Form2Fit: Learning Shape Priors for Generalizable Kit Assembly from Disassembly

Kevin Zakka[1] **Andy Zeng**[1,2] **Johnny Lee**[1] **Shuran Song**[1,3]
[1]Google [2]Princeton University [3]Columbia University

Abstract: Is it possible to learn policies for robotic assembly that can generalize to new objects? We explore this idea in the context of the kit assembly task (*i.e.*, placing object(s) into a blister pack or corrugated display to form a single unit). Since classic methods rely heavily on object pose estimation, they often struggle to generalize to new objects without 3D CAD models or task-specific training data. In this work, we propose to formulate the kit assembly task as a shape matching problem, where the goal is to learn a shape descriptor that establishes geometric correspondences between object surfaces and their target placement locations (*e.g.* empty holes/cavities in their respective kits) from visual input. This formulation enables the model to acquire a broader understanding of how shapes and surfaces fit together for assembly – allowing it to generalize to new objects and kits. To obtain training data for our model, we present a self-supervised data-collection pipeline that obtains ground truth object-to-placement correspondences by disassembling complete kits. Our resulting real-world system, **Form2Fit**, learns effective pick and place strategies for assembling objects into a variety of kits – achieving 90% average success rates under different initial conditions (*e.g.* varying object and kit poses), 94% success under new configurations of multiple kits, and over 86% success with completely new objects and kits.

Keywords: assembly, geometry, deep learning

1 Introduction

Across many assembly tasks, the shape of an object can often inform how it should be fitted with other parts. For example, in kit assembly (*i.e.*, placing object(s) into a blister pack or corrugated display to form a single unit – see examples in Fig. 1), the profile of an object likely matches the silhouette of the cavity in the paperboard packaging (*i.e.*, kit) that it should be placed into.

While these low-level geometric signals can provide useful cues for both perception and planning, they are often overlooked in many modern assembly methods, which typically abstract visual observations into 6D object poses and then plan on top of the inferred poses. By relying heavily on accurate pose information, these algorithms remain unable to generalize to new objects and kits without task-specific training data and cost functions. As a result, these systems also struggle to quickly scale up to the subclass of real-world assembly lines that may see new kits every two weeks (*e.g.* due to seasonal items and packages).

In this work, we explore the following question: if we formulate the kit assembly task as a shape matching problem, is it possible to learn policies that can generalize to new objects and kits? To this end, we propose **Form2Fit**, an end-to-end pick-and-place formulation for kit assembly that leverages shape priors for generalization. There are two key aspects to our system:

- **Shape-driven assembly for generalization**. We establish geometric correspondences between object surfaces and their target placement locations (*e.g.* empty holes/cavities) by learning a fully convolutional network that maps from visual observations of a scene to dense pixel-wise feature descriptors. During training, the descriptors are supervised in a Siamese fashion and regularized so that they are more similar for ground truth object-to-placement correspondences. The key idea is that as the network trains over a variety of objects and target locations across multiple kitting tasks, it acquires a broader understanding of how shapes and surfaces fit together for assembly – subsequently learning a more generalizable descriptor that is capable of matching new objects and target locations unseen during training.

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

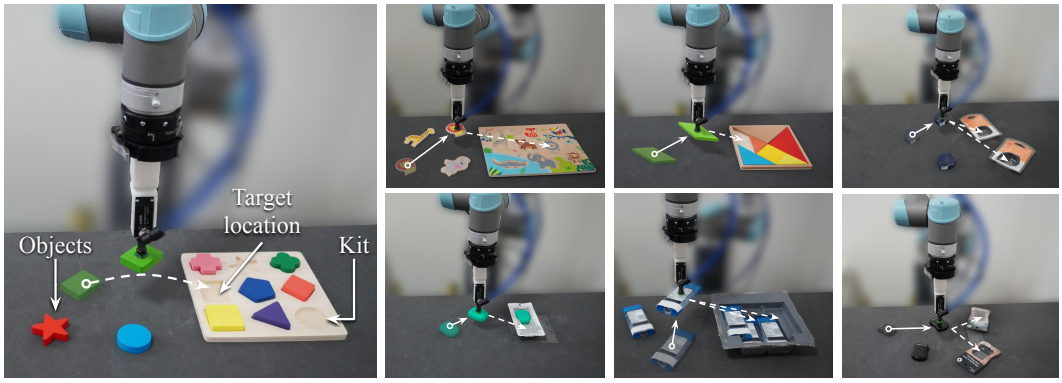

Figure 1: **Form2Fit** learns to assemble a wide variety of kits by finding geometric correspondences between object surfaces and their target placement locations. By leveraging data-driven shape priors learned from multiple kits during training, the system generalizes to new objects and kits.

- **Learning assembly from disassembly.** We present a self-supervised data-collection pipeline that obtains training data for assembly (*i.e.*, ground truth motion trajectories and correspondences between objects and target placements) by disassembling completed kits. Classic methods of obtaining training data for assembly (*e.g.* via human teleoperated demonstrations or scripted policies) are often time-consuming and expensive. However, we show that for kit assembly, it is possible to autonomously acquire large amounts of high-quality assembling data by disassembling kits through trial and error with pick and place, then rewinding the action sequences over time.

This formulation enables our system to assemble a wide variety of kits under different initial conditions (*e.g.* different rotations and translations) with accuracies of 90% (measured by the rate at which an object is placed in its target kit with the correct configuration), and generalizes to new settings with mixtures of multiple kits as well as entirely new objects and kits.

The primary contribution of this paper is to provide new perspectives on the robot assembly task: in particular, we study the extent to which we can achieve generalizable kit assembly by formulating the task as a shape matching problem. We also demonstrate that it is possible to acquire substantial amounts of training data for kit assembly by reversing the action sequences collected from self-supervised disassembly. We provide extensive experiments in real settings to evaluate the key components of our system. In this work, we also discuss some extensions of our formulation, as well as its practical limitations. Videos of our system are available in the supplementary material.

2 Related Work

Object pose estimation for assembly. Classic methods for assembly are often characterized by a perception module that first estimates the 6D object poses of observed parts [1, 2, 3, 4, 5], followed by a planner that then optimizes for picking and placing actions based on the inferred object poses and task-informed goal states. For example, Choi *et al.* [6] introduces a voting-based algorithm to estimate the poses of component parts for assembly from 3D sensor data. Litvak *et al.* [7] leverages simulated depth images to amass the training data for pose estimation. Jorg *et al.* [8] uses a multi-sensory approach with both vision and force-torque sensing to improve the accuracy of engine assembly. While these methods have seen great success in highly structured environments, they require full knowledge of all object parts (*e.g.* with high-quality 3D object models) and/or substantial task-specific training data and task definitions (which requires manual tuning). This limits their practical applicability to kit assembly lines in retail logistics or manufacturing, which can be exposed to new object and kits as frequently as every two weeks. Handling such high task-level variation requires assembly algorithms that can quickly scale or adapt to new objects, which is the focus of our formulation, Form2Fit.

Reinforcement learning for assembly. Another line of work focuses on learning policies for assembly tasks to replace classic optimization-based or rule-based planners. To simplify the task, these works often assume full state knowledge (*i.e.*, object and robot poses). For example Popov *et al.* [9] tackles the task of Lego brick stitching, where the state of the environment including brick positions are provided by the simulation environment. Thomas *et al.* [10] learns a variety of robotic assembly tasks from CAD models, where the state of each object part is detected using QR codes.

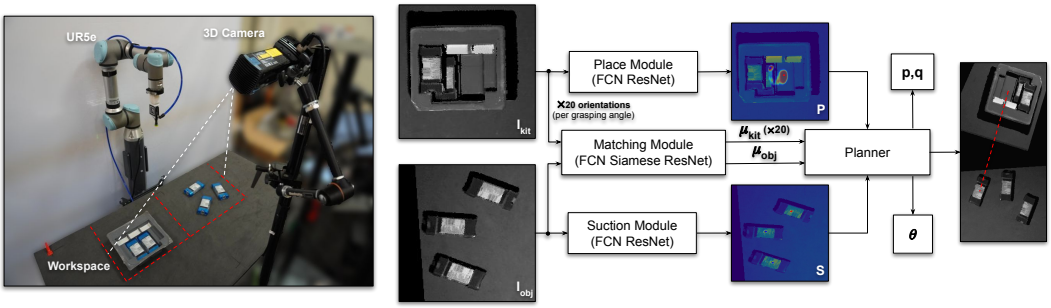

Figure 2: **Overview.** Kit and object heightmaps are generated from a visual observation of the workspace. The kit heightmap I_{kit} is fed to the place module to predict a dense probability map of place success P, while the object heightmap I_{obj} is fed to the suction module to predict a dense probability map of suction success S. In parallel, the matching module ingests both kit and object heightmaps to produce kit and object descriptor maps μ_{kit} and μ_{obj} which are fed along with P and S to the planning module. The result is a picking location p, placing location q, and angle θ encoding the end-effector rotation about the z-axis.

More recent works [11, 12] eliminate the need for accurate state estimation by learning a policy that directly maps from raw pixel observations to actions through reinforcement learning. However, these end-to-end models require large amounts of training data and remain difficult to generalize to new scenarios (train and test cases are often very similar with the same set of objects). In contrast our system is able to learn effective assembly policies with a much smaller amount of data (500 disassembly sequences), and generalizes well to different object types and scene configurations (different pose and number of objects) without extensive fine-tuning.

Learning shape correspondences. Learning visual and shape correspondences is a fundamental task in vision and graphics – studied extensively in prior work via descriptors [13, 14, 15, 16, 17, 18]. These descriptors are either designed or trained to match between points of similar geometry across different meshes or from different viewpoints. Hence, rotation invariance is a desired property from these descriptors. In contrast to prior work, our goal is to learn a general matching function between objects and their target placements locations (*i.e.*, holes/cavities in the kits) instead of matching between two similar shapes. We also want the descriptor to be "rotation-sensitive", so that the shape matching result can inform the actions necessary for successful assembly.

Learning from reversing time. Time-reversal is a classic trick in computer vision for learning information from sequences of visual data (*i.e.*, videos). For example, many work [19, 20, 21, 22] study how predicting the arrow of time or the order of frames in a video can be used to learn image representations and feature embedding. Nair *et al.* [12] uses time-reversal as supervision for video prediction to learn effective visual foresight policies. While many of these methods use time-reversal as a means to extract additional information from the order of frames, we instead use time-reversal as a way to generate correspondence labels from pick and place. We observed that a pick and place sequence for disassembly, when reversed, can serve as a valid sequence for assembly. This is not true for all assembly tasks (particularly when more complex dynamics are involved), but this observation holds true for a substantial number of kit assembly tasks.

3 Method Overview

Form2Fit takes as input a visual observation I of the workspace (including objects and kits), and outputs a prediction of three parameters: a picking location p, a placing location q, and an angle θ that defines a change in orientation between the picking and placing locations. These parameters are used with motion primitives on the robot to execute a respective pick, orient, and place operation. Our learning objective is to optimize our predictions of p, q, θ such that after each physical execution, an object is thereby correctly placed into its target location and orientation in its corresponding kit. In this work, each prediction of p, q, θ is *i.i.d.* and conditioned only on the visual input – which is sufficient for sequential planning in kit assembly (see Sec. 3.3).

The system consists of three network modules: 1) a suction network that outputs dense pixel-wise predictions of picking success probabilities (*i.e.*, affordances) using a suction cup, 2) a placing net-

work that outputs dense pixel-wise predictions of placing success probabilities on the kit, and 3) a matching network that outputs dense pixel-wise rotation-sensitive feature descriptors to match between objects and their corresponding placement locations in the kit. We use the placement network directly to infer q, then use the suction and matching networks together to infer both p and θ (see Sec. 3.5). Fig. 2 illustrates an overview of our approach.

Our system is trained through self-supervision from time-reversed disassembly: randomly picking with trial and error to disassemble from a fully-assembled kit, then reversing the disassembly sequence to obtain training labels for the suction, placing, and matching networks. In the following subsections, we provide an overview of each module, then describe the details of the data collection process and training procedure.

Hardware setup. Our system (shown in Fig. 2) consists of a 6DoF UR5 robot with a 3D printed suction end-effector, and a Photoneo PhoXi camera that provides high quality grayscale and depth images. The camera is localized with respect to the robot base using the calibration pipeline in [23].

3.1 Visual Representation

We represent the visual observation of the workspace as a grayscale-depth heightmap image of the workspace. To compute this heightmap, we capture intensity and depth images from a statically-mounted camera, project the data onto a 3D point cloud, and orthographically back-project upwards in the direction of gravity to construct a 2-channel heightmap image representation I with both grayscale (intensity) and height-from-bottom (depth) channels concatenated. Both grayscale and depth heightmaps are normalized by their respective dataset mean and standard deviation.

The workspace covers a $1.344\text{m} \times 0.672\text{m}$ tabletop surface. The heightmap has a pixel resolution of 360×464 with a spatial resolution of 0.002m per pixel. We constrain the kit to lie within the left half of the workspace and the objects within the right half. This separation allows us to split the heightmap image into two halves: I_{kit} containing the kit, and I_{obj} containing the objects, each with a pixel resolution of 360×232.

3.2 Suction Module: Learning Where to Pick

The suction module uses a deep network that takes as input the heightmap image I, and predicts favorable picking locations using a suction primitive on the objects inside the kit (during disassembly) and outside the kit (during assembly). Stable suction points often correspond to flat, nonporous surfaces near an object's center of mass.

Suction primitive. The suction primitive takes as input a 3D location and executes a top-down suction-based grasp centered at that location. This primitive executes in an open loop fashion using stable, collision-free IK solves [24].

Network architecture. The suction network is a fully convolutional 8-layer dilated residual network [25, 26, 27], interleaved with 2 max-pooling layers and 2 spatial bilinear upsampling layers respectively. The network takes as input a grayscale-depth heightmap I and outputs the probability of picking success using the suction primitive across a dense pixel-wise sampling of end-effector locations over I. More specifically, the network outputs a suction confidence map S with the same size and resolution as that of the input I. Each pixel $s_i \in S$ represents the predicted probability of suction success (*i.e.*, suction affordance) when the suction primitive is executed at the 3D surface location (inferred from calibration) of the corresponding pixel $i \in I$.

3.3 Placing Module: Learning Ordered Placing

For certain kit assembly tasks, there may be sequence-level constraints that define the order in which objects should be placed into the kit. For example, to correctly assemble a five-pack kit of deodorants (see this test case in Fig. 3), the bottom layer must be filled with deodorants before the robot can proceed to fill the top layer.

To enable sequential ordering, our system uses a placing module that predicts the next best placing location conditioned on the current state (*i.e.*, visual observation I) of the environment. The placing module consists of a fully convolutional network (same architecture as suction network) which takes as input the kit heightmap I_{kit} and outputs a dense pixel-wise prediction P of placing confidence values over I. The 3D locations of the pixels with higher confidence serve as better locations for the suction gripper to approach from a top-down angle while holding an object. In Fig. 3, the placing

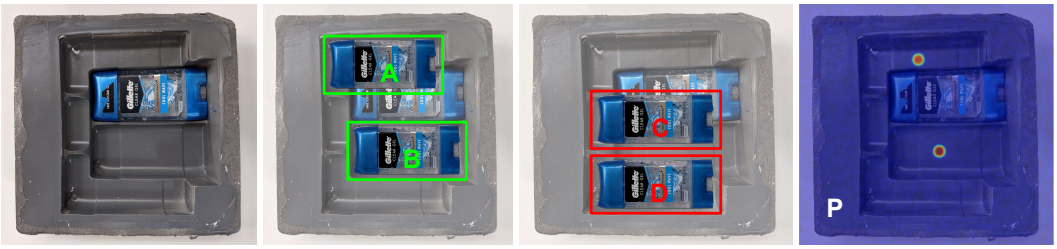

Figure 3: **An example of ordered assembly** for the deodorant kit. At timestep $t = 1$, locations A and B are valid place positions, while C and D are examples of invalid locations. The placing network output P reflects this ordering constraint, displaying hotter sampling regions at A and B compared to C and D.

module would predict higher confidence values at locations A and B with lower confidence values at locations C and D.

3.4 Matching Module: Learning How to Pick and Place with Shape Matching

While the suction and placing modules provide a list of candidate picking and placing locations, the system requires a third module to 1) *associate* each suction location to a corresponding placing location in the kit and 2) infer the change in object orientation. This matching module serves as the core of our algorithm, which learns dense pixel-wise *orientation-sensitive* correspondences between the objects on the table and their placement locations in the kit.

Network architecture. The matching module consists of a two-stream Siamese network [28], where each stream is a fully convolutional residual network (same architecture as suction network) with shared weights across streams. Its goal is to learn a function f that maps each pixel of the observed kit and objects in the heightmap I to a d-dimensional descriptor space ($d = 64$ in our experiments), where closer feature distances indicate better object-to-placement correspondences, *i.e.*, $f : I \in \mathbb{R}^{H\times W\times 2} \rightarrow \mathbb{R}^{H\times W\times d}$.

The first stream of the network maps the object heightmap I_{obj} to a dense object descriptor map μ_{obj}. The second stream maps a batch of kit heightmaps I_{kit} with 20 different orientations (*i.e.*, multiples of 18°), to 20 kit descriptor maps μ_{kit}. Each pixel $i \in I_{\text{kit}}$ in the kit heightmap maps to 20 kit descriptors (one for each rotation), but only one of them (the most similar) will match to its corresponding object descriptor in μ_{obj}. The index of the rotation with the most similar kit descriptor informs the change in object orientation θ between the picking and placing, *i.e.*, $\theta = \arg\min_j \|\mu^j_{\text{kit}} - \mu^i_{\text{obj}}\|^2_2$ where $j \in \mathbb{Z} : j \in [1, 20]$ and $i \in \mathbb{Z} : i \in [1, H \times W]$. In this way, the matching network not only establishes correspondences between object picking and kit placement locations, but also infers the change in object orientation θ between the pick and place.

Training is done using a pixel-wise contrastive loss, where for every pair of kit and object heightmaps $(I_{\text{kit}}, I_{\text{obj}})$, we sample non-matches from I_{obj} and all 20 rotations of I_{kit} and matches from I_{obj} but only the rotation j of I_{kit} corresponding to the ground-truth angle. The loss function thus encourages descriptors to match solely at the correct rotation of the kit image while non-matches are pushed to be at least a feature distance margin M apart. At each training iteration, we sample match and non-matches with a 1:5 ratio and compute the loss: $\mathcal{L} = \mathcal{L}_{\text{match}} + \mathcal{L}_{\text{nonmatch}}$ where $\mathcal{L}_{\text{match}}(I_{\text{kit}}, I_{\text{obj}}) = \frac{1}{N_{\text{match}}} \sum_{N_{\text{match}}} \|\mu^j_{\text{kit},i} - \mu_{\text{obj},i}\|^2_2$ and $\mathcal{L}_{\text{nonmatch}}(I_{\text{kit}}, I_{\text{obj}}) = \frac{1}{N_{\text{match}}} \sum_{j\in\mathbb{Z}:j\in[1,20]} \sum_{N_{\text{nonmatch}}} \max\left(0, M - \|\mu^j_{\text{kit},i} - \mu_{\text{obj},i}\|_2\right)^2$.

3.5 Planner

The planner is responsible for integrating information from all three modules and producing the final assembly parameters p, q and θ. Specifically, top-k pick candidates are sampled from the suction module output S and top-k place candidates are sampled across all 20 rotations of the place module output P. Then, for each pick-and-place pair in the Cartesian product of candidates, kit and object descriptors are indexed and their L2 distance is evaluated after which the pair with the lowest L2 distance across all rotations and all candidates is chosen to produce the final kit descriptor, object descriptor and rotation index.

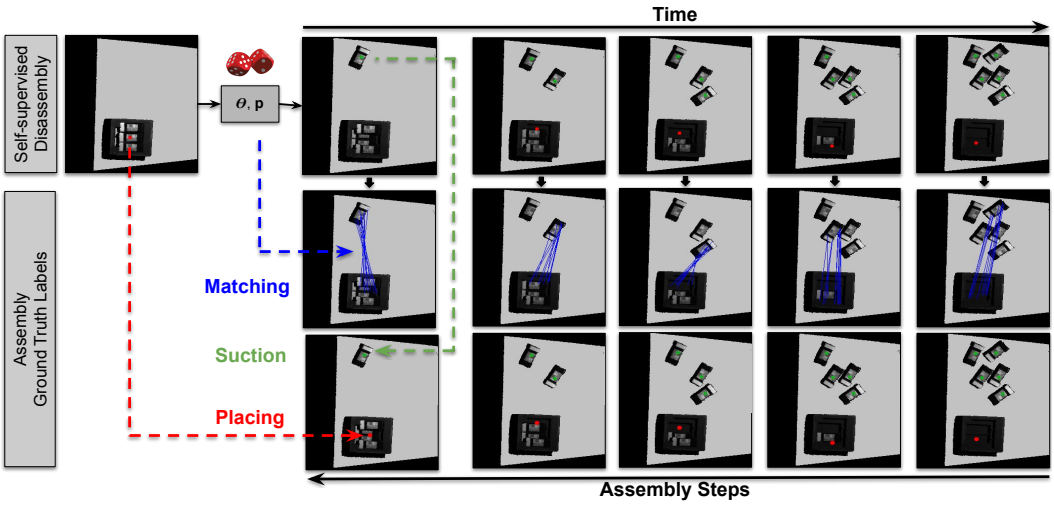

Figure 4: **Self-supervised time reversal** enables us to efficiently generate ground-truth labels from disassembly data. Concretely, at every timestep (*i.e.*, columns), disassembly trajectories are executed by querying the suction network (*i.e.*, red) and randomly generating placing poses (*i.e.*, green). By capturing the state of the environment before and after the trajectory, the recorded information can be reversed to produce assembly data: the place point is used as suction training data, the suction point is used as placement data and the robot motion (θ, p) allows us to create dense pixel-wise correspondences between the object and its target location in the kit.

Calibrated camera parameters are then used to convert descriptor pixels into world coordinates, *i.e.*, the kit descriptor is converted to q, the object descriptor to p, and the rotation index to an end-effector rotation about the z-axis θ. Because the Cartesian product of candidate pairs grows exponentially with the number of candidates, a tradeoff between planner speed and accuracy can be controlled through the sampling parameter. In our experiments, this translates to an execution time that can range from just a few milliseconds to up to 20 seconds with dense sampling.

4 Automatic Data Collection via Disassembly

To generate the inputs and ground-truth labels needed to train our various networks, we create a self-resetting closed-loop system wherein the robot continuously generates a random sequence of disassembly trajectories $S = \{T_1, T_2, \ldots, T_N\}$ to empty a kit of N objects, then performs it in reverse $S' = \{T_N, T_{N-1}, \ldots, T_1\}$ to reset the system to its initial state. Fig. 4 show the pipeline of the disassembly data collection process.

Specifically, for every object in the kit, a trajectory T is generated as follows: first, the robot captures a grayscale-depth image to construct kit and object heightmaps I_{kit} and I_{obj}, then it performs a forward pass of the suction network to make a prediction of parameter p which is executed by the suction primitive to grasp the object. If the suction action is successful, it places the object at a position q and rotation θ sampled uniformly at random in the bounds of the workspace.

If the suction action is not successful (*i.e.*, no object gets picked up), this suction point is labeled as negative for the online learning process. The suction success signal is obtained by visual background subtraction, and the suction air flow. We tape the kit to the table to prevent failed grasps from causing accidental kit displacements and to ease this trial-and-error process.

All the parameters (*e.g.* p, q, θ) are stored for resetting the scene. Once all N objects have been disassembled, the robot indexes the trajectories in reverse, suctioning the objects using the place parameters and placing them back into the kit using the suction parameters. For each trajectory, it stores the grayscale-depth heightmaps captured before and after the disassembly step, the predicted suction pose, and the randomly generated place pose. To bootstrap the learning, we used a pretrained suction network with 50 manually labelled suction examples. For each kit in the training set, we collect 500 disassembly sequences, which takes 8-10 hours of wall clock time.

Place Network Dataset. To generate the training data for the placing network, we use the suction location p at time t and the heightmap I_{kit} at time $t+1$ (*i.e.*, image taken after the suction action) as one training pair. Thus, the placing network is encouraged to look at empty locations in the kit and predict valid place positions for the next object.

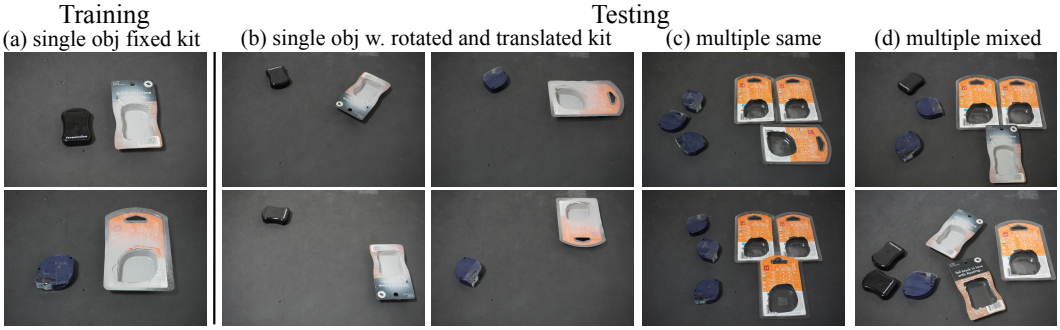

Figure 5: Generalization to novel kit configurations with multiples and mixtures.

Configuration	one black-floss	one tape	three tape	2 tape and 2 black-floss
Success Rate	0.95	0.90	1.00, 0.95, 0.85	0.80, 1.00, 1.00, 0.95

Table 2: Model trained on 2 individual kits tested on combinations of identical and mixtures of kits.

Suction Network Dataset. The training data for the suction network consists of two sets of input-label pairs: (1) the kit heightmap and the suction position (I_{kit}, p) and (2) the object heightmap and the place position (I_{obj}, q). Thus, the suction network is encouraged to predict favorable suction locations on the object of interest inside *and* outside the kit.

Matching Network Dataset. To label the correspondences for the matching network, we first compute the masks of the object (both inside and outside the kit) using the images' difference. The relationship of every pixel in the cavity of the kit and its corresponding pixel on the object outside the kit is calculated using the rotation angle θ, *i.e.*, $[u_{obj}, v_{obj}] = R_{z,\theta}[u_{kit}, v_{kit}, 1]$ where $R_{z,\theta}$ is a 2×3 rotation matrix defining a rotation of θ° around the z-axis.

5 Experiments

We design a series of experiments to evaluate the effectiveness of our approach in various assembly settings. In particular, we would like to examine the following questions: (1) How accurate and robust is our system across a wide range of rotations and translations of the objects and kit? (2) Is our system capable of generalizing to new kit configurations such as multiple versions of the same kit and mixtures of different kits when trained solely on individual kits? (3) Does our system learn descriptors that can generalize to previously unseen objects and kits?

Kit Type	Name	Assembly Success
single object kits	tape-runner	0.90
	black-floss	0.95
multi-object kits	fruits	0.65, 0.60, 0.95, 0.90
	zoo-animals	0.90, 0.95, 0.95, 0.95, 0.85
	deodorants	1.00, 1.00, 1.00, 0.90, 0.85

Table 1: Assembly success rate on different kits.

Evaluation metric. To answer these questions, we execute a series of tests in which the system is tasked with consecutively assembling various objects into their respective kits. For each test, we record the average assembly accuracy $\bar{s}$, defined as the percentage of attempts where the objects are successfully placed into their target locations. For the kits that contain multiple objects (*e.g.* deodorants, zoo animals, fruits), we record the average individual success rate of each object and average over all objects to compute the overall kit success rate, *i.e.*, $\bar{s} = \sum_{n=1}^{N} \sum_{i=1}^{20} s_{ni}$.

Generalization to initial conditions. In this first series of tests, we set out to measure the robustness of our system to varying initial conditions. For each kit, during training, the kit is fixed in the same position and orientation (Fig. 5 (a)) while during testing, we randomly position and orient it on the workspace (Fig. 5 (b)). Specifically, we record assembly accuracy on 5 random kit poses, 4 times each for a total of $N = 20$ trials. Note that for kits with multiple objects, if the execution of an object fails, we intervene and place it in its correct location to allow the system to resume. As seen in Table 1, our system achieves 90% average assembly success on both single and multi-object kits. We observe that frequent modes of failure come from the robot placing objects (*e.g.* floss and tape) 180° flipped from their correct orientation.

Generalization to multiple kits. In the next series of tests, we study how well our system can generalize to different kit configurations. During training, the system sees only 2 individual kits (Fig. 5 (a)), while during testing, we create combinations of the same kit and mixtures of kits (Fig.

Figure 6: Our system is able to generalize to new kit configurations and object geometries.

5 (c) and (d)). Similarly to above, we perform $5 \times 4 = 20$ trials. While our system has never been trained on these novel settings, it is able to achieve an assembly success rate of 94.27% Table 2.

Generalization to novel kits. Finally, we study how well our system can generalize to novel kits that are unseen during training. Specifically, the testing kits are both single-object kits and multi-object kits with square, rectangular and circular shapes as seen in Fig. 6. For perfectly symmetrical objects (*e.g.* circles), we consider the assembly to be successful as long as the object is placed into kit, otherwise we count it as a failure. On a set of 20 trials, our system is able to achieve over 86% generalization accuracy on these novel kits, with mistakes mostly 180°rotational errors.

Feature visualization. Fig. 7 shows the t-SNE embedding [29] of the learned feature descriptors for different kits. From the visualization, we can observe that the descriptors have learned to encode: (a) rotation: objects oriented differently have different descriptors (A, C, D, E) and (H, F), (b) spatial correspondence: same points on the same oriented objects share similar descriptors (A, B) and (F, G), and (c) object identity: zoo animals and fruits exhibit unique descriptors (columns 3 and 4).

Limitations and failure cases. While our system presents a step towards generalizable kit assembly, it also has a few limitations. First, since our system relies substantially on geometry, it often makes mistakes for objects with symmetrical shapes, placing them with 180° rotational errors. This occurs more frequently for new objects unseen during training. Second, our system only handles 2D rotations (*i.e.*, planar object rotations) and assumes that objects are face-down. To extend this work, it would be interesting to explore a more complex (*e.g.* higher DoF) action representation for 3D assembly. Finally, while our system is able to handle partially transparent kits (*e.g.* the performance for transparent and spray-painted tape-runner and floss kits is similar), it has trouble handling fully transparent kits like the deodorant blister pack (which we spray-paint to support stereo matching for our 3D camera). As future work, it would be interesting to use external vision algorithms [30, 31, 32, 33] to estimate the geometry of the transparent kits before using the visual data.

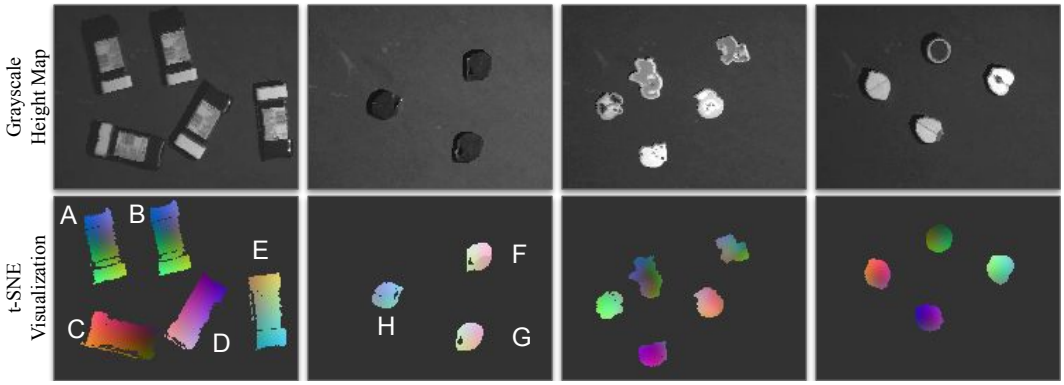

Figure 7: The t-SNE embedding visualizations of object descriptors for different kits show that the descriptors have learned to encode rotation (A, B, C), spatial correspondences (identical points on A and B share similar descriptors) and object identity (all zoo animals have unique descriptors).

6 Conclusion

We present Form2Fit, a framework for generalizable kit assembly. By formulating the assembly task as a shape matching problem, our method learns a general matching function that establishes geometric correspondences between objects and their kit placement locations from visual input, without the need for 3D CAD models. The system is self-supervised – obtaining its own training labels for assembly by disassembling kits through trial and error with pick and place, then rewinding the action sequences over time. Our experiments demonstrate that our system is robust to a variety of initial conditions, handles new kit combinations, and generalizes to new objects and kits.

Acknowledgments

We would like to thank Nick Hynes, Alex Nichol, and Ivan Krasin for fruitful technical discussions, Adrian Wong, Brandon Hurd, Julian Salazar, and Sean Snyder for hardware support, and Ryan Hickman for valuable managerial support. We are also grateful for hardware and financial support from Google.

References

[1] M. Braun, Q. Rao, Y. Wang, and F. Flohr. Pose-rcnn: Joint object detection and pose estimation using 3d object proposals. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 1546–1551. IEEE, 2016.

[2] A. Collet, M. Martinez, and S. S. Srinivasa. The moped framework: Object recognition and pose estimation for manipulation. *IJRR*, 30(10):1284–1306, 2011.

[3] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *ICCV*, 2011.

[4] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. *arXiv preprint arXiv:1901.02970*, 2019.

[5] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker Jr, A. Rodriguez, and J. Xiao. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In *ICRA*, 2017.

[6] C. Choi, Y. Taguchi, O. Tuzel, M.-Y. Liu, and S. Ramalingam. Voting-based pose estimation for robotic assembly using a 3D sensor. In *2012 IEEE International Conference on Robotics and Automation*, pages 1724–1731. IEEE, 2012.

[7] Y. Litvak, A. Biess, and A. Bar-Hillel. Learning pose estimation for high-precision robotic assembly using simulated depth images, 2018.

[8] S. Jorg, J. Langwald, J. Stelter, G. Hirzinger, and C. Natale. Flexible robot-assembly using a multi-sensory approach. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 4, pages 3687–3694. IEEE, 2000.

[9] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation, 2017.

[10] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, and P. Abbeel. Learning robotic assembly from cad. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[11] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[12] S. Nair, M. Babaeizadeh, C. Finn, S. Levine, and V. Kumar. Time reversal as self-supervision. *arXiv preprint arXiv:1810.01128*, 2018.

[13] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1802–1811, 2017.

[14] P. R. Florence, L. Manuelli, and R. Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *CoRL*, 2018.

[15] T. Schmidt, R. Newcombe, and D. Fox. Self-supervised visual descriptor learning for dense correspondence. *IEEE Robotics and Automation Letters*, 2(2):420–427, 2016.

[16] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433–449, 1999.

[17] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *Computer Vision-ECCV 2004*, pages 224–237. Springer, 2004.

[18] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3384–3391. IEEE, 2008.

[19] D. Wei, J. J. Lim, A. Zisserman, and W. T. Freeman. Learning and using the arrow of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8052–8060, 2018.

[20] V. Ramanathan, K. Tang, G. Mori, and L. Fei-Fei. Learning temporal embeddings for complex video analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4471–4479, 2015.

[21] B. Fernando, H. Bilen, E. Gavves, and S. Gould. Self-supervised video representation learning with odd-one-out networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3636–3645, 2017.

[22] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016.

[23] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4238–4245. IEEE, 2018.

[24] R. Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute.

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[26] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[27] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 472–480, 2017.

[28] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.

[29] L. Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[30] Y. Xu, H. Nagahara, A. Shimada, and R.-i. Taniguchi. Transcut: Transparent object segmentation from a light-field image. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[31] C. Guo-Hua, W. Jun-Yi, and Z. Ai-Jun. Transparent object detection and location based on rgb-d camera. In *Journal of Physics: Conference Series*, volume 1183, page 012011. IOP Publishing, 2019.

[32] G. Chen, K. Han, and K.-Y. K. Wong. Tom-net: Learning transparent object matting from a single image. In *CVPR*, 2018.

[33] Y. Ji, Q. Xia, and Z. Zhang. Fusing depth and silhouette for scanning transparent object with rgb-d sensor. *International Journal of Optics*, 2017, 2017.