Self-Supervised Goal-Conditioned Pick and Place

Coline Devin
UC Berkeley
coline@eecs.berkeley.edu

Payam Rowghanian
Osaro Inc.
payam@osaro.com

Chris Vigorito
Osaro Inc.
chris@osaro.com

Will Richards
Osaro Inc.
will@osaro.com

Khashayar Rohanimanesh
Osaro Inc.
khash@osaro.com

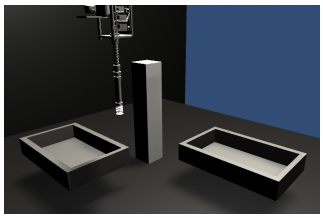
***Abstract*—Robots have the capability to collect large amounts of data autonomously by interacting with objects in the world. However, it is often not obvious *how* to learning from autonomously collected data without human-labeled supervision. In this work we learn pixel-wise object representations from unsupervised pick and place data that generalize to new objects. We introduce a novel framework for using these representations in order to predict where to pick and where to place in order to match a goal image. Finally, we demonstrate the utility of our approach in a simulated grasping environment.**

I. Introduction

Industrial robotics uses, such as piece-picking and kitting, need robots to be able to manipulate diverse objects. Human-supervised learning based-approaches for detecting objects have shown promise in robotics. However, given the immense variety of objects in the world, requiring human-annotated segmentation data or CAD models for all objects a robot may face is not a scalable approach. Learning visual representations of objects *without* human annotations could be cheaper and provide faster adaptability to novel objects. Robots offer promising avenues for self supervised learning as they interact with the world and record data. By taking actions and changing the state of their environment, robots generate supervision signals without relying on human annotators.

Self-supervised learning has been approached from various directions in the context of robotics. Time-contrastive methods use multiple viewpoints or simulations of a trajectory to learn features over the progress of task [10, 8, 6, 1]. Another approach is to move objects around in order to learn about their visual appearance [3, 9, 11].

We use contrastive learning to learn object-embeddings from unsupervised grasping data in order to perform goal-conditioned grasping and placing. Several prior approaches have shown that robotic grasping data can be used to learn object embeddings useful for goal-conditioned grasping [5] and assembly [12]. These approaches learn useful high dimensional embeddings of object images by moving objects with a robot. For example, by grasping an object the robot has generated two views of an object: the view of the object in the workspace, and the view in the gripper. Contrastive learning [4] can then be used to push the embeddings of these two views closer together and the embeddings of other points farther away. We differ from prior work by learning a unified embedding space to solve both grasping and placing simultaneously without relying on assembly priors. One appealing aspect of this framework is that it does not need an additional data collection step specific to this problem beyond normal pick and place datasets which are widely generated as more robots are employed in standard pick and place scenarios across many warehouse environments.



(a) Robot cell

(b) Gripper image, $(x, y)_w$ is the center pixel. I_{wrist}

(c) Grasp bin I_{grasp} with grasp point $(x, y)_g$ shown in red.

(d) Place bin I_{place} with place point $(x, y)_p$ shown in red.

Fig. 1: Robot cell for grasping and placing objects.

Our contributions can be summarized as follows: (1) we present an end-to-end self-supervised framework for learning grasp embeddings that predict grasp and place poses conditioned on a goal image; (2) we present results in a simulated domain and show that our model generalizes to unseen objects.

II. Learning Pixel-wise Object Embeddings without Supervision

We focus on grasping with suction cups, which provides pixel-level supervision about where an object was grasped and placed. We aim to learn embeddings from grasping data that can be used for deciding both where to grasp and where to place given an image of a goal.

As illustrated in Figure 1, our environment consists of a robotic arm, a source bin, and a target bin (Figure 1a). There are three cameras mounted: fixed overhead cameras for each bin (Figures 1c and 1d), and also a wrist-cam mounted on the end-effector that captures an image of the grasped object (Figures 1b). An episode of the manipulation task consists of picking an object from the grasp bin conditioned on the placement goal and placing it in the place bin. In this environment, we collect a dataset of an image of the grasp bin before grasping I_{grasp}, an image of the place bin after

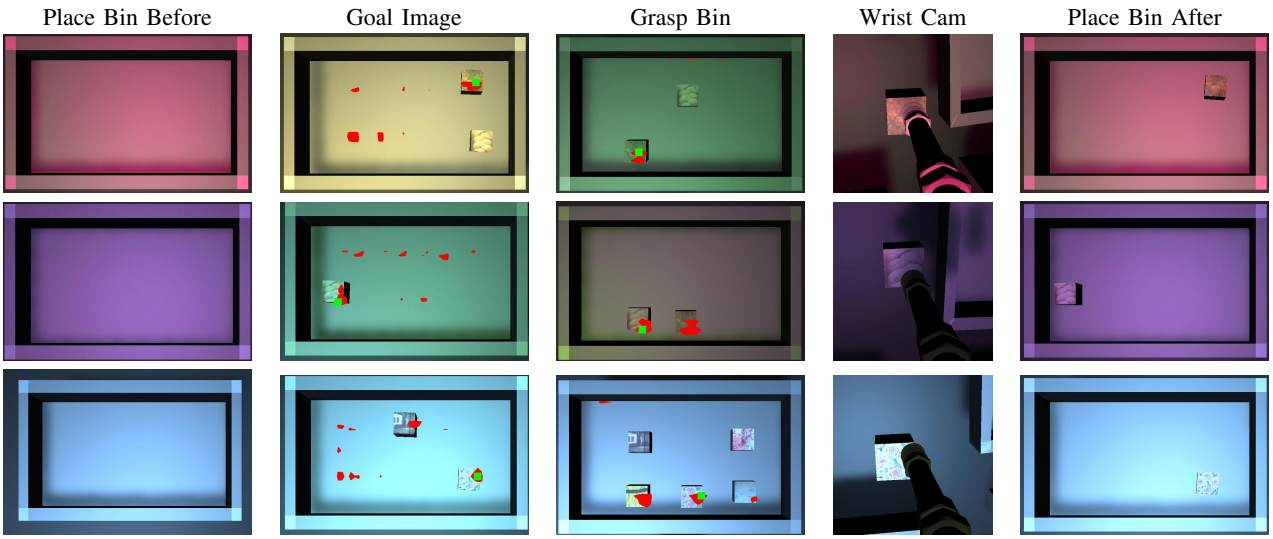


Fig. 2: Examples of goal-conditioned pick and place. The red areas mark the top 10% of pixels that maximize Equation 1 (the green squares show the argmax) with our learned model. We grasp at the green square in the grasp bin, and then use Equation 2 to choose where to place. The place bin after placing is shown on the right. These examples show how our embeddings learned from random grasps can be used. The color of the lights is randomized for each trial.

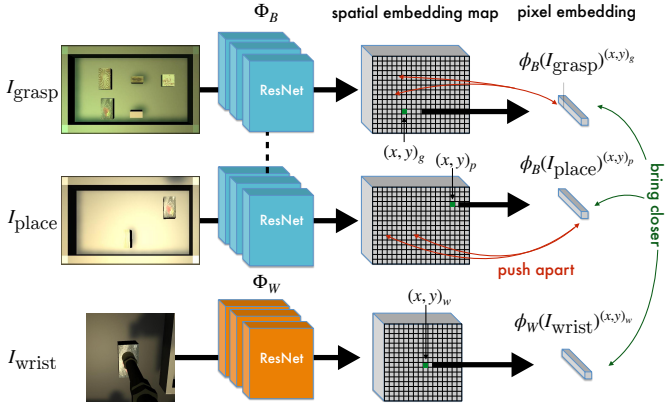


Fig. 3: Our model encodes images into pixelwise embeddings. The grasp and place positions provide supervision about which pixels should have similar embeddings across images.

place I_{place}, and a wrist-cam image of the grasped object in the gripper I_{wrist}. We also record the grasp $(x, y)_g$ and the place $(x, y)_p$ pixel positions, as well as the pixel position of the end-effector $(x, y)_w$ in the wrist image. For performing goal-conditioned actions, we will also have a goal image I_{goal} which show a goal configuration of objects in the place bin.

We aim to learn an image encoder ϕ which embeds images into a spatial feature map that is useful for goal conditioned grasping and placing. Given a bin of objects and novel goal object configuration, this embedding should indicate where to grasp from the bin and where to place in the target area. In order to learn this from random pick and place data, we use a contrastive representation learning approach, as illustrated in Figure 3. We initialize 2 encoders, using the ResNet35 architecture, Φ_B and Φ_W which are applied to bin images (I_{grasp} and I_{place}) and the wrist image I_{wrist}, respectively. These encoders are fully convolutional. They share the same architecture, but have separate weights. The encoders reduce the size of the images by a factor of 8 (512 to 64 on the long edge). Operations over pixel embeddings are done at this reduced dimension.

A. *Contrastive Metric Learning*

Our learning objective for the encoders is that the grasp pixel, place pixel, and end-effector pixel should have similar embeddings while being different from the embeddings at other pixels in the images. Similarity is measured as the dot product of the two embedding vectors, which allows the model to output vectors of small magnitude for regions which are never grasped, such as the background. To optimize for this we use a contrastive approach with a cross entropy loss. We simplify notation and use the superscript (x, y) to index into the spatial embedding map output by Φ.

$$\begin{aligned}\phi^{x,y}_{\text{grasp}} &= \Phi_B(I_{\text{grasp}})^{(x,y)} \\ \phi^{x,y}_{\text{place}} &= \Phi_B(I_{\text{place}})^{(x,y)} \\ \phi_{\text{w}} &= \Phi_W(I_{\text{wrist}})^{(x,y)_w}\end{aligned}$$

Then, $\phi^{(x,y)_g}_{\text{grasp}}$ is the embedding at the grasp pixel in the pick bin image, $\phi^{(x,y)_p}_{\text{place}}$ is the embedding at the place pixel in the place bin image, and ϕ_{w} is the embedding at the end-effector in the wrist image. We want to maximize ${\phi^{(x,y)_g}_{\text{grasp}}}^{\top} \phi_{\text{w}}$, ${\phi^{(x,y)_p}_{\text{place}}}^{\top} \phi_{\text{w}}$, and ${\phi^{(x,y)_g}_{\text{grasp}}}^{\top} \phi^{(x,y)_p}_{\text{place}}$ while minimizing the dot product of these vectors with embeddings at other pixels. To do this, we sample pixels from the bin images as "negatives" for the contrastive loss. Let $v_{n_{g,k}}$ for $k \in [1, K]$ be the embeddings of K "negative" pixels in the pick bin image.

The contrastive loss is implemented as a classification using negative log likelihood. For an anchor embedding a, a positive p and negatives $n_{1:K}$

$$\mathcal{L}_{\text{con}}(a, p, n) = -\log\left(\frac{e^{a^\top p}}{e^{a^\top p} + \sum_{k=1}^{K} e^{a^\top n_k}}\right).$$

We also apply a regularization loss to the magnitudes of the embeddings. The regularization is only applied to embeddings with magnitude greater than 1, and is

$$\mathcal{L}_{\text{reg}}(v) = \begin{cases} ||v||_2 & \text{if } ||v||_2 > 1 \\ 0 & \text{otherwise} \end{cases}$$

The total loss over all the images is then,

$$\begin{aligned}\mathcal{L} =& \mathcal{L}_{\text{con}}(\phi_{\text{grasp}}^{(x,y)_g}, \phi_{\text{w}}, v_{n_{g,1:K}}) + \mathcal{L}_{\text{con}}(\phi_{\text{grasp}}^{(x,y)_g}, \phi_{\text{place}}^{(x,y)_p}, v_{n_{g,1:K}}) \\ &+ \mathcal{L}_{\text{con}}(\phi_{\text{place}}^{(x,y)_p}, \phi_{\text{w}}, v_{n_{p,1:K}}) + \mathcal{L}_{\text{con}}(\phi_{\text{place}}^{(x,y)_p}, \phi_{\text{grasp}}^{(x,y)_g}, v_{n_{p,1:K}}) \\ &+ \sum \mathcal{L}_{\text{reg}}(v) \text{ for } v \text{ in } [\phi_{\text{grasp}}^{(x,y)_g}, \phi_{\text{place}}^{(x,y)_p}, \phi_{\text{w}}, v_{n_{g,1:K}}, v_{n_{p,1:K}}]\end{aligned}$$

We train Φ_B and Φ_W to minimize this loss function over the dataset of grasps using the Adam optimizer [7].

B. Choosing negative samples

We propose two strategies for choosing the negative embeddings in the contrastive loss. The first is to use all other pixels in the spatial embedding as negatives, effectively performing a classification over the image. However, this pushes the embeddings at the selected pixels and its adjacent pixels away from each other even if they are on the same object. As an alternative, we sample distances from the selected pixel according to a Gamma distribution $\gamma(x; \alpha, \beta) = \beta^\alpha x^{\alpha-1} e^{-\beta x}/\Gamma(\alpha)$ and use pixels at those distances as negatives. $\Gamma(\alpha)$ is the gamma function equal to $(\alpha - 1)!$ for integer α. Parameters $\alpha = 4$ and $\beta = w/8$, where w is the image width, were set such that the number of negatives on the selected object are reduced. In our ablations in Section IV-A we find that using both of these strategies together performs best.

III. Using pixel-wise embeddings for goal-conditioned pick and place

We now present an algorithm for using the trained encoders Φ_B and Φ_W to choose where to grasp and place without additional training. We address a goal conditioned pick and place task where the robot is given a goal image I_{goal} and a grasp bin image I_{parts} from which to pick objects and an empty place bin to place into I_{kit}. It must decide where to grasp and where to place to match the goal image. An example of this task is shown in Figure 2. We use the learned embedding space as a similarity metric over bin images

$$\text{sim}(I_{\text{kit}}, I_{\text{goal}}) = \sum_x \sum_y \Phi_B(I_{\text{kit}})^{(x,y)^\top} \Phi_B(I_{\text{goal}})^{(x,y)}$$

for all pixels (x, y). A kit that has been successfully assembled to match a goal should have a very similar embedding map as the goal.

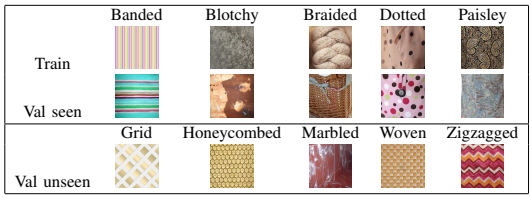


Fig. 4: Example textures from the describable textures dataset that we use to texture objects in the grasping environment. The training set and validation seen set use textures from the same description classes, while validation unseen uses held out description classes.

A. Grasping

In order to assemble a kit that matches the goal, the robot should grasp an object out of the grasp bin that, when added to the place bin, improves $\text{sim}(I_{\text{kit}}, I_{\text{goal}})$. To find the best grasp position in the part tray $(x, y)_g$, we iterate over the pixels in each image to find the pixel leading to most similarity increase:

$$(x, y)_g = \arg\max_{(x,y)} [\max_{(i,j)} [{\phi_{\text{grasp}}^{x,y}}^\top \phi_{\text{goal}}^{i,j} - {\phi_{\text{place}}^{i,j}}^\top \phi_{\text{goal}}^{i,j}]].$$

which can be simplified as

$$(x, y)_g = \arg\max_{(x,y)} [\max_{(i,j)} [(\phi_{\text{grasp}}^{x,y} - \phi_{\text{place}}^{i,j})^\top \phi_{\text{goal}}^{i,j}]] \quad (1)$$

The intuition behind this objective is the idea that the pick and place process will replace the kit's embedding at position $(i, j)^*$ with the grasp bin's embedding at $(x, y)_g$.

B. Placing

After performing the grasp at $(x, y)_g$, we must decide what position to place at in the kit. One option is to place in the kit at pixel position $(i, j)^*$, which should be the optimal place to put the object that was picked. However, a more robust approach is to use the image from the wrist camera after the grasp I_{wrist}. As this image encodes the object as it was actually grasped, we can compare $\Phi_W(I_{\text{wrist}})$ to $\Phi_B(I_{\text{goal}})$ in order to decide where to place the grasped object. Using the wrist image allows the algorithm to correct for errors in the picking process. To choose the place position $(x, y)_p$, we search over the pixels of the kit to see where placing the object would most improve similarity to the goal:

$$(x, y)_p = \arg\max_{(x,y)} \phi_{\text{w}}{}^\top (\phi_{\text{goal}}^{x,y} - \phi_{\text{place}}^{x,y}) \quad (2)$$

The whole pick and place algorithm can be iterated to move multiple objects into the kit.

IV. Experiments

For all datasets, we collect grasps in the simulated environment shown in Figure 1 which is implemented in Unity. The environment does not simulate grasping dynamics: all surfaces of the objects are considered graspable[1]. Data is collected by

[1]Note that the main focus of this work has been on learning visual embeddings and not on learning grasp dynamics, which we leave to future work.

Num Train Textures	**Gamma**	**Full Image**	**Grasp:Place**	Train	Val Train	Val Seen	Val Unseen
10	✓			23	22	20	21
10	✓		✓	20	19	19	18
10		✓		51	29	31	31
10	✓	✓		82	31	32	34
10		✓	✓	23	26	30	29
10	✓	✓	✓	**88**	**54**	**59**	**60**
50	✓			23	24	25	24
50	✓		✓	25	27	27	27
50		✓		70	60	59	59
50		✓	✓	45	29	29	28
50	✓	✓		26	34	31	32
50	✓	✓	✓	**83**	**69**	**70**	**71**

TABLE I: Ablations, all numbers are % accuracy. We report the average object-level accuracy of our model in an offline evaluation. We find that best performance is obtained when **Gamma** and **Full Image** negative samplings are used together in addition to the **Grasp:Place** contrastive loss. We also find that generalization to new textures is no more difficult than generalization to held out grasp data using the same textures as in training set. Training on more textures leads to increased performance on all validation sets.

dropping 6 randomly chosen objects into the grasp bin and I_{grasp} is saved from the camera above the grasp bin. A point on an object is randomly sampled as the grasp point $(x, y)_g$. The object is moved to the gripper by that point and I_{wrist} is saved from the wrist-camera. Finally, a random point $(x, y)_p$ in the place bin is selected at which to place the object, the object is placed, and I_{place} is saved from the camera above the place bin. The light color and brightness are randomized for each step in the data collection. The objects are sampled from a set of rectangular prisms textured with images from the describable textures dataset [2], with examples shown in Figure 4. All training datasets contain 40000 grasps and places. The validation sets contain 500 grasps and places.

A. Offline Evaluation

We first evaluate our model in several offline settings. As we use textures from the describable textures dataset [2], we measure how our model generalizes to new textures both within and outside of the texture classes seen during training. As illustrated in Figure 4, the training data and "val seen" uses different textures form the same class, while the "val unseen" uses textures from held out texture classes. The "val train" setting evaluates the models on held out grasp data that uses the same textures as "train".

For these offline evaluations, we measure grasp and place accuracy by finding the pixel in ϕ_{grasp} and ϕ_{place}, respectively, which has the largest dot product with the embedding of ϕ_{w} at the end-effector. If this pixel is on the correct object, the grasp or place is considered successful. We report the average of both grasp and place offline accuracy.

As discussed in Section II-B, we proposed two methods for sampling negatives: **Full Image**, where all other pixels are used as negatives, and **Gamma** where negatives are sampled to a gamma distribution to avoid pushing apart the embeddings of nearby pixels. We ablate the necessity of contrasting the place embedding against the grasp embedding and vice versa (which we denote **Grasp:Place**), compared to only contrasting each against the wrist-cam image. The results in Table I show that using both negative sampling methods together along with the **Grasp:Place** contrastive loss leads to the best performance at both training and evaluation.

Table I also compares training on just 10 different textures vs 50. We find that training on a greater variety of textures leads to better performance on all validation sets, even the set with the same textures as seen during training. Interestingly, while there is a drop in performance from 83% accuracy on the training set of grasps to 69% on the train-textures validation set, there is no drop from the train-textures validation to the unseen textures: all obtain about 70% accuracy with a model trained on 50 different object textures.

B. Online evaluation

We present preliminary results of using our self-supervised object embeddings in a goal-conditioned pick and place task. As shown in Figure 2, we apply our model to the goal-conditioned pick and place task described in Section III. We freeze Φ_B and Φ_W and use Equations 1 to determine where to grasp and place in order to make the kit match the goal. We find that the similarity metric learned by the embedding functions is highly sensitive to position within an object and can successfully select a correct object and place it into the right location. These preliminary results only show a single step of grasping/placing. Further work is needed to evaluate the multi-step case.

V. Conclusion

We presented a self-supervised approach to learning pixel-wise object-embeddings from random grasping data, and we developed a similarity metric based method for goal-conditioned kitting. We found that our embeddings generalize to unseen textures and unseen textures classes. Our work suggests that these embeddings can be learned from real robots as no part of training requires access to privileged information. We expect that this visual representation learning can be combined with self-supervised approaches for learning *how* to grasp in order to extend this work to objects with more realistic grasp dynamics.

REFERENCES

[1] A. Kuefler B. Goodrich and W. Richards. Depth by poking:learning to estimate depth from self-supervised grasping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.

[2] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[3] Peter R Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv:1806.08756*, 2018.

[4] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[5] Eric Jang, Coline Devin, Vincent Vanhoucke, and Sergey Levine. Grasp2vec: Learning object representations from self-supervised grasping. *arXiv preprint arXiv:1811.06964*, 2018.

[6] Rae Jeong, Yusuf Aytar, David Khosid, Yuxiang Zhou, Jackie Kay, Thomas Lampe, Konstantinos Bousmalis, and Francesco Nori. Self-supervised sim-to-real adaptation for visual robotic manipulation. *arXiv preprint arXiv:1910.09470*, 2019.

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] Guilherme Maeda, Joni Vaatinen, and Hironori Yoshida. Visual task progress estimation with appearance invariant embeddings for robot control and planning. *arXiv preprint arXiv:2003.06977*, 2020.

[9] Deepak Pathak, Yide Shentu, Dian Chen, Pulkit Agrawal, Trevor Darrell, Sergey Levine, and Jitendra Malik. Learning instance segmentation by interaction. In *CVPR Workshop on Benchmarks for Deep Learning in Robotic Vision*, 2018.

[10] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[11] Priya Sundaresan, Jennifer Grannen, Brijen Thananjeyan, Ashwin Balakrishna, Michael Laskey, Kevin Stone, Joseph E Gonzalez, and Ken Goldberg. Learning rope manipulation policies using dense object descriptors trained on synthetic depth data. *arXiv preprint arXiv:2003.01835*, 2020.

[12] Kevin Zakka, Andy Zeng, Johnny Lee, and Shuran Song. Form2fit: Learning shape priors for generalizable assembly from disassembly. *arXiv preprint arXiv:1910.13675*, 2019.