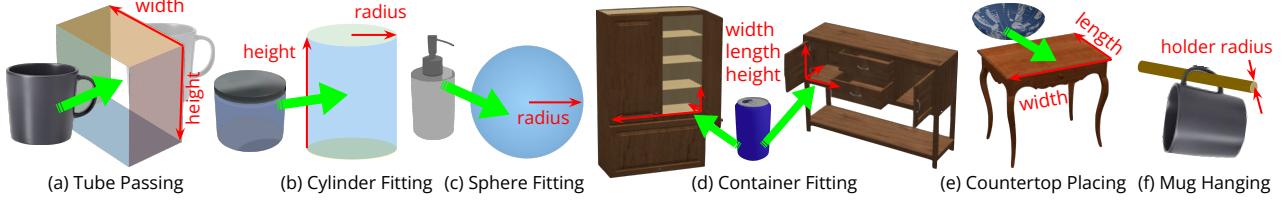
Figure 3. **Summary of tasks** and their human-hypothesized key measurements/eigen-lengths.
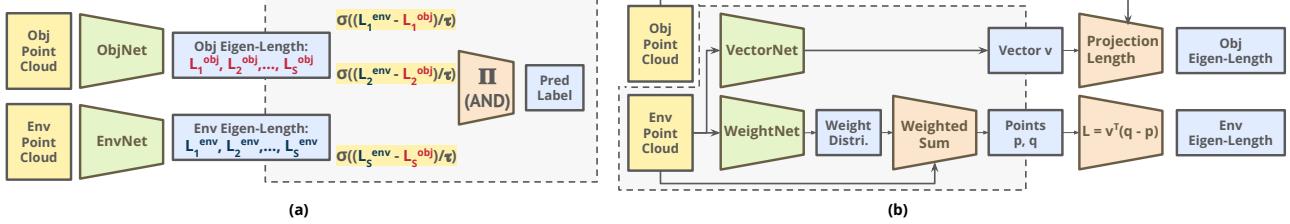
Figure 4. **Network architectures.** (a) A minimal eigen-length learning pipeline where we separately encode environment and object into eigen-length values, perform pair-wise comparison, and take the logical AND of results. (b) A geometry-grounded framework where we first predict vectors and points as the geometry grounding, then compute eigen-lengths from them.

eigen-lengths. Specifically, here we base the analysis on comparisons to human-hypothesized eigen-lengths: given a task, humans can identify related key eigen-lengths (referred to as "ground truth" in the following), *e.g.*, object height when putting them on shelves. Comparing the learned eigen-lengths to these "ground truth" may provide important insights. To achieve this, we need accessible ground truth eigen-lengths to begin with. Primitive shapes like cylinders are ideal as they are parameterized by eigen-lengths like radius and height.

Task Specifications In all tasks, we aim to determine whether a placement/motion of the object exists in a certain environment, specifically:

(a) **Tube passing. (Tube)** Pass an object through a rectangular tube. A *tube* is a cuboid without the front and back faces. Width and height of the tube/object are the key eigen-lengths.

(b) **Cylinder fitting. (Cylinder)** Place an object into a cylindrical container. Bounding sphere radius of the object in XY plane, its height, the radius and height of the cylinder container are the key eigen-lengths.

(c) **Sphere fitting. (Sphere)** Place an object into a spherical container. Radii of the bounding sphere of the object and the container are the key eigen-lengths.

(d) **Container fitting. (Fit)** Place an object into cavities in a ShapeNet container object. Example *cavities* are drawers or shelves (See Fig. 3d). Most cavities have cuboid-like shapes. Thus, key eigen-lengths are width, length and height of cavities and objects.

(e) **Countertop placing. (Top)** Place an object on top of another ShapeNet environment object, such that its projection along the gravity axis is fully enclosed by

the environment countertop. Width and length of the countertop surface and the object are key eigen-lengths.

(f) **Mug hanging. (Mug)** Hang a mug on a cylinder-shaped mug holder by its handle. Key eigen-lengths are the distance between the handle and mug body and the diameter of the mug holder.

Data Generation Details For objects to be fitted in tasks **(a)-(e)**, we use ∼1200 common household object models from 8 training and 4 testing categories in ShapeNet (Chang et al., 2015), following Mo et al. (2021b). We apply random scaling *and rotation* to the object model, then sample $N = 1024$ points from its surface. In **(d),(e)**, we use furniture and appliances from ShapeNet as the environment, including ∼550 shapes from 7 categories. In **(f)**, we use ∼200 ShapeNet mugs. We randomly sample the parameters of primitive shapes and the scaling factors of ShapeNet shapes, then sample $M = 1024$ points from their surfaces. For all tasks, we generated 75k training and 20k testing environment-object pairs. Please refer to Appendix A.3 for more data generation details.

4.2. A Minimal Network Architecture

Intuitively, we can measure the object and the environment separately and see if the object is "smaller" than the environment. Thus we come up with the minimal network architecture shown in Fig. 4(a). We separately map the object and environment geometries into two sets of eigen-lengths, perform pairwise comparisons between them, and compose comparison results using logical AND.

More specifically, we encode object point cloud O and environment point cloud E using two PointNet (Qi et al., 2017) networks, ObjNet and EnvNet. Both networks out-
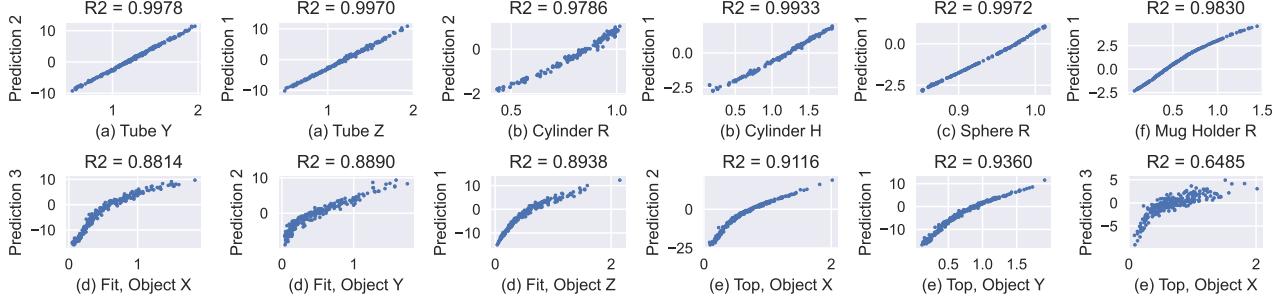
Figure 5. **Correlation Analysis.** Each plot shows the relationship between one learned eigen-length (Y coord.) and its matching "ground truth" measurement (X coord.). Higher R^2 values imply a stronger correlation. Please refer to Appendix B.2 for complete $S \times S'$ plots.

put S-dim vectors $\vec{L^{obj}} = (L_1^{obj}, L_2^{obj}, \ldots, L_S^{obj})$, $\vec{L^{env}} = (L_1^{env}, L_2^{env}, \ldots, L_S^{env})$. We then compute task success as $\hat{T}(E,O) = \bigwedge_{s=1}^{S}[L_s^{env}(E) > L_s^{obj}(O)]$. During training, we use a differentiable approximation $\tilde{T}(E,O) = \prod_{s=1}^{S} \sigma((L_s^{env}(E) - L_s^{obj}(O))/\tau)$, where τ is a learnable parameter. We set $S = 1$ for **(c)** Sphere, **(f)** Mug, $S = 2$ for **(a)** Tube, **(b)** Cylinder, $S = 3$ for **(d)** Fit, **(e)** Top. We supervise $\tilde{T}$ with binary cross-entropy loss.

4.3. Analysis of Learned Eigen-Lengths

We analyze the eigen-lengths learned by the network by comparing them to "ground truth" eigen-lengths as shown in Fig. 3. For each task, we randomly sample $N = 512$ test data points and obtain the corresponding N eigen-length predictions $\{L_{s,i}^{pred}\}_{i=0,\ldots,N-1}$ for each of the S learned eigen-lengths, as well as N values $\{L_{s',i}^{gt}\}_{i=0,\ldots,N-1}$ for each of the S' "ground truth" eigen-lengths.

For each pair of predicted and "ground truth" eigen-lengths (s, s'), we draw a scatter plot of points $(L_{s',i}^{gt}, L_{s',i}^{pred})$ and perform least squares linear regression over them to get corresponding R^2-scores. We match the predictions and groundtruths by maximizing the sum of R^2-scores and show the scatter plots of matched pairs in Fig. 5. Note that in **(e)** Top, since we predict $S = 3$ eigen-lengths while there are only $S' = 2$ groundtruth eigen-lengths, we show the unmatched prediction with its most correlated groundtruth. For complete $S \times S'$ plots, please refer to Appendix B.2.

Learned eigen-lengths are strongly correlated with human-hypothesized measurements. As Fig. 5 shows, R^2 values between predictions and "ground truths" are close to or greater than 0.9 except for the redundant prediction slot 3 in **(e)** Top. They also have clear one-to-one correspondences with ground truth in tasks with multiple eigen-lengths, suggesting good disentanglement is learned.

Knowing the number of eigen-lengths beforehand is not a requirement for successful learning. The number S of eigen-lengths to learn is a hyperparameter set before learning. However, it does not have to be the exact number of relevant eigen-lengths. As shown in **(e)** Top, when we have more slots for eigen-lengths than needed, "ground truth"

eigen-lengths are still captured by the first two predictions. The third prediction does not strongly correlate with any "ground truth". A further probe reveals that comparisons of this eigen-length almost never (only in 0.4% of the cases) contribute to the final result, outputting `True` most of the time. The network learns a pair of degenerate eigen-lengths as there is no more necessary information to capture. Please see Appendix E.3 for further discussion.

5. Can Geometry Groundings be Discovered for Eigen-Lengths?

While Fig. 5 shows a strong correlation between learned eigen-lengths and "ground truth", their relationship is not always perfectly linear, as can be observed in **(d)** Fit and **(e)** Top with complex geometries. Even in more linear cases, the scaling and offset make the raw eigen-length value hard to understand, *e.g.*, negative "length" values are less intuitive. As eigen-lengths can be seen as measurements of the object, many of them have sparse supports or geometry groundings on the objects, *e.g.*, height is the distance between the base plane supporting the object and its highest point. These geometry groundings anchor the corresponding eigen-length values, provide an intuitive explanation of these values, and usually carry geometric/semantic importance themselves. We are therefore interested in the following question: can we ground the eigen-lengths on geometry? From a high level, instead of directly predicting eigen-length values, if we first predict some geometric entities like points, vectors, and planes, then derive eigen-lengths from them, is it possible to learn meaningful eigen-lengths and geometry groundings?

5.1. Grounding Eigen-Length Predictions on Geometric Primitives

Consider fitting tasks like **(d)** Container Fitting and **(e)** Countertop Placing where the spaces affording the task can be roughly described by a set of parallel planes. [1] To compute

---

[1]Note that other tasks may require other inductive bias. We focus on this type of tasks to study the feasibility of geometry-grounded eigen-length learning. We leave a more versatile system as future work.
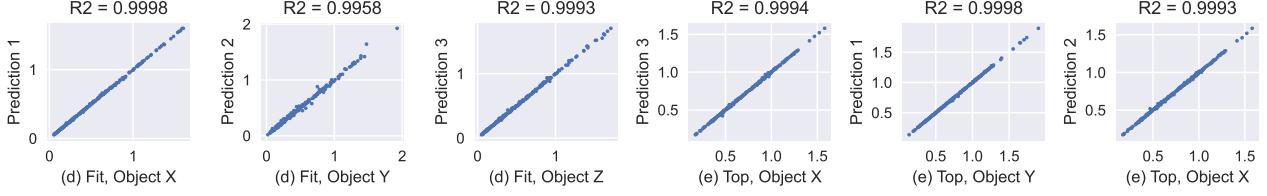
Figure 6. **Improved correlation after using geometry groundings.** We show scatter plots of predicted eigen-length (Y coord.) and their matching "ground truth" (X coord.) in **(d)** Fit and **(e)** Top. Please refer to Appendix B.2 for complete $S \times S'$ plots.
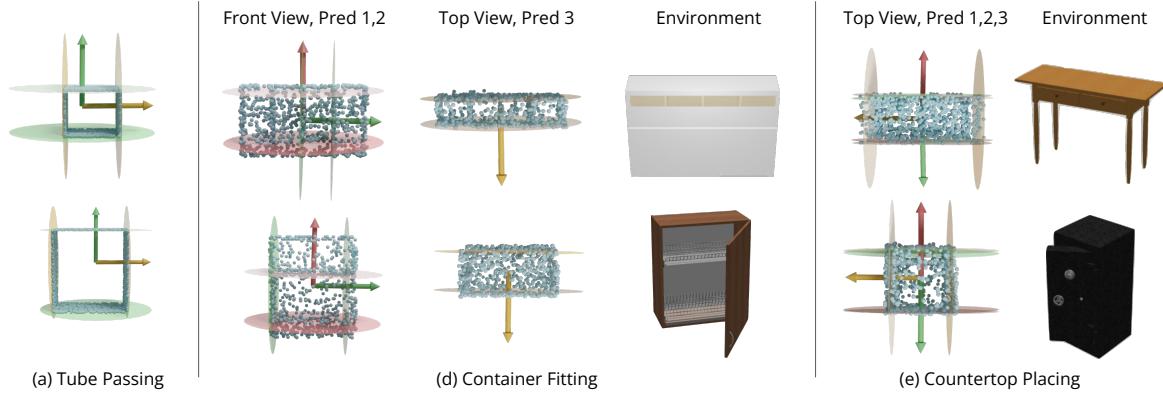
Figure 7. **Geometry Grounding Visualizations.** We plot the learned vectors (as arrows) and planes (as disks) on top of input environment point clouds. We also show the object model next to point clouds for a clearer view of object structure. For **(d)** Fit, we visualize predictions in two views for clarity. Please refer to Appendix B.1 for more visualizations.

the success label of the task, say fitting an object into a night-stand, we can measure the size of the spaces of interest in the environment (the drawer part) along important directions (its main axes) and compare it to the measurement of the object. Inspired by this, we ground a pair of eigen-lengths on a tuple of one unit vector and two planes $(\vec{v}, \Pi_p, \Pi_q)$ as illustrated in Fig. 8: we measure both the object and the environment along $\vec{v}$. We take the object measurement as the diameter of the projection of the object point cloud O on the vector $\vec{v}$, *i.e.* $L^{obj}(O) = \max_{p \in O} \vec{v}^T p - \min_{p \in O} \vec{v}^T p$. For the environment, we use a pair of parallel planes Π_p, Π_q with normal $\vec{v}$ to separate out a certain region relevant to the task (the drawer), then measure the distance between the planes. In practice, we adopt the (point, normal) plane representation and predict a point pair (p, q) that determines the plane pair. The environment eigen-length is then computed as $L^{env}(E) = \vec{v}^T(q - p)$.

Figure 4(b) illustrates our network. In **VectorNet**, we employ a PointNet classification backbone to extract global feature of the environment point cloud $E \in \mathbb{R}^{M \times 3}$, then use an MLP to predict S 3D vectors $\{\vec{v}_s\}_{s=1,2,\dots,S}$. In **WeightNet**, we employ a PointNet segmentation backbone to extract per-point features, then use $S \times 2$ MLPs to predict S pairs of probability distributions W_s^p, W_s^q over the point cloud. The point coordinates of (p_s, q_s) are then computed as the weighted average of original point cloud coordinates,
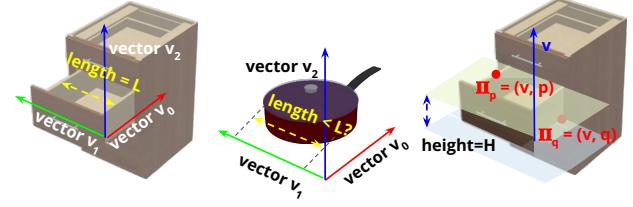
Figure 8. **Eigen-Length Geometry Groundings.** We ground each eigen-length L with a unit vector v and two parallel planes Π_p, Π_q with normal v. L is computed as the distance between Π_p, Π_q.

namely $p_s = W_s^{pT} E, q_s = W_s^{qT} E$.

5.2. Analysis of Learned Geometric Primitives and Eigen-Length Values

We perform the same correlation analysis and visualize the results in Fig. 6. Compared to Fig. 5, learned eigen-lengths are now almost equal to "groundtruth" thanks to the anchoring effect of the geometry grounding. The extra predicted eigen-length in **(e)** Top also behaves differently, capturing the same "ground truth" as another learned eigen-length. This suggests the regularization from geometry grounding makes learned eigen-lengths more likely to be meaningful measurements. It also reaffirms the fact that the number S of eigen-lengths we set in advance can be different from the actual number of key eigen-lengths. Please see Appendix E.3 for a detailed discussion.
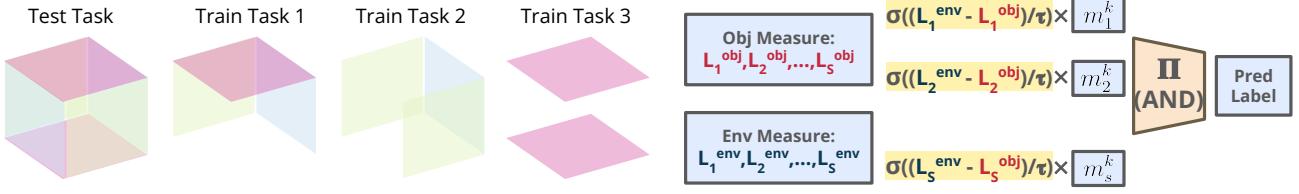
Figure 9. **(a) Multi-Task Setting** where each train task uses boxes with certain faces missing as the environment geometry, and the test task uses a complete box; and **(b) Learning Framework**, where we use trainable masks to select eigen-length comparison results.

We also visualize the learned geometry groundings in Fig. 7. The learned vectors align with the main axes of object geometry. The learned planes overlap with tube surfaces in **(a)** Tube, surround the edge of countertops in **(e)** Top, and separate out the region of interest in **(d)** Fit, *e.g.* the higher one out of two storage spaces. These meaningful geometric entities provide a clear interpretation of learned eigen-lengths, *e.g.* in **(e)** Top's case, red and green predictions coincide with each other and both capture the back-to-front length of the countertop.

5.3. A Study on the Data Efficiency of Geometry-Grounded Eigen-Lengths

Geometry grounding of eigen-lengths can be seen as a form of regularization. We are therefore curious how the introduction of geometry groundings may influence the model's data efficiency. Fig. 10 shows the trend of test performances as we change the size of training data. We compare our geometry-grounded version to *Direct*, a no-eigen-length approach, where an MLP directly predicts the final label from the concatenation of object and environment latent features. We also plot the difference between "ground truth" eigen-length measurement directions (local up and right) and predicted vectors as a way to quantify eigen-length quality. Results suggest that the geometry-grounded version is more data efficient if meaningful geometry groundings emerge. When the training data is limited (< 3000 samples), however, the predicted directions of groundings are far from ground truth measurement directions, suggesting that the model fails to learn meaningful groundings for eigen-lengths, and thus the final accuracy is lower than *Direct*.
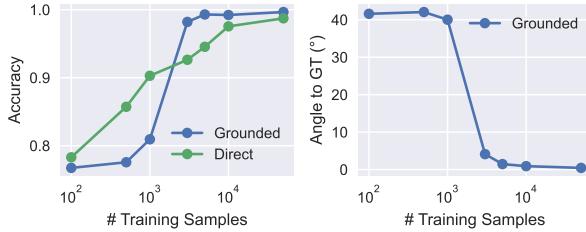
Figure 10. Trend of *Left:* test accuracy and *Right:* average angle between learned vector groundings and "ground truth" directions w.r.t. # training samples.

6. Can Eigen-Lengths be Learned in Multi-Task Settings and Applied to New Tasks?

As humans, we are able to develop a library of useful measurements/eigen-lengths like height from past experience. Given a new task, instead of trying cluelessly, we would start with known measurements and investigate their role in the task. In this section, we ask if learned eigen-lengths can work in a similar way, *i.e.*, given a set of training tasks, is it possible to learn a shared set of eigen-lengths from them? Further, given a novel task, can we learn to select a subset of learned eigen-lengths that are sufficient for it? In other words, can agents accumulate and transfer knowledge in the form of eigen-lengths?

6.1. Multi-Task Testbed

We design a set of tasks that share key eigen-lengths as the testbed for multi-task learning. As shown in Fig. 9(a), we consider box-fitting tasks where the box only has a subset of six faces. Each mode of face existence corresponds to a different task with different geometric constraints. For example, to be able to fit, an object has to be narrower than the box in Task 2 and shorter than the box in Task 3. We set aside the box with all six faces present as the test task. We expect to learn width, height, and length from the training task set, and learn to use all of them during testing. By boxes with partial faces, we aim to mimic different types of cavities in the furniture, e.g., a closed drawer as a box with all faces, an open space on the shelf as a box without the front face, etc.

6.2. Multi-Task Learning Framework

Fig. 9(b) shows the multi-task learning framework we experiment with. From a high level, we learn a set of S eigen-lengths and allow each task to select relevant ones from them. This selection step is implemented as a learnable binary mask $\{m_s^k\}_{s=1,2,...,S}$ over eigen-lengths for each task T_k. We simply insert the mask in the AND-composition and compute the outcome for T_k as $\prod_{s=1}^{S} m_s^k \cdot \sigma((L_s^{env}(E) - L_s^{obj}(O))/\tau)$.

During training, we optimize both the eigen-length prediction networks and a continuous version of per-task masks $\tilde{m}^k \in [0, 1]$. At test time, we freeze network weights and only learn a mask to choose from eigen-lengths learned dur-
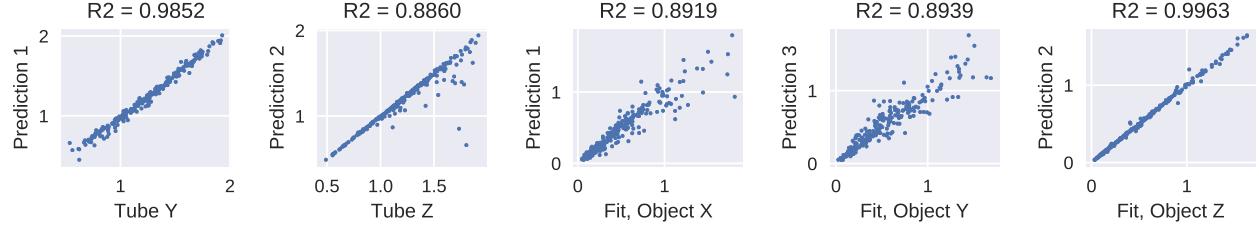
Figure 11. **Correlation scatter plots** of predicted eigen-length and their matching "ground truth" in rotated Tube Passing and rotated Container Fitting. Please refer to appendix B.3 for the full plot.

ing training. Notably, we limit the size of test task data to 10 batches (320 samples) to examine if learned eigen-lengths help in few-shot adaptation scenarios.

6.3. Multi-Task Learning and Few-Shot Test Task Adaptation

We experiment with both implicit and geometry-grounded eigen-length prediction networks. To analyze the learned eigen-lengths and per-task masks, we visualize learned geometry groundings that are selected ($m_s > 0.5$) in each task in Fig. 12. Meaningful groundings are learned and correctly selected for each task, including the test task.

To explore whether eigen-lengths learned during training help quicker adaptation to new tasks, we compare the test task performance of *Implicit* (Section 4), *Grounded* (Section 5) to *Direct* trained from scratch on the test task. Here *Direct* directly predicts the final label from object and scene latent codes. All methods are limited to 10 batches of test task samples. As shown in Table 1, within one epoch of finetuning, methods based on the reuse of learned eigen-lengths already achieve high performance, surpassing *Direct* trained from scratch by a large margin, even when the latter has been trained for 100 epochs.
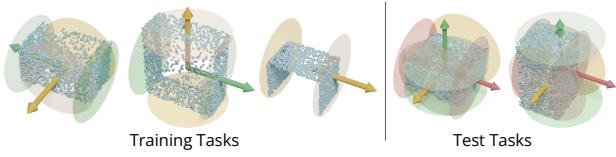
Figure 12. **Learned Geometry Grounding in Multi-Task Setting.** We only show learned geometry grounding (vectors as arrows, planes as disks) selected by the mask in each task.

Table 1. Multi-Task learning, novel task adaptation results. We finetune eigen-length-based methods on novel task for 1 epoch and compare them to the direct method trained from scratch for 1 and 100 epochs.

	Single Task Direct		Eigen-Length	
			Implicit	Grounded
Epoch	1	100	1	1
Accuracy	73.14	88.47	97.71	99.48

7. Extension to More Challenging Tasks

So far, we have focused on single-step fitting tasks in controlled settings for better understanding and analysis of the learned eigen-lengths. In this section, we extend our method to more challenging settings and tasks to demonstrate its potential in a wider application scope.

7.1. Applying Random Rotations to Input Environments

In previous experiments, we take environment geometry directly from ShapeNet(Chang et al., 2015) where shapes are roughly axis-aligned. We now consider a more challenging setting where we randomly rotate the environment geometry in Container Fitting and Tube Passing tasks. We apply the geometry-based method described in Section 5. Fig.11 shows the correlation scatter plots of the learned eigen-lengths with the closest "ground-truth" measurements. Despite the increased difficulty, strong correlations can still be observed. Fig. 13 visualizes the learned geometry groundings, where the predicted planes roughly align with the main cavities of the objects.
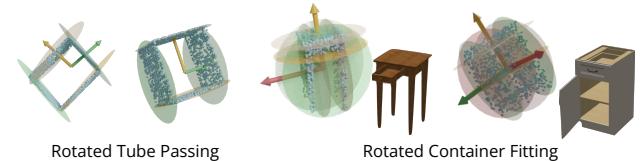
Figure 13. **Visualization of learned geometry groundings** in rotated Tube Passing and rotated Container Fitting. The learned vectors and planes roughly align with the rotated object. Regions of interest like drawers are also selected by planes.

7.2. Relative Rotation Estimation

The network learned in 7.1 not only discovers geometric eigen-lengths but also vectors as their geometry groundings. These vectors are intrinsic properties of the geometry and can serve as a strong indicator of the object's rotation. In this section, we show how they can be leveraged to estimate object relative rotation. Specifically, given two randomly rotated versions $(R_a P, R_b P)$ of point cloud P, estimate the relative rotation $R_b R_a^{-1}$ between them.

We directly use the network trained in 7.1, which outputs

S vectors $\vec{v_1}, \vec{v_2}, \ldots, \vec{v_S}$ as part of the geometry grounding. We feed the input point cloud pair $(R_a P, R_b P)$ to the network, get two sets of unit-length vectors $V^a, V^b \in \mathbb{R}^{3 \times 3}$, and compute their relative rotation difference. Please see C.1 for more details.

The network trained on Tube Passing/Container Fitting tasks with only binary supervision achieves an average rotation error of 25.39º on ShapeNet furniture, and 13.63º on Tubes. Figure 14 visualizes the predicted rotations, which aligns considerably well with ground truth.
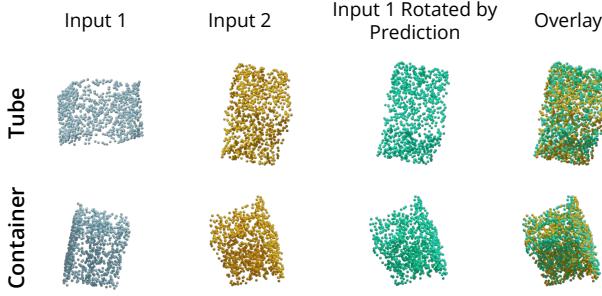
Figure 14. **Visualization of predicted relative rotations**. From left to right, we show the two input point clouds P_a, P_b, the first point cloud transformed by the predicted rotation RP_a, and its overlay with P_b. Good alignment can be observed, suggesting good relative rotation estimation.

7.3. Embodied Visual Navigation

Eigen-lengths are widely useful in geometric tasks that go beyond fitting. Here we demonstrate its application in embodied visual navigation. We consider cylinder-shaped robots (real-life examples are robot vacuums or robot waiters) navigating scenes from AI2-THOR (Kolve et al., 2017). The goal is to devise a navigation policy that avoids obstacles, based on the robot's egocentric visual input, *i.e.* the single-view point cloud from its forward-facing depth camera.

We employ the minimal framework in Section 4 to learn whether the robot can move forward by $d = 0.2m$. We use the robot point cloud as the object, its egocentric point cloud observation as the environment, and collect training labels by moving randomly-placed robots in simulation. The network achieves 93.3% accuracy on views from the unseen testing scene, showing good generalization.

Based on the learned feasibility, we implement a visual navigation policy and deploy it on a robot vacuum operating in real-time in the testing scene. With a simple policy of moving forward when the network outputs positive and turning clockwise by 20 degrees otherwise, the robot is able to navigate around the room. Figure 15 shows a snapshot of the navigation process. Please refer to the project website for the full demo video and see C.2 for more details.
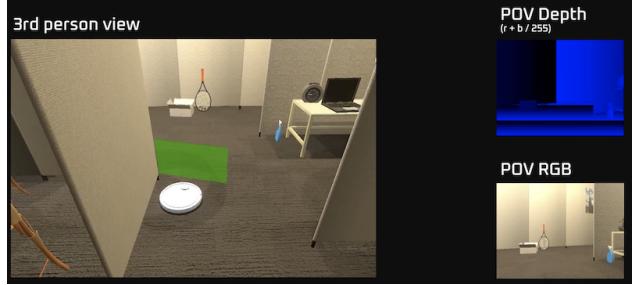
Figure 15. **Snapshot of the visual navigation scenario**. The predicted environment eigen-lengths are visualized with a rectangle, indicating the size of the navigable space. The color (green/red) represents the final output (positive/negative) obtained by comparing the eigen-lengths of the robot and the environment.

8. Conclusion

In this work, we formulate a novel learning problem of automatically discovering low-dimensional geometric eigen-lengths crucial for fitting tasks. We set up a benchmark suite comprising a curated set of fitting tasks and corresponding datasets, as well as metric and tools for analysis and evaluation. We demonstrate the feasibility of learning meaningful eigen-lengths as sufficient geometry summary only from binary task supervision. We show that proper geometry grounding of the eigen-lengths contributes to their accuracy and interpretability. We also make an initial attempt at learning shared eigen-lengths in multi-task settings and applying them to novel tasks.

Our exploration suggests broad opportunities in this new research direction and reveals many challenges. For example, grounding eigen-length predictions on geometries requires reasonable choice of geometric primitives, which relies on inductive bias of the specific tasks considered. It would be a challenging future direction to build a universal framework that accommodates a wide range of tasks by leveraging all kinds of geometric primitives. In many task instances, we may have access to signals beyond binary success or failure, *e.g.*, a possible placement position of the object. How to leverage these task signals in eigen-length learning remains an open problem. As a first-step attempt at defining and exploring the challenging problem of eigen-length learning, we do hope our work can inspire more researchers to work on this important yet underexplored direction.

Acknowledgements

References

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Chen, B., Abbeel, P., and Pathak, D. Unsupervised learning of visual 3d keypoints for control. In *International Conference on Machine Learning*, pp. 1539–1549. PMLR, 2021.

Chen, N., Liu, L., Cui, Z., Chen, R., Ceylan, D., Tu, C., and Wang, W. Unsupervised learning of intrinsic structural representation points. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9121–9130, 2020.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 2180–2188, 2016.

Deng, B., Genova, K., Yazdani, S., Bouaziz, S., Hinton, G., and Tagliasacchi, A. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 31–44, 2020.

Deng, S., Xu, X., Wu, C., Chen, K., and Jia, K. 3d affordancenet: A benchmark for visual object affordance understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1778–1787, June 2021.

Genova, K., Cole, F., Vlasic, D., Sarna, A., Freeman, W. T., and Funkhouser, T. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7154–7164, 2019.

Genova, K., Cole, F., Sud, A., Sarna, A., and Funkhouser, T. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4857–4866, 2020.

Higgins, I., Matthey, L., Glorot, X., Pal, A., Uria, B., Blundell, C., Mohamed, S., and Lerchner, A. Early visual concept learning with unsupervised deep learning. *arXiv preprint arXiv:1606.05579*, 2016.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

Jakab, T., Tucker, R., Makadia, A., Wu, J., Snavely, N., and Kanazawa, A. Keypointdeformer: Unsupervised 3d keypoint discovery for shape control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12783–12792, 2021.

Kim, D. I. and Sukhatme, G. S. Semantic labeling of 3d point clouds with object affordance for robot manipulation. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5578–5584. IEEE, 2014.

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., and Farhadi, A. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017.

Manuelli, L., Gao, W., Florence, P., and Tedrake, R. kpam: Keypoint affordances for category-level robotic manipulation. *arXiv preprint arXiv:1903.06684*, 2019.

Mo, K., Guibas, L. J., Mukadam, M., Gupta, A., and Tulsiani, S. Where2act: From pixels to actions for articulated 3d objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6813–6823, 2021a.

Mo, K., Qin, Y., Xiang, F., Su, H., and Guibas, L. O2O-Afford: Annotation-free large-scale object-object affordance learning. In *Conference on Robot Learning (CoRL)*, 2021b.

Paschalidou, D., Ulusoy, A. O., and Geiger, A. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Paschalidou, D., van Gool, L., and Geiger, A. Learning unsupervised hierarchical part decomposition of 3d objects from a single rgb image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

Qin, Z., Fang, K., Zhu, Y., Fei-Fei, L., and Savarese, S. Keto: Learning keypoint representations for tool manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7278–7285. IEEE, 2020.

Reddy, P., Gharbi, M., Lukac, M., and Mitra, N. J. Im2vec: Synthesizing vector graphics without vector supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7342–7351, 2021.

Redmon, J. and Farhadi, A. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.

Sharma, G., Liu, D., Maji, S., Kalogerakis, E., Chaudhuri, S., and Měch, R. Parsenet: A parametric surface fitting network for 3d point clouds. In *European Conference on Computer Vision*, pp. 261–276. Springer, 2020.

Siddharth, N., Paige, B., Van de Meent, J.-W., Desmaison, A., Goodman, N. D., Kohli, P., Wood, F., and Torr, P. H. Learning disentangled representations with semi-supervised deep generative models. *arXiv preprint arXiv:1706.00400*, 2017.

Smirnov, D., Fisher, M., Kim, V. G., Zhang, R., and Solomon, J. Deep parametric shape predictions using distance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 561–570, 2020.

Sun, C.-Y., Zou, Q.-F., Tong, X., and Liu, Y. Learning adaptive hierarchical cuboid abstractions of 3d shape collections. *ACM Transactions on Graphics (TOG)*, 38(6): 1–13, 2019.

Tulsiani, S., Su, H., Guibas, L. J., Efros, A. A., and Malik, J. Learning shape abstractions by assembling volumetric primitives. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

Turpin, D., Wang, L., Tsogkas, S., Dickinson, S., and Garg, A. GIFT: Generalizable Interaction-aware Functional Tool Affordances without Labels. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.060.

Wang, J., Lin, S., Hu, C., Zhu, Y., and Zhu, L. Learning semantic keypoint representations for door opening manipulation. *IEEE Robotics and Automation Letters*, 5(4): 6980–6987, 2020.

Xiang, F., Qin, Y., Mo, K., Xia, Y., Zhu, H., Liu, F., Liu, M., Jiang, H., Yuan, Y., Wang, H., Yi, L., Chang, A. X., Guibas, L. J., and Su, H. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
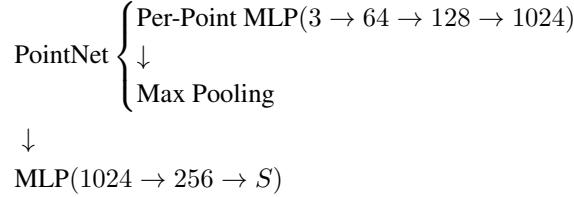
Yang, Y., Chen, Y., and Soatto, S. Learning to manipulate individual objects in an image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6558–6567, 2020.

A. Implementation Details

Below we provide network and dataset details. We will also release our code and data to facilitate future research.
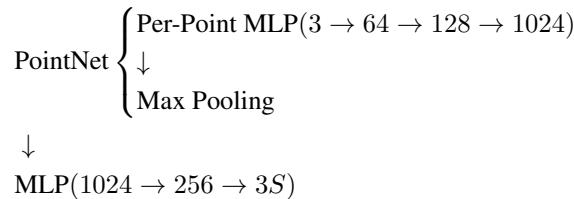
A.1. Network Architecture

The framework in Section 4 consists of a PointNet and an MLP output head that maps the PointNet global feature to S scalar values. The architecture is outlined below, where the numbers in the parenthesis refer to the number of channels in each layer. We use batch normalization and LeakyReLU after all FC layers, except for the output layer.

$$\text{PointNet} \begin{cases} \text{Per-Point MLP}(3 \rightarrow 64 \rightarrow 128 \rightarrow 1024) \\ \downarrow \\ \text{Max Pooling} \end{cases}$$
$$\downarrow$$
$$\text{MLP}(1024 \rightarrow 256 \rightarrow S)$$
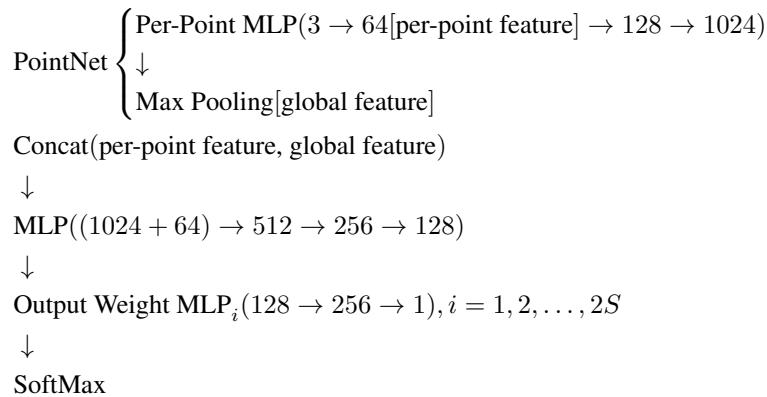
Output: S scalars.

The framework in Section 5 consists of VectorNet and WeightNet. VectorNet consists of a PointNet classification backbone and an MLP output head, as outlined below.

$$\text{PointNet} \begin{cases} \text{Per-Point MLP}(3 \rightarrow 64 \rightarrow 128 \rightarrow 1024) \\ \downarrow \\ \text{Max Pooling} \end{cases}$$
$$\downarrow$$
$$\text{MLP}(1024 \rightarrow 256 \rightarrow 3S)$$

Output: S vectors.

WeightNet consists of a PointNet segmentation backbone and $2S$ parallel MLP output heads, each outputs a weight distribution over all points, as outlined below.

$$\text{PointNet} \begin{cases} \text{Per-Point MLP}(3 \rightarrow 64[\text{per-point feature}] \rightarrow 128 \rightarrow 1024) \\ \downarrow \\ \text{Max Pooling}[\text{global feature}] \end{cases}$$
$$\text{Concat(per-point feature, global feature)}$$
$$\downarrow$$
$$\text{MLP}((1024 + 64) \rightarrow 512 \rightarrow 256 \rightarrow 128)$$
$$\downarrow$$
$$\text{Output Weight MLP}_i(128 \rightarrow 256 \rightarrow 1), i = 1, 2, \dots, 2S$$
$$\downarrow$$
$$\text{SoftMax}$$

Output: $2S$ sets of per-point weights.

We use LeakyReLU and batch normalization after each FC layer except for the output layers.

A.2. Training Details

All networks are implemented using PyTorch and optimized by the Adam optimizer, with a learning rate starting at 10^{-3} and decay by half every 10 epochs. Each batch contains 32 data points; each epoch contains around 1600 batches. We train models for ~ 100 epochs on all tasks. The learnable parameter τ is initialized with $\tau = 1$. All experiments are run on a single NVIDIA TITAN X GPU.

A.3. Dataset Details

Table 2 and 3 summarizes the statistics of environment/object shapes used in our dataset. Each shape is drawn with probability in inverse proportion to the number of shapes in its category, such that each object category appears with similar frequency in the final dataset.

During data generation for the tasks where both the environment and the object are ShapeNet objects, we apply random scaling $s \sim U([0.9, 1.1])$ to the environment objects, set all joints to closed state and sample $M = 1024$ points from the object model. Given an object-environment pair, we randomly sample $T = 1000$ candidate positions in the environment point cloud, and check whether placement of the object at each candidate satisfy the task specification using SAPIEN (Xiang et al., 2020) simulation. If all candidates fail, we label the pair as negative, otherwise as positive. Specifically, the candidate positions are sampled from "applicable and possible regions" following Mo et al. (2021b)'s definition. For example, we only consider points with upward facing normals, and for task (e) only consider points with close to highest z coordinates. We generated around 75K training data and 20K testing data for each task.

Table 2. Environment Shape Statistics.

	Box	Microwave	Refrigerator	Safe	Storage Furniture	Table	Washing Machine	Total
Train	21	9	34	21	272	70	13	440
Test	7	3	9	7	73	25	3	127

Table 3. Object Shape Statistics.

				Train				
Basket	Bottle	Bowl	Box	Can	Pot	Mug	TrashCan	Total
77	16	128	17	65	16	134	25	478
				Test				
	Bucket		Dispenser		Jar		Kettle	Total
	33		9		528		26	554

B. Additional Results

B.1. Geometric Grounding Visualization and Failure Case Discussion

Fig. 16 and 17 show more visualizations of the learned eigen-lengths in the three tasks from the main paper. Our framework is able to learn reasonable eigen-lengths that measure along crucial directions. These eigen-lengths are also grounded by planes that suggest the relevant part of object which supports the task. In experiments with primitive shapes as environments, the learned planes almost overlap with the box/tube faces. In experiments with ShapeNet container objects as environments, especially task (d) (*Fit*, or container fitting) as shown in Fig. 17, locating the relevant part becomes more challenging. As this usually involves finding cavities in a shape and selecting the largest one. Fig. 17 shows examples of our learned eigen-lengths, most of which make sense, as shown in (a)-(o). We are able to ignore irrelevant parts, e.g. the legs of tables, and find the part of object that affords the "containment" task, e.g. the drawer in (b), the closet in (c). When there are many cavities that afford the same task, the network picks the largest one, e.g. in (d) and (k).

Failure Cases. We also observe some failure cases where the learned eigen-lengths are inaccurate. Fig. 17(p)-(t) shows the most representative ones. (p) shows a relatively complex shape, where the network struggles to find the correct width of the drawer. (q) and (r) show cases where the network finds the wrong cavity. According to our task definition, the object can only be placed in the drawer part in (q). Instead, the network finds the part on top of the drawer. In (r), the network finds the second largest cavity instead of the largest one at the bottom. (s) shows an extreme case where the height of the pizza box is much smaller than the other two extents. As objects usually have correlated extents, comparing height suffices most of the time. The network probably lacks the motivation to precisely capture the width and the length of the pizza box, resulting in the underestimation of width and length in (s). Finally, our formulation, i.e. the AND clause of three eigen-length comparisons, can not fully and precisely describe the nature of this task. The washing machine in (t) has a cylinder-shaped cavity, which our network tries to approximate by a cuboid, which is reasonable within the range of its

(a) Tube Passing

(e) Countertop Placing
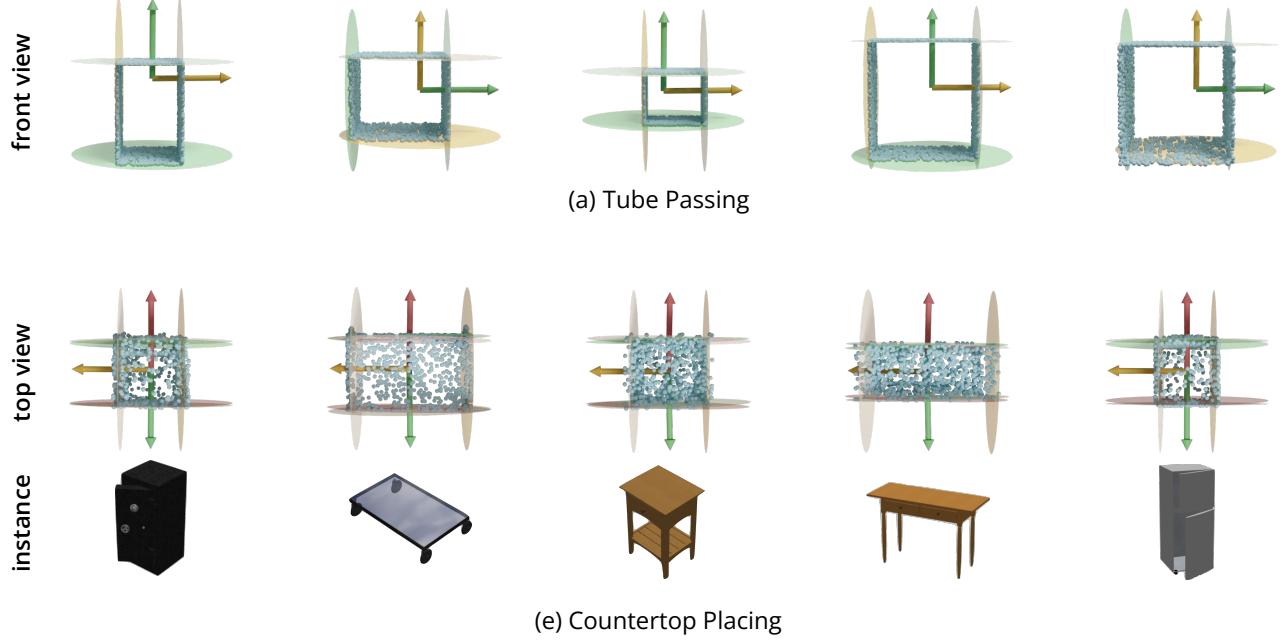
Figure 16. **Additional qualitative results.** We visualize the learned vectors and planes for (a) Tube Passing and (e) Countertop Placing. We show all eigen-lengths in the front(a)/top(e) view. We also show the underlying instances in task (e) countertop placing for a clearer understanding of the object structure. Note that though some joints are "open" for visualization purpose, all instances in the dataset are at their rest state.

expressive power but not accurate. Also, there could be shapes that do not have a "largest" cavity, e.g. some drawers in a closet may be designed for tall and narrow things, while others are designed for flat things. To deal with arbitrary objects, the extents of both types of drawers are useful. Introducing more complex and flexible formulations, e.g. in Section D, would help better capture the complexity of the task.

B.2. Correlation Analysis Results

We show here the scatter plots and correlation R^2 values between all prediction eigen-lengths and all presumable geometric measurements. R^2 value, or coefficient of determination, is a metric in $[0, 1]$ reflecting linear correlation between two variables. The closer R^2 is to 1, the more linearly correlated the two variables are. Given two set of samples x_i, y_i, where $i = 1, 2, \ldots, n$, R^2 is defined between y_i and the least squares linear regression of y_i on x_i, $\tilde{y}(x_i)$:

$$R^2 = 1 - \frac{\sum_i \left(y_i - \tilde{y}(x_i)\right)^2}{\sum_i \left(y_i - \bar{y}\right)^2},$$

where $\bar{y} = \frac{1}{n} \sum_i y_i$ is the mean value of y_i.

Results from *Eigen-Length-Implicit* are shown in Fig. 18. Results from *Eigen-Length-Grounded* are shown in Fig. 19. We can clearly see the one-to-one correspondence between predictions and presumable measurements. R^2 is close to or greater than 0.9 where the prediction is the match for the measurements, otherwise the value is much smaller. It is more apparent in the *Eigen-Length-Grounded* variant, where R^2 values are close to the theoretical bound 1 when it matches. The models can learn a compact and appropriate set of eigen-lengths from binary task supervision. Also note that the extraneous prediction slot in task (e) (*Top*, or countertop placing) become degenerate with another prediction slot, as has mentioned before in main text.
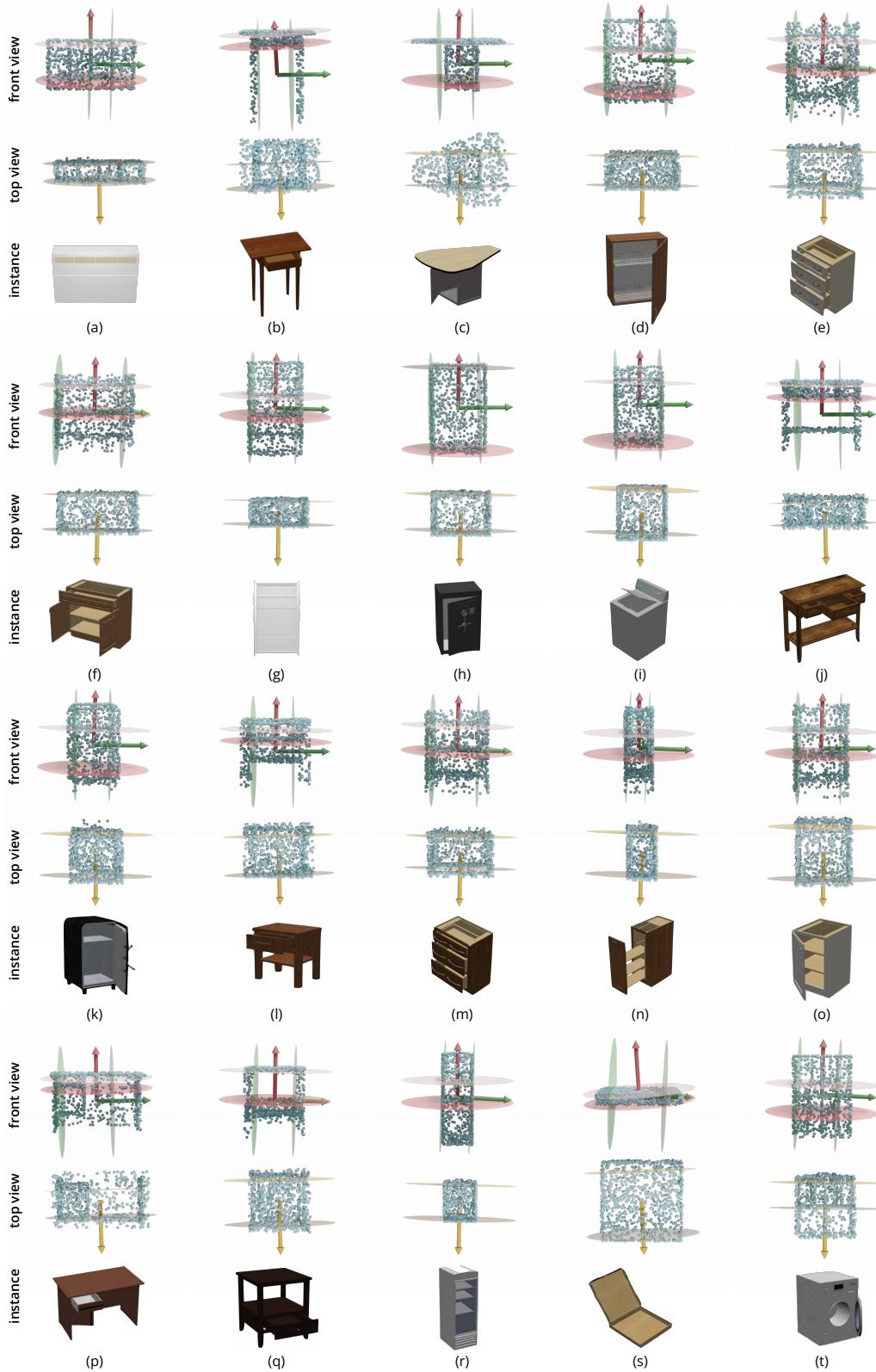
Figure 17. **Additional qualitative results in Container Fitting**. We show eigen-lengths in two views together with the underlying object following Fig. 16 (d). (a)-(o) are successful cases where the learned planes correctly separate out the largest cavity in the object. (p)-(t) show failure cases.
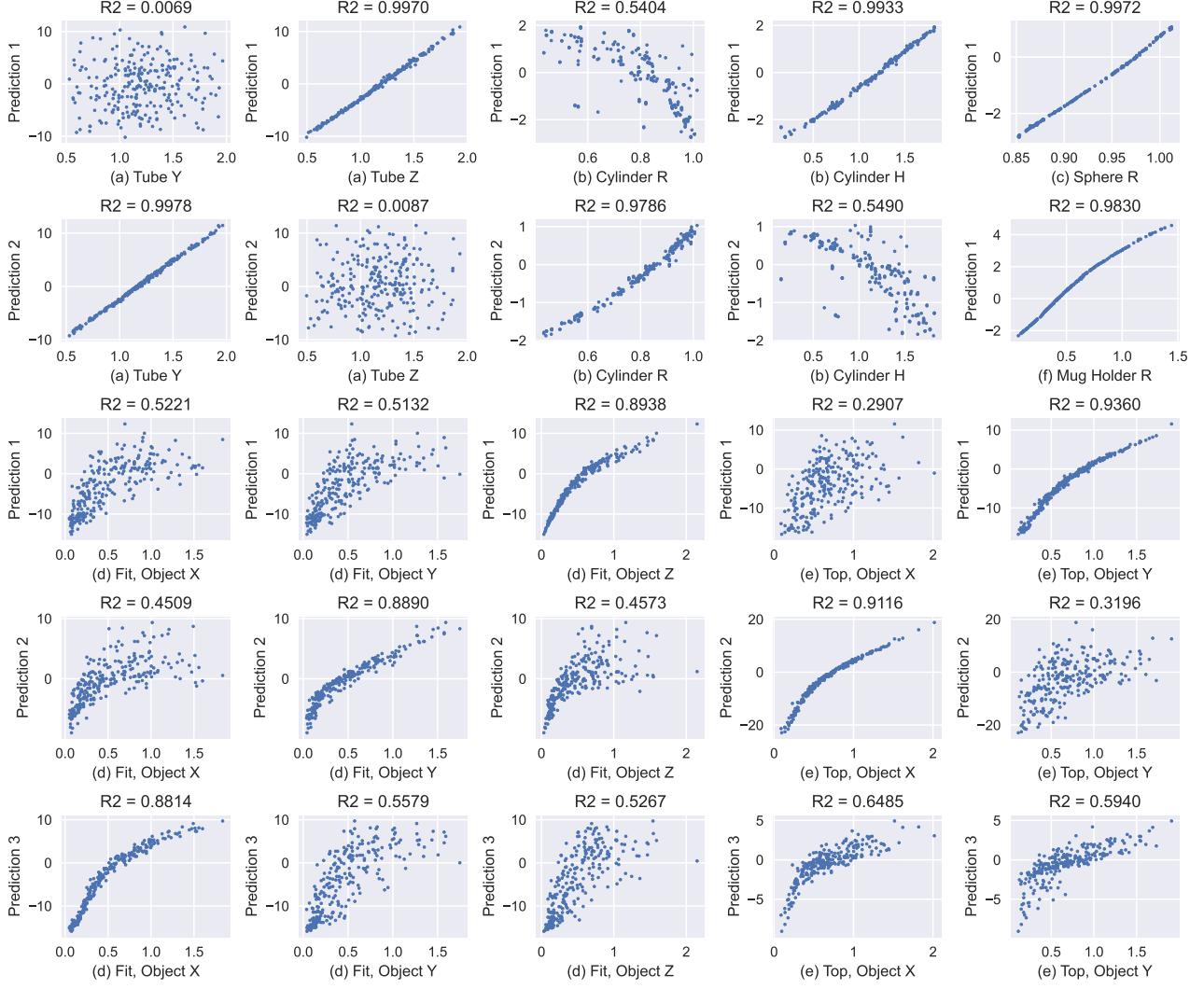
Figure 18. **Full correlation plots and respective** R^2 **values** between ground truth measurements and predicted eigen-lengths from *Eigen-Length-Implicit*.

B.3. Correlation Analysis Results from Rotated Environment Experiments

Fig. 20 shows the full correlation analysis of learned eigen-lengths from rotated Tube Passing and rotated Container Fitting, where the environment geometries are randomly rotated. The challenge from rotations results in slightly looser scatters. Still, a strong, disentangled correlation between learned eigen-lengths and human-hypothesized ones can be observed.

C. Additional Experiment Details of Extension Tasks

C.1. Experiment Details of Relative Rotation Estimation

Data   We use the environment point clouds from Tube Passing and Container Fitting, i.e. 1) Tubes as in Tube Passing; 2) multi-category household furniture/appliances from ShapeNet as in Container Fitting. To create input pairs, we sample two random rotations from $R_a, R_b \in SE(3)$ and apply them to the original point cloud P.

Relative Rotation Computation   Given point cloud pair $(R_a P, R_b P)$, we feed both of them to the network to get two sets of unit-length vectors $V^a = (v_1^a, v_2^a, v_3^a), V^b = (v_1^b, v_2^b, v_3^b) \in \mathbb{R}^{3\times3}$. As these vectors are intrinsic to the object and should rotate with the object, we can compute their relative rotation difference and use it as the object's rotation change.
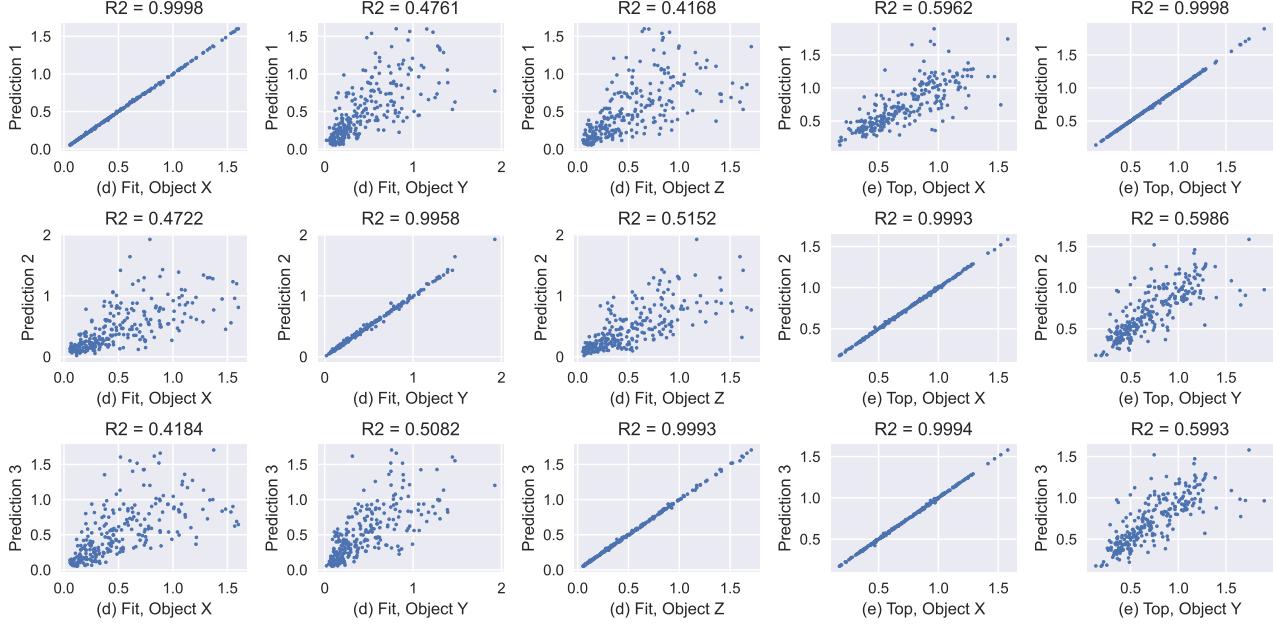
Figure 19. **Full correlation plots and respective** R^2 **values** between ground truth measurements and predicted eigen-lengths from *Eigen-Length-Grounded*.
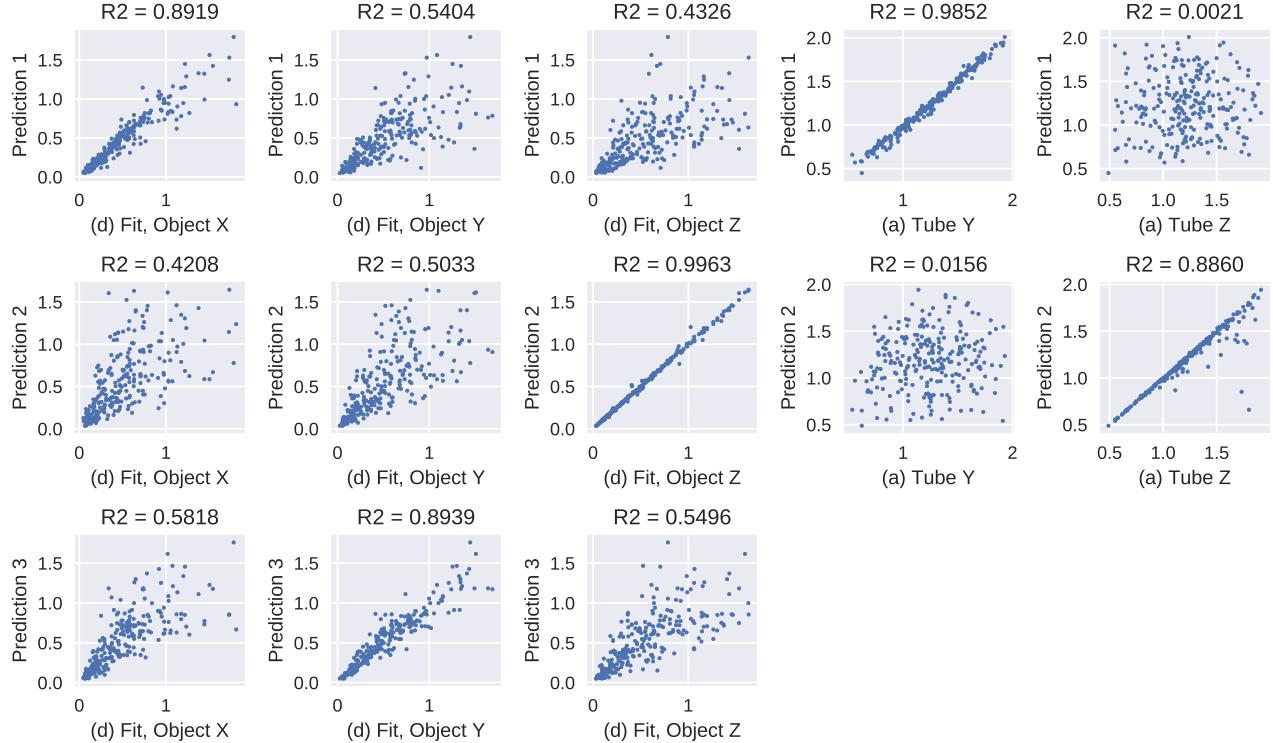
Figure 20. **Full correlation plots and respective** R^2 **values** between human-hypothesized measurements and predicted eigen-lengths in rotated Tube Passing and rotated Container Fitting, respectively. Correspondences between predicted eigen-lengths and human-hypothesized ones can be observed.

Specifically, we use the least-square solution that minimizes $|RV^a - V^b|_2^2$. In practice, we also enumerate all possible matchings between the two sets of vectors and their negatives, e.g. v_1^a can match with $-v_2^b$. For Tubes where we only predict two vectors v_1, v_2, we let $v_3 = v_1 \times v_2$.

C.2. Embodied Visual Navigation

Data   We collect our training data by randomly placing robots of varying sizes in one AI2THOR scene. We record the egocentric depth observation and the robot point cloud, as well as a label indicating whether the robot can move forward by 0.2m by running simulation. We test with depth images taken in another scene. In total, we have collected 6,989 views for training and validation, and 756 views for testing.

Navigation Demo Visualization   We visualize the predicted environment eigen-lengths as a rectangle in front of the robot, which indicates the size of the "hole" or navigable space in the environment. Since learned eigen-lengths can differ from ground truth by a linear transformation, we mapped the raw eigen-length outputs of the network back to the real-world scale for better visualization. The linear mapping coefficients are obtained by fitting a linear model to the predicted robot eigen-lengths and ground truth robot sizes, which are known. Since collisions mostly happen because the environment is too narrow or the robot is too wide, the eigen-length in the horizontal direction is better learned.

The color (green/red) represents the final output (positive/negative) obtained by comparing the eigen-lengths of the robot and the environment. Since collisions mostly happen because the environment is too narrow or the robot is too wide, the eigen-length in the horizontal direction is better learned.

D. Extending AND Clauses to Disjunctive Normal Form (DNF)

D.1. Formulation

We employ the AND clause formulation for all tasks shown in the main paper. Namely, after learning a library of paired object/environment eigen-lengths $\{(L_s^{env}, L_s^{obj})\}_s$, we compose them by

$$\hat{T}(\mathcal{E}, \mathcal{O}) = \bigwedge_{s=1,2,\ldots,S} [L_s^{env}(\mathcal{E}) > L_s^{obj}(\mathcal{O})],$$

(selection mask m is omitted for clarity), approximated by

$$\tilde{T}(\mathcal{E}, \mathcal{O}) = \prod_{s=1,2,\ldots,S} \sigma((L_s^{env}(\mathcal{E}) - L_s^{obj}(\mathcal{O}))/\tau).$$

Here we show we can extend this formulation to the more general Disjunctive Normal Form (DNF), where an OR connects multiple AND clauses. Each AND clause composes eigen-length comparison results of a subset of eigen-lengths. The result of each AND clause is then aggregated by an OR operator. More precisely,

$$\hat{T}(\mathcal{E}, \mathcal{O}) = \bigvee_{U_a \in \mathcal{U}} \bigwedge_{s \in U_a} [L_s^{env}(\mathcal{E}) > L_s^{obj}(\mathcal{O})].$$

$\mathcal{U} = \{U_a\}_a$ specifies the subset U_a of eigen-lengths in each AND clause. We similarly use a differentiable approximation during training:

$$\tilde{T}(\mathcal{E}, \mathcal{O}) = 1 - \prod_{U_a \in \mathcal{U}} (1 - \prod_{s \in U_a} \sigma((L_s^{env}(\mathcal{E}) - L_s^{obj}(\mathcal{O}))/\tau)).$$

The introduction of two-level logic and the OR operator helps express more complex reasoning and deal with a wider range of tasks. For example, many realistic tasks have multiple solutions. OR captures the relationship that the task can be executed if any, not necessarily all, of the solutions work.

D.2. Task and Implementation Details

To demonstrate our framework's compatibility with this new formulation, we experiment with the *Multi-Tube Passing* task. This is a variant of task (a) (*Tube*, or tube passing) in the main paper, where we have two tubes of random sizes placed

next to each other. As long as the object can be translated and passed through any of these tubes, the task is considered as successful.

Similar to tube passing, we randomly sample the extents of the tubes, the shape, scale, and rotation of the object. The center of the two tubes are always at two fixed positions on the y-axis.

We set the number of eigen-lengths to learn as $S = 4$ and split them into two disjoint AND groups, namely $\mathcal{U} = \{\{1, 2\}, \{3, 4\}\}$. Ideally, the learned eigen-lengths should correspond to the height and width of the tubes. Also, the height and width of the same tube should be in the same AND group.

D.3. Result Visualization

Fig. 21 visualizes the learned eigen-lengths, where green and yellow belong to one group, purple and red belong to another group. We successfully learn eigen-lengths that measure along the height/width directions of the tubes. We also learn them in correct groups, where width and height of the same tube are paired together. Numerically, the network achieves a test accuracy of 99.59%.
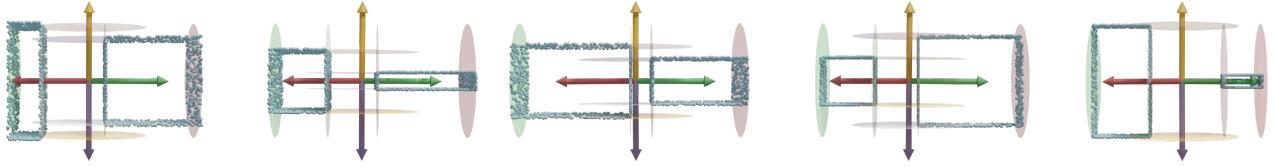
Figure 21. **Visualization of the eigen-lengths learned with OR-AND clauses**. Green and yellow, purple and red eigen-lengths belong to the same AND-group. It turns out that each group attends to one of the tubes and captures its width and height.

E. Discussion and Future Work

E.1. Definition of Eigen-Lengths and Application Scope of the Explored Framework

In our setting, an eigen-length is whatever scalar measurement (i.e., just a 1D scalar) the network invents to best perform its stated downstream task. While this definition for eigen-dimensions is quite general and could be applicable to any object as long as there exist certain 1D eigen-lengths that are crucial and useful for checking the feasibility of accomplishing a downstream task, we are assuming in our current experiments that having such sets of 1D eigen-lengths are *sufficient* for the tasks. Therefore, our current setting would not apply to the tasks where having only such low-dimensional eigen-lengths is not sufficient, such as the tasks of geometric contour matching and object collision checking.

E.2. Broader Implication of the Studied Approach for AI and Robotics

We believe the general approach we suggest can have very general applicability in AI and robotics, where the solution to downstream tasks suggests the emergence of generally useful geometric concepts such as length, height, width, and radius in unsupervised ways. As we described in the introduction, learning such compact useful geometric eigen-lengths is beneficial in the ways that 1) they are highly interpretable, while most of the current learned representations in neural networks are opaque and learned as black-box hidden features which may be unreliable or untrustworthy, 2) they could be shared and reused across different tasks, enabling fast adaptation to novel test-time tasks, and 3) the proposed learning formulation may discover novel yet crucial geometric eigen-lengths that are even unknown to us humans given the new test-time tasks. Furthermore, there could be more geometric concepts of great interest and importance that future work can explore in this direction. Examples can be 1) symmetry, as a result of trying to complete 3D shapes, 2) regular object arrangements and poses as a tool for efficient search, and 3) tracking, as an essential capability for predicting the outcome of sports games. In other words, we want learning networks to invent the notions so symmetry, regularity, or tracking. If such capabilities could emerge from purely unsupervised learning, we no longer need to rely on black-box-like neural networks and human annotations for this geometric information over 3D objects.

E.3. Determining the Number of Eigen-Lengths to Learn

The number of eigen-lengths to learn, i.e. S, is a hyperparameter of our learning framework and has to be set in advance. However, it should be interpreted as the upper bound on the number of eigen-lengths the system can learn, and does not have to be the "groundtruth" number of relevant eigen-lengths. As shown in Sec. 4.3 and Sec. 5.2, when we set $S = 3$ for the countertop fitting task where only two eigen-lengths matter, the extra "slot" either degenerates or coincides with other slots. Such cases can be easily detected and filtered, and the actual number of relevant eigen-lengths can be discovered. Setting a maximum number for an unknown number of targets is also a common practice in problems like object detection (Redmon & Farhadi, 2017). That being said, a more flexible mechanism that allows an arbitrary number of eigen-lengths would be desirable, especially for objects with complex compositional structures like robotic arms or closets with many drawers. We leave this as a future direction.