Supervised Training of Dense Object Nets using Optimal Descriptors for Industrial Robotic Applications

Andras Gabor Kupcsik[1], Markus Spies[1], Alexander Klein[2], Marco Todescato[1], Nicolai Waniek[1], Philipp Schillinger[1], Mathias Bürger[1]

[1]Bosch Center For Artificial Intelligence
Renningen, Germany
firstname(s).lastname@de.bosch.com
[2]Technische Universität Darmstadt
Darmstadt, Germany
alex.klein@stud.tu-darmstadt.de

Abstract

Dense Object Nets (DONs) by Florence, Manuelli, and Tedrake (2018) introduced dense object descriptors as a novel visual object representation for the robotics community. It is suitable for many applications including object grasping, policy learning, etc. DONs map an RGB image depicting an object into a descriptor space image, which implicitly encodes key features of an object invariant to the relative camera pose. Impressively, the self-supervised training of DONs can be applied to arbitrary objects and can be evaluated and deployed within hours. However, the training approach relies on accurate depth images and faces challenges with small, reflective objects, typical for industrial settings, when using consumer grade depth cameras. In this paper we show that given a 3D model of an object, we can generate its descriptor space image, which allows for supervised training of DONs. We rely on Laplacian Eigenmaps (LE) to embed the 3D model of an object into an optimally generated space. While our approach uses more domain knowledge, it can be efficiently applied even for smaller and reflective objects, as it does not rely on depth information. We compare the training methods on generating 6D grasps for industrial objects and show that our novel supervised training approach improves the pick-and-place performance in industry-relevant tasks.

1 Introduction

Dense object descriptors for perceptual object representation received considerable attention in the robot learning community (Florence, Manuelli, and Tedrake 2018, 2020; Sundaresan et al. 2020). To learn and generate dense visual representation of objects, Dense Object Nets (DONs) were proposed by Florence, Manuelli, and Tedrake (2018). DONs map an $h \times w \times 3$ RGB image to its descriptor space map of size $h \times w \times D$, where $D \in \mathbb{N}^+$ is an arbitrarily chosen dimensionality. DONs can be trained in a self-supervised manner using a robot and a wrist-mounted consumer grade RGBD camera, and can be deployed within hours. Recently, several impactful applications and extensions of the original approach have been shown, including rope manipulation (Sundaresan et al. 2020), behavior cloning (Florence, Manuelli, and Tedrake 2020) and controller learning (Manuelli et al. 2020).

DONs can be readily applied to learn arbitrary objects with relative ease, including non-rigid objects. The self-supervised training objective of DONs uses contrastive loss (Hadsell, Chopra, and LeCun 2006) between pixels of image pairs depicting the object and its environment. The pixel-wise contrastive loss minimizes descriptor space distance between corresponding pixels (pixels depicting the same point on the object surface in an image pair) and pushes away non-correspondences. In essence, minimizing the contrastive loss defined on descriptors of pixels leads to a view invariant map of the object surface in descriptor space.

The contrastive loss formulation belongs to the broader class of projective nonlinear dimensionality reduction techniques (Van Der Maaten, Postma, and Van Den Herik 2009). The motivation behind most of these methods is to perform a mapping from an input manifold to a typically lower dimensional output space while ensuring that similar inputs map to similar outputs. While the contrastive loss formulation relies on a similarity indicator (e.g., matching vs. non-matching pixels), other techniques also exploit the *magnitude* of similarity implying local information about the data (e.g., ISOMAP (Tenenbaum, De Silva, and Langford 2000) and LE (Belkin and Niyogi 2003)).

While generating similarity indicators, or pixel correspondences, is suitable for self-supervised training, its accuracy inherently depends on the quality of the recorded data. Based on our observations, noisy depth data can deteriorate the quality of correspondence matching especially in the case of smaller objects. As an alternative, in this paper we generate an optimal descriptor space embedding given a 3D mesh model, leading to a supervised training approach. Optimality here refers to embedding the model vertices into a descriptor space with minimal distortion of their local connectivity information. We rely on discrete exterior calculus to compute the Laplacian of the object model (Crane et al. 2013), which generates the geometrically accurate local connectivity information. We exploit this information in combination with Laplacian Eigenmaps to create the corresponding optimal descriptor space embedding. Finally, we render target descriptor space images to input RGB images

Figure 1: Illustration of the proposed approach (best viewed in color). Given the 3D model of an object **(left top)** we generate its optimal descriptor space embedding using Laplacian Eigenmaps **(left bottom)**. Then, for a given input image **(middle)** with known object pose we render its descriptor space target **(right)**. For illustration purposes we use 3 dimensional descriptor space.

depicting the object (see Fig. 1). Ultimately, our approach generates dense object descriptors akin the original, self-supervised method without having to rely on depth information.

While our approach uses more domain knowledge (3D model of the object and its known pose), it has several benefits over the original self-supervised method. Primarily, we do not rely on pixel-wise correspondence matching based on noisy or lower quality consumer grade depth cameras. Thus, our approach can be applied straightforwardly to small, reflective, or symmetric objects, which are often found in industrial processes. Furthermore, we explicitly separate object and background descriptors, which avoids the problem of amodal correspondence prediction when using self-supervised training (Florence 2020). We also provide a mathematical meaning for the descriptor space, which we generate optimally in a geometrical sense irrespective of the descriptor dimension. We believe that, overall, this improves explainability and reliability for practitioners.

2 Background

This chapter briefly reviews self-supervised training of Dense Object Nets (Florence, Manuelli, and Tedrake 2018) and nonlinear dimensionality reduction by Laplacian Eigenmaps.

2.1 Self-supervised Training of DONs

To collect data for the self-supervised training approach, we use a robot and a wrist-mounted RGBD camera.[4] We place the target object to an arbitrary but fixed location in the workspace of the robot. Then, using a quasi-random motion with the robot and the camera pointing towards the workspace,[2] we record a scene with registered RGBD images depicting the object and its environment. To overcome noisy and often missing depth data of consumer grade depth sensors,[2] all registered RGBD images are then fused into a single 3D model and depth is recomputed for each frame. While this will improve the overall depth image quality, we noticed that in practice this also over-smooths it, which results in a loss of geometrical details particularly for small objects. With knowledge of the object location in the fused model we can also compute the object mask in each frame.

After recording a handful of scenes with different object poses, for training we repeatedly and randomly choose two different RGB images within a scene, I_a and I_b, to evaluate the contrastive loss. We use the object masks to sample N_m corresponding and N_{nm} non-corresponding pixels. Then, the contrastive loss consists of match loss L_m of corresponding pixels and non-match loss L_{nm} of non-corresponding pixels:

$$L_m = \frac{1}{N_m} \sum_{N_m} D(I_a, u_a, I_b, u_b)^2, \tag{1}$$

$$L_{nm} = \frac{1}{N_{nm}} \sum_{N_{nm}} \max\left(0, M - D(I_a, u_a, I_b, u_b)\right)^2, \tag{2}$$

$$L_c(I_a, I_b) = L_m(I_a, I_b) + L_{nm}(I_a, I_b), \tag{3}$$

where $D(I_a, u_a, I_b, u_b) = \|f(I_a; \boldsymbol{\theta})(u_a) - f(I_b; \boldsymbol{\theta})(u_b)\|_2$ is the descriptor space distance between pixels u_a and u_b of images I_a and I_b. $M \in \mathbb{R}^+$ is an arbitrarily chosen margin and $f(\cdot; \boldsymbol{\theta}) : \mathbb{R}^{h \times w \times 3} \mapsto \mathbb{R}^{h \times w \times D}$ represents a fully convolutional network (Shelhamer, Long, and Darrell 2017) with parameters $\boldsymbol{\theta}$.

The efficiency of the self-supervised training approach lies in the automatic generation of pixel correspondences from registered RGBD images. Using the object mask in image I_a we can sample a pixel u_a and identify corresponding pixel u_b in image I_b by reprojecting the depth information. In a similar way we can sample non-correspondences on the object and on the background. While this approach automatically labels tens of thousands of pixels in a single image pair, its accuracy inherently relies on the quality of the depth image. In practice we noticed that this considerably limits the accuracy of the correspondence matching for smaller objects with consumer grade depth sensors. For further details of the training and evaluation of DONs we refer the reader to (Florence, Manuelli, and Tedrake 2018).

2.2 Laplacian Eigenmaps

Assume a dataset of N data points is given as $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$, where $\boldsymbol{x} \in \mathcal{M}$ lies on a manifold. Furthermore, we are given the connectivity information between these points as $w_{ij} \in \mathbb{R}^{\geq 0}$. For example, if $\boldsymbol{x}_i$ is a node in a graph, then we can define w_{ij} to be either 1 or 0 given it is connected

to $\boldsymbol{x}_j$ or not. The Laplacian Eigenmap method considers the problem of finding an embedding of the data points in $\boldsymbol{X}$ to $\{\boldsymbol{y}_i\}_{i=1}^N$, with $\boldsymbol{y} \in \mathbb{R}^D$, such that the local connectivity w_{ij} measured in a Euclidean sense is preserved. We can express this objective as a constrained optimization problem

$$\boldsymbol{Y}^* = \arg\min_{\boldsymbol{Y}} \frac{1}{2} \sum_{j=1}^{N} \sum_{i=1}^{N} w_{ij} \left\| \boldsymbol{y}_i - \boldsymbol{y}_j \right\|_2^2 \quad (4)$$

$$\text{s.t. } \boldsymbol{Y}\boldsymbol{C}\boldsymbol{Y}^T = \boldsymbol{I}, \quad (5)$$

where $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N] \in \mathbb{R}^{D \times N}$, $\boldsymbol{C}_{ii} = \sum_j \boldsymbol{A}_{ij}$ is a diagonal matrix with $\boldsymbol{A}_{ji} = \boldsymbol{A}_{ij} = w_{ij}$ as the connectivity matrix. $\boldsymbol{C} \in \mathbb{R}^{N \times N}$ measures how strongly the data points are connected. The constraint removes bias and enforces unit variance in each dimension of the embedding space. As it was shown in the seminal paper by Belkin and Niyogi (2003), this optimization problem can be expressed using the Laplacian $\boldsymbol{L}_{N \times N}$ as

$$\boldsymbol{Y}^* = \arg\min_{\boldsymbol{Y}} \text{Tr}\left(\boldsymbol{Y}\boldsymbol{L}\boldsymbol{Y}^T\right) \quad (6)$$

$$s.t.\ \boldsymbol{Y}\boldsymbol{C}\boldsymbol{Y}^T = \boldsymbol{I},$$

where $\boldsymbol{L} = \boldsymbol{C} - \boldsymbol{A}$ is a positive semi-definite, symmetric matrix. The solution to this optimization problem can be found by solving a generalized eigenvalue problem

$$\boldsymbol{L}\boldsymbol{Y}^T = \text{diag}(\boldsymbol{\lambda})\boldsymbol{C}\boldsymbol{Y}^T, \quad (7)$$

with eigenvectors corresponding to the dimensions of $\boldsymbol{y}$ (or rows of $\boldsymbol{Y}$) and non-decreasing real eigenvalues $\boldsymbol{\lambda} \in \mathbb{R}^{\geq 0}$. Interestingly, the first dimension is constant for each data point, i.e., $\boldsymbol{Y}_{1,:} = \text{const}$ and $\lambda_1 = 0$. This corresponds to embedding every data point in $\boldsymbol{X}$ to a single point. For practical reasons the very first dimension is ignored ($D = 0$). Therefore, choosing $D = 1$ corresponds to embedding $\boldsymbol{X}$ on a line, $D = 2$ to a plane, etc. Given that the Laplacian is symmetric positive semi-definite (when we use the definition $\boldsymbol{L} = \boldsymbol{C} - \boldsymbol{A}$), the eigenvectors are orthogonal to each other. Additionally, with the optimal solution we have $\text{diag}(\boldsymbol{\lambda}) = \boldsymbol{Y}^* \boldsymbol{L} \boldsymbol{Y}^{*T}$, that is, the eigenvalues represent the embedding error in each dimension. Consequently, as these eigenvalues are non-decreasing, we always arrive at an optimal embedding irrespective of the dimensionality D.

Note that both contrastive learning and Laplacian Eigenmaps achieve a highly related objective, that is, minimizing distances in embedding space between related, or similar data. However, they differ in how they avoid the trivial solution (mapping to a single point), either by pushing dissimilar data by a margin away, or by normalizing the solution. Finally, the contrastive formulation gives rise to learning parametric models (e.g., neural networks), while LE directly generates the embedding from the Laplacian, which is specific to a manifold (e.g., 3D mesh).

3 Method

In this Section we describe our proposed solution to generate optimal training targets for training input RGB images and describe the supervised training setup. We first describe how we generate an optimal descriptor space map for a 3D object that is represented by a triangle mesh and how we can handle symmetries. Then we explain the full target image generation and supervised training pipeline.

Note that the contrastive loss of the self-supervised training approach is defined over pixel descriptors of RGB images depicting the object. In our case, we exploit the LE embedding directly on the mesh of the model. This allows us to exploit the geometry of the object and to address optimality of the embedding. As a second step we render the descriptor space representation of the object to generate descriptor space images for the supervised training objective.

3.1 Embedding Object Models using Laplacian Eigenmaps

We assume that a 3D model of an object is a triangle mesh consisting of $N = |V|$ vertices V, edges E and faces F. We consider the mesh as a Riemannian manifold $\mathcal{M}$ embedded in $\mathbb{R}^3$. The vertices $\boldsymbol{x}_i \in \mathcal{M}$ are points on this manifold. We are looking for the descriptor space maps of the vertices $\boldsymbol{y}_i \in \mathbb{R}^D$. To apply the LE solution we need to compute the Laplacian of the object model. For triangle meshes this can be computed based on discrete exterior calculus (Crane et al. 2013). For an efficient and easy to implement solution we refer to (Sharp, Soliman, and Crane 2019). After computing the Laplacian, we can solve the same generalized eigenvalue problem as in Eq. (7) to compute the D-dimensional descriptor space embedding of the vertices $\boldsymbol{Y} \in \mathbb{R}^{D \times N}$.

Fig. 2 illustrates the solution of the eigenvalue problem for the Stanford bunny and the resulting descriptor space embedding with $D = 3$. We project the solution on the vertices of the object model and use a renderer to color the faces.

3.2 Handling Symmetry

So far we map every vertex of the mesh to a unique point in descriptor space. However, for objects with symmetric geometric features, typical in industrial settings, this approach will assign unique descriptors to indistinguishable vertices. Consider the case of the torus in Fig. 3. The mesh is invariant to rotations around the z-axis. If we apply the Laplacian embedding approach, we end up with descriptors that do not preserve this symmetry (top right in Fig. 3). The descriptor values will be assigned purely based on the ordering of the vertices in the mesh. Instead, we are looking for an embedding as in the bottom of Fig. 3, which only differentiates between "inside-outside" vertices and which appears invariant to rotation around z-axis.

To overcome this problem, we have to (i) detect intrinsic symmetries of shapes and (ii) compress symmetric embeddings, such that symmetric vertices map to the same descriptor. Ovsjanikov, Sun, and Guibas (2008) discuss an approach for detecting intrinsic symmetries for shapes represented as compact (Riemannian) manifolds. In particular, they showed that a shape has intrinsic symmetry if the eigenvectors of the Laplacian, that is, its Euclidean embedding appear symmetric. Following this result, Wang et al. (2014) defined Global Intrinsic Symmetry Invariant Functions (GISIFs) on vertices

Figure 2: Illustration of the Laplacian Eigenmap embedding of the Stanford bunny (best viewed in color). **(left):** the triangle mesh representation, **(middle three):** the first three eigenvector projected on the mesh, scaled between $[0, 1]$ and visualized with hot color map, **(right):** the descriptor space representation of the mesh visualized with red corresponding to the first eigenvector, or descriptor dimension, green for the second and blue for the third.

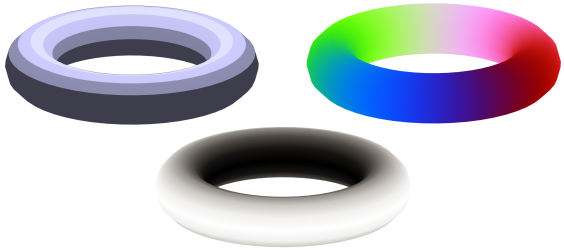

Figure 3: **(top left)** the mesh of the torus, **(top right)** the asymmetric 3-dimensional descriptor projected on the mesh, **(bottom)** the symmetric 3-dimensional descriptor projected on the mesh.

that preserve their value in the face of any homeomorphism (such as rotation around z-axis in the case of the torus). They also showed that such a GISIF can be composed of eigenvectors of the Laplacian.

In particular, they propose that in case of identical consecutive eigenvalues $\lambda_i = \lambda_{i+1} = \cdots = \lambda_{i+L}$, such a GISIF is the squared sum of the corresponding eigenvectors $\boldsymbol{y}_{\text{sym}} = \sum_i^{i+L} \boldsymbol{y}_i^2$. In practice eigenvalues are rarely identical due to numerical limitations. To resolve this problem we heuristically consider eigenvalues in an ϵ-ball as identical.

3.3 Generating Target Images and Network Training

Given the optimal embedding of an object model in descriptor space and the registered RGB images of a static scene, we can now generate the target descriptor space images for training. In the collected dataset we have K registered RGB images I_i with camera extrinsic $\boldsymbol{T}_i^c \in SE(3)$, $\{I_i, \boldsymbol{T}_i^c\}_{i=1}^K$ depicting the object and its environment. We compute the descriptor space embedding of the model (see Sec. 3.1 and 3.2) and we assume the object pose in world coordinates $\boldsymbol{T}^o \in SE(3)$ is known. To generate target descriptor space images I_i^d we can use a rendering engine to project the descriptor space model to the image plane of the camera, see Fig. 1 for an illustration. The pose of the object in the camera coordinate system can be computed from the camera poses and the object pose with $\boldsymbol{T}_i^{o,c} = \boldsymbol{T}_i^{c-1}\boldsymbol{T}^o$. To generate the background image we first normalize the descriptor dimensions between $[0, 1]$. Then, as background we choose the descriptor which is the furthest away from object descriptors within a unit cube. By explicitly separating object and background it becomes more unlikely to predict object descriptors in the background, which may occur when using the self-supervised training approach, also reported in (Florence 2020).

To train the network we rely on ℓ_2 loss between DON output and generated descriptor space images. However, in a given image typically the amount of pixels representing the object is significantly lower than that of the background. Therefore, we separate the total loss into object and background loss and normalize them with the amount of pixels they occupy

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{K} \Big(\frac{1}{P_{i,obj}} \left\| M_{i,obj} \circ \left(f(I_i; \boldsymbol{\theta}) - I_i^d \right) \right\|_2^2 + \frac{1}{P_{i,back}} \left\| M_{i,back} \circ \left(f(I_i; \boldsymbol{\theta}) - I_i^d \right) \right\|_2^2 \Big), \tag{8}$$

where $\boldsymbol{\theta}$ are the network parameters, $P_{i,obj/back}$ is the amount of pixels the object, or background occupies in the image I_i and $M_{i,obj/back}$ is the object or background mask. For convenience here we overloaded the notation M, which refers to margin in the self-supervised training approach (Sec. 2.1) and mask here. The mask can be generated via rendering the object in the image plane, similar to how we generate target images I_i^d.

We compare the training pipeline of the original self-supervised DON framework (Florence, Manuelli, and Tedrake 2018) and our proposed supervised approach in Fig. 4. We use the same static scene recording with a wrist-mounted camera for both training approaches. While the self-supervised training approach takes the depth, RGB, and camera extrinsic parameters, ours takes the model pose in world coordinates and its mesh instead of depth images. Then, both approaches perform data preprocessing, 3D model fusion, depth and mask rendering for the contrastive loss, object descriptor generation and descriptor image rendering for the ℓ_2 loss. Finally, both training approaches optimize the network parameters.

3.4 View-dependent Descriptors

Note that our framework can be extended with additional descriptor dimensions that not necessarily minimize the embedding objective defined in Eq. (4). In practice we noticed

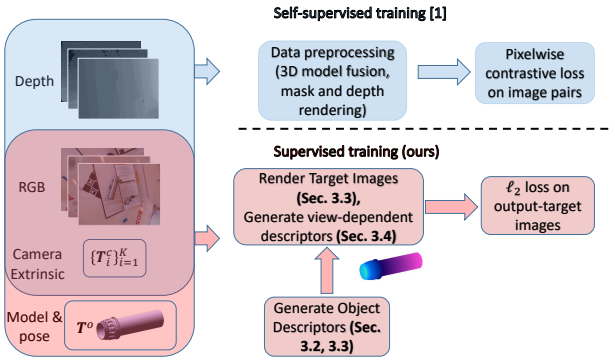

Figure 4: Comparison of the self-supervised and the supervised training (ours) pipelines. The first utilizes depth images to find pixel correspondences and relies on the contrastive pixel-wise loss. Ours exploits the object model and its pose in world coordinates to render training target images, which will be used in the loss function (Eq. (8)).

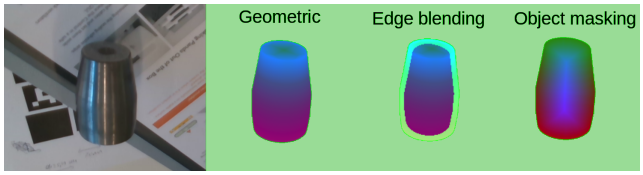

Figure 5: Visualization of view-dependent descriptors. On the left side we can see the original RGB image, depicting the toolcap. Using $D = 3$ we generate the optimal geometric descriptor image. Then, we can either blend the object edge with the background, or generate an object mask.

that the self-supervised training approach learns descriptor dimensions which mask the object and its edge resulting in *view-dependent* descriptors. This is likely a consequence of the self-supervised training adapting to slightly inaccurate pixel correspondence generation in the face of camera extrinsic errors and oversmoothed depth images. As our training method relies on the same registered RGBD scene recording, to improve training robustness we considered two view-dependent descriptors as an extension to our optimally generated geometric descriptors.

First, we considered blending descriptors on the edge of the object with the background descriptor. This mimics a view-dependent edge detection descriptor that separates object and background. Second, we can use a descriptor dimension as object mask that is 0 outside the object and gradually increases to 1 at the center of the object in the image plane. The former option smooths the optimal descriptor images around the edges, while the latter extends or replaces descriptor dimensions with an object mask. These two additional descriptors together with the optimal geometric descriptors are shown in Fig. 5 for a metallic toolcap object. Note that these features can be computed automatically from the object mask of the current frame.

We experimented with these extensions to improve grasp pose prediction tasks in practice. This extension shows on one hand the flexibility of using our approach by manipulating descriptor dimensions. On the other hand, it highlights the benefits of using the self-supervised training approach to automatically address inaccuracies in the data.

4 Evaluation

In this section we provide a comparison of the trained networks. While the DON framework can be applied for different robotic applications, in this paper we consider the oriented grasping of industrial work pieces. To this end we first provide a quantitative evaluation of the prediction accuracy of the networks. We measure accuracy by how consistently the network predicts descriptors for specific points on the object from different points of view. Then, we derive a grasp pose prediction pipeline and use it in a real-world robot grasp and placement scenario.

4.1 Hardware, Network and Scene Setup

We use a Franka Emika Panda 7-DoF robot with a parallel gripper (Franka 2018), equipped with a wrist-mounted Intel RealSense D435 camera. The relative transformation from end-effector to camera coordinate system is computed with ChArUco board calibration (Garrido-Jurado et al. 2014) and forward kinematics of the robot. We record RGB and depth images of size 640×480 and at $100\mu m$ resolution at 15Hz. For a given scene we record ~ 500 images with a precomputed robot end-effector trajectory resulting in highly varying camera poses. Our target objects are a metallic shaft and a toolcap, both of which appear invariant to rotation around one axis (see Fig. 6).

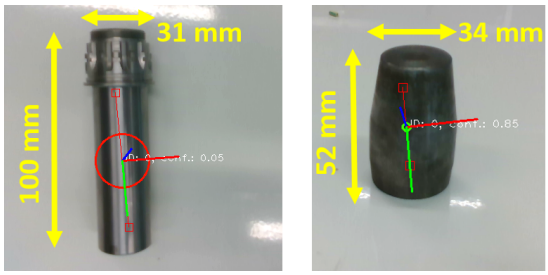

Figure 6: The metallic shaft and toolcap objects with predicted grasp poses by tracking 2 descriptors, or points on the objects.

For both, self-supervised and supervised training, we use descriptor dimension $D = 3$, consider only a single object per scene, and train with background randomization. For the supervised approach we consider two ways of generating data. We first compute the optimal embedding, then we either use edge blending, or object masking. In the latter case we use the first two geometric descriptor dimensions and add the view-dependent object mask as the third (see Fig. 5). For the self-supervised approach we use the hyper-parameters defined in (Florence, Manuelli, and Tedrake 2018). That is, we use $M = 0.5$ as margin for on-object non-match loss, $M = 2.5$ for background non-match loss. We do not use normalization of the descriptors and use scaling by hard-negatives. The network structure for both method is ResNet-34 pretrained on ImageNet.

4.2 Quantitative Evaluation of Prediction Accuracy

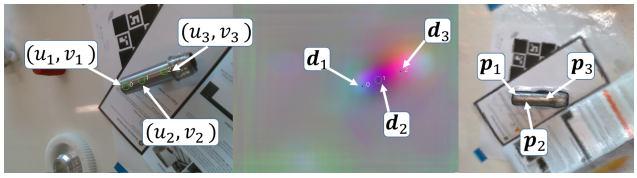

Figure 7: Given a scene during evaluation, we select a few pixels on the RGB image corresponding to points on an object (shown for 3 points). We record their descriptors and world coordinates as a tracking reference.

We now provide an evaluation for how accurately the networks can track points on an object from different views. For a given static scene we choose an RGB image I and select an arbitrary amount of reference pixels *on the object* $(u_i, v_i),\ i = 1, \ldots, N$ (see Fig. 7 left). Then, we record the 3D world coordinates of the pixels $\boldsymbol{p}_i \in \mathbb{R}^3$ using the registered depth images and their corresponding descriptors $\boldsymbol{d}_i = I^d(u_i, v_i) \in \mathbb{R}^D$, where $I^d = f(I; \boldsymbol{\theta})$ is the descriptor image evaluated by the network (see Fig. 7 middle and right). Then, we iterate over every image $I_j,\ j = 1, \ldots, M$ in the scene and evaluate the network $I^d_j = f(I_j; \boldsymbol{\theta})$. In every descriptor space image I^d_j we find the closest descriptors to our reference set $\boldsymbol{d}_i$ and their pixel location (u_i, v_i) by solving $(u_i, v_i)^*_j = \arg\min_{u_i, v_i} \left\| I^d_j(u_i, v_i) - \boldsymbol{d}_i \right\|^2_2,\ \forall j$. We only consider those reference points which are visible and have valid depth data. Using the registered depth images and pixel values $(u_i, v_i)^*_j$ we can compute the predicted world coordinates of the points $\tilde{\boldsymbol{p}}_i$ and the tracking error $\|\tilde{\boldsymbol{p}}_i - \boldsymbol{p}_i\|_2$ for each frame j and for each chosen descriptor i.

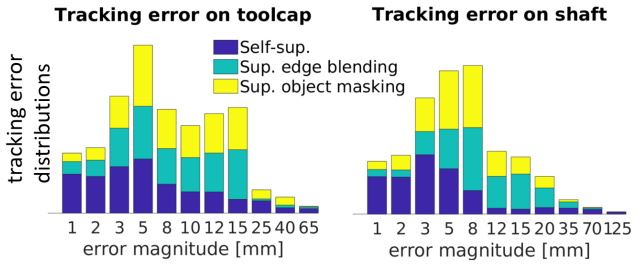

Figure 8: Tracking error distributions evaluated on every scene for both objects. Note the log-scale of the x-axis. While self-supervised training provides higher accuracy, it has a long-tail error distribution, which sometimes leads to significantly higher errors. The supervised approaches have a slightly lower accuracy, but increased robustness.

Fig. 8 shows the tracking error distributions of the supervised and self-supervised approaches. For improved visualization, the normalized distributions are summarized with stacked histograms and with log-scale on the x axis. In general, the self-supervised training method seems more accurate, but we notice a long-tail error distribution that can lead to significant prediction errors. The supervised approaches provide better robustness without the long-tail distribution, albeit being slightly less accurate. We noticed that the self-supervised training leads to a descriptor space that often mistakes parts of the background as the object, which could lead to higher tracking errors. With our supervised training method this occurs less frequently as we explicitly separate object and background in descriptor space, while this separation is less apparent in the self-supervised case.

4.3 Grasp Pose Prediction

The dense descriptor representation gives rise to a variety of grasp pose configurations. In the simplest case, consider a 3D top-down grasp: choose a descriptor $\boldsymbol{d}$ that corresponds to a specific point on the object. Then, in a new scene with a registered RGBD image, evaluate the network and find the closest descriptor $\boldsymbol{d}$ with corresponding pixel values (u, v). Finally, project the depth value at (u, v) in world coordinates to identify the grasp point $\tilde{\boldsymbol{p}} \in \mathbb{R}^3$. This method previously described in Florence, Manuelli, and Tedrake (2018) together with grasp pose optimization.

Following this method, we can also encode orientation by defining an axis grasp. For this purpose we identify 2 descriptors $\boldsymbol{d}_1, \boldsymbol{d}_2$ and during prediction find the corresponding world coordinates $\tilde{\boldsymbol{p}}_1, \tilde{\boldsymbol{p}}_2$. To define the grasp position we use a linear combination of $\tilde{\boldsymbol{p}}_1, \tilde{\boldsymbol{p}}_2$, e.g. taking their mean. To define the grasp orientation we align a given axis with the direction $\tilde{\boldsymbol{p}}_1 - \tilde{\boldsymbol{p}}_2$ and choose an additional axis arbitrarily, e.g. align it with the z-axis of the camera, or world-coordinates. The third axis can then be computed by taking the cross product. See Fig. 6 for predicted axis-grasps on two test objects. Finally, to fully specify a given 6D pose we can extend the above technique to track 3 or more descriptors.

For our test objects we have chosen an axis grasp representation by tracking 2 descriptors. We align the x-axis of the grasp pose with the predicted points and we choose as z-axis the axis orthogonal to the image plane of the camera.

Experiment setup. Consider an industrial setting where a human places an object with an arbitrary position and orientation in the workspace of a robot. The robot's goal is to grasp the object and place it at a handover location, such that another, pre-programmed robot (without visual feedback) can grasp and further manipulate the object in an assembly task. The setup is shown in Fig. 9.

For real robot grasping evaluations we have chosen the toolcap as our test object. We observed that the experiment with shaft grasping was not challenging for any of the networks. However, we noticed that due to the smaller size of the toolcap and its symmetric features the self-supervised training approach was pushed to the limit in terms of accuracy. To reduce variance, we have chosen 8 *fixed* poses for the toolcap and repeated the experiment for each trained DON on each of the 8 poses. We categorize the experiment outcome as follows:

- `fail`: if the grasp was unsuccessful, the toolcap remains untouched
- `grasp only`: if the grasp was successful, but placement failed (e.g., the toolcap was dropped, or was grasped

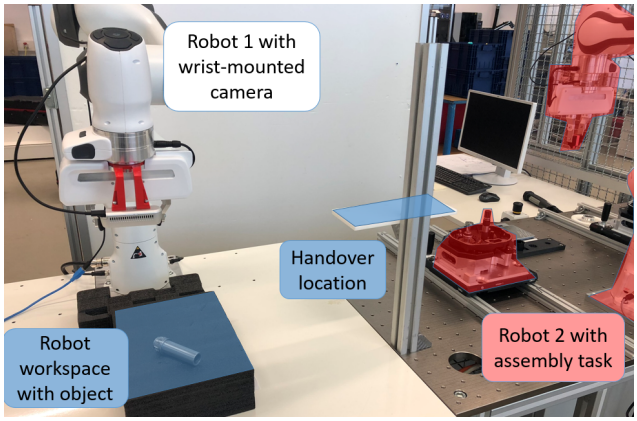

Figure 9: The summary of the experiment setup.

with the wrong orientation)

- `grasp & place`: if grasping and placement with the right orientation were successful

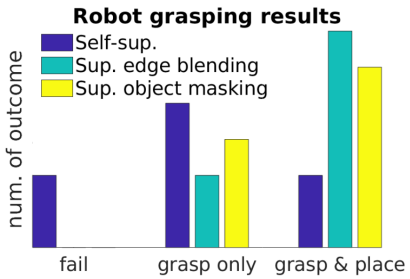

Figure 10: Summary of the real robot experiments.

Fig. 10 illustrates results of the experiment. The self-supervised training resulted in grasps that are often misaligned, which could lead to a failed placement or even a failed grasp. The supervised training approaches show better robustness with at least a successful grasp in all the cases. Most failed placements were due flipped orientations leading to upside-down final poses. Overall, the results are highly consistent with our previous evaluation on reference point tracking accuracy. The supervised trained approaches provided more robust performance.

5 Related Work

Dense descriptor learning via DONs was popularized by Florence, Manuelli, and Tedrake (2018) for the robotics community, offering a flexible perceptual world representation with self-supervised training, which is also applicable to non-rigid objects. Recently, this work was extended to learn descriptors in dynamic scenes for visuo-motor policy learning (Florence, Manuelli, and Tedrake 2020). Manuelli et al. (2020) propose to define model predictive controllers based on tracking key descriptors, similar to our grasp pose prediction pipeline. Sundaresan et al. (2020) showed how ropes can be manipulated based on dense descriptor learning in simulation. Similar to these approaches Vecerik et al. (2020) propose to learn and track human annotated keypoints for robotic cable plugging. Visual representation by dense descriptor is not novel though. Choy et al. (2016) introduced a deep learning framework for dense correspondence learning of image features using fully convolutional neural networks by Shelhamer, Long, and Darrell (2017). Later, Schmidt, Newcombe, and Fox (2017) showed how self-supervised training can be incorporated when learning descriptors from video streams.

Recently, several authors used dense object descriptors for 6D pose estimation. Zakharov, Shugurov, and Ilic (2019) generate 2D or 3D surface features (colors) of objects via simple spherical or cylindrical projections. Then, they train a network in a supervised manner to predict these dense feature descriptors and a mask of the object. For 6D pose estimation, they project the colored mesh into the predicted feature image. Periyasamy, Schwarz, and Behnke (2019) and Li et al. (2018) propose similar training procedures to estimate dense feature descriptors. Periyasamy, Schwarz, and Behnke (2019) do not report details for generating the dense descriptors and use differentiable rendering for object pose estimation. Li et al. (2018) use relative translations of the object as dense feature descriptors and a combination of rendering and an additional refinement network for pose estimation. The supervised training procedure proposed in all of the aforementioned methods is similar to ours. The main difference is our novel method of computing optimal dense descriptors that locally preserves geometric properties.

Our work also borrows ideas from the geometry processing community. Lee and Verleysen (2005) consider embedding data manifolds with non-linear dimensionality reduction that have specific geometric features. Liu, Prabhakaran, and Guo (2012) propose a spectral processing framework that also works with point clouds to compute a discrete Laplacian. The works by Crane et al. (2013); Crane, Weischedel, and Wardetzky (2017) build on discrete exterior calculus to efficiently compute the LB operator for meshes, which we also use in this work. Finally, detecting and processing symmetrical features of 3D objects represented as Riemannian manifolds is considered by the works of Ovsjanikov, Sun, and Guibas (2008); Wang et al. (2014).

6 Summary

In this paper we presented a supervised training approach for Dense Object Nets from registered RGB camera streams without depth information. We showed that by embedding an object model into an optimally generated descriptor space we achieve a similar objective as in the self-supervised case. Our experiments show increased grasp pose prediction robustness for smaller, metallic objects. While self-supervised training has obvious benefits, our supervised method improved results for consumer grade cameras by incorporating domain knowledge suitable for industrial processes.

Future work will investigate how to extend supervised training with multiple object classes efficiently and a wider range of applications. For example, 6D pose estimation from RGB images, region of interest detection and generalizing grasp pose prediction to multiple object instances and classes.

References

Belkin, M.; and Niyogi, P. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation* 15(6): 1373–1396. ISSN 08997667. doi:10.1162/089976603321780317.

Choy, C. B.; Gwak, J.; Savarese, S.; and Chandraker, M. 2016. Universal Correspondence Network. In *Advances in Neural Information Processing Systems 30*.

Crane, K.; de Goes, F.; Desbrun, M.; and Schröder, P. 2013. Digital Geometry Processing with Discrete Exterior Calculus. In *ACM SIGGRAPH 2013 courses*, SIGGRAPH '13. New York, NY, USA: ACM.

Crane, K.; Weischedel, C.; and Wardetzky, M. 2017. The heat method for distance computation. *Communications of the ACM* 60(11): 90–99. ISSN 15577317. doi:10.1145/3131280.

Florence, P.; Manuelli, L.; and Tedrake, R. 2018. Dense Object Nets: Learning Dense Visual Object Descriptors By and For Robotic Manipulation. *Conference on Robot Learning* .

Florence, P.; Manuelli, L.; and Tedrake, R. 2020. Self-Supervised Correspondence in Visuomotor Policy Learning. *IEEE Robotics and Automation Letters* 5(2): 492–499. ISSN 23773766. doi:10.1109/LRA.2019.2956365.

Florence, P. R. 2020. *Dense Visual Learning for Robot Manipulation*. Ph.D. thesis, Massachusetts Institute of Technology.

Franka, E. 2018. Panda Arm. https://www.franka.de/panda/.

Garrido-Jurado, S.; Muñoz Salinas, R.; Madrid-Cuevas, F.; and Marín-Jiménez, M. 2014. Automatic Generation and Detection of Highly Reliable Fiducial Markers under Occlusion. *Pattern Recogn.* 47(6): 2280–2292. ISSN 0031-3203. doi:10.1016/j.patcog.2014.01.005. URL https://doi.org/10.1016/j.patcog.2014.01.005.

Hadsell, R.; Chopra, S.; and LeCun, Y. 2006. Dimensionality reduction by learning an invariant mapping. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2: 1735–1742. ISSN 10636919. doi:10.1109/CVPR.2006.100.

Lee, J. A.; and Verleysen, M. 2005. Nonlinear dimensionality reduction of data manifolds with essential loops. *Neurocomputing* 67(1-4 SUPPL.): 29–53. ISSN 09252312. doi:10.1016/j.neucom.2004.11.042.

Li, Y.; Wang, G.; Ji, X.; Xiang, Y.; and Fox, D. 2018. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 683–698.

Liu, Y.; Prabhakaran, B.; and Guo, X. 2012. Point-based manifold harmonics. *IEEE Transactions on Visualization and Computer Graphics* 18(10): 1693–1703. ISSN 10772626. doi:10.1109/TVCG.2011.152.

Manuelli, L.; Li, Y.; Florence, P.; and Tedrake, R. 2020. Keypoints into the Future: Self-Supervised Correspondence in Model-Based Reinforcement Learning. *Conference on Robot Learning* URL http://arxiv.org/abs/2009.05085.

Ovsjanikov, M.; Sun, J.; and Guibas, L. 2008. Global intrinsic symmetries of shapes. *Computer Graphics Forum* 27(5): 1341–1348. ISSN 14678659. doi:10.1111/j.1467-8659.2008.01273.x.

Periyasamy, A. S.; Schwarz, M.; and Behnke, S. 2019. Refining 6D Object Pose Predictions using Abstract Render-and-Compare. *arXiv preprint arXiv:1910.03412* .

Schmidt, T.; Newcombe, R.; and Fox, D. 2017. Self-Supervised Visual Descriptor Learning for Dense Correspondence. *IEEE Robotics and Automation Letters* 2(2): 420–427. ISSN 2377-3766. doi:10.1109/LRA.2016.2634089.

Sharp, N.; Soliman, Y.; and Crane, K. 2019. The Vector Heat Method. *ACM Transactions on Graphics* 38(3): 1–19. ISSN 07300301. doi:10.1145/3243651.

Shelhamer, E.; Long, J.; and Darrell, T. 2017. Fully Convolutional Networks for Semantic Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 39(4): 640–651. ISSN 0162-8828. doi:10.1109/TPAMI.2016.2572683. URL https://doi.org/10.1109/TPAMI.2016.2572683.

Sundaresan, P.; Grannen, J.; Thananjeyan, B.; Balakrishna, A.; Laskey, M.; Stone, K.; Gonzalez, J. E.; and Goldberg, K. 2020. Learning Rope Manipulation Policies Using Dense Object Descriptors Trained on Synthetic Depth Data (3). URL http://arxiv.org/abs/2003.01835.

Tenenbaum, J. B.; De Silva, V.; and Langford, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science* 290(5500): 2319–2323. ISSN 00368075. doi:10.1126/science.290.5500.2319.

Van Der Maaten, L. J. P.; Postma, E. O.; and Van Den Herik, H. J. 2009. Dimensionality Reduction: A Comparative Review. *Journal of Machine Learning Research* 10: 1–41. ISSN 0169328X. doi:10.1080/13506280444000102.

Vecerik, M.; Regli, J.-B.; Sushkov, O.; Barker, D.; Pevceviciute, R.; Rothörl, T.; Schuster, C.; Hadsell, R.; Agapito, L.; and Scholz, J. 2020. S3K: Self-Supervised Semantic Keypoints for Robotic Manipulation via Multi-View Consistency. *Conference on Robot Learning* URL http://arxiv.org/abs/2009.14711.

Wang, H.; Simari, P.; Su, Z.; and Zhang, H. 2014. Spectral global intrinsic symmetry invariant functions. *Proceedings - Graphics Interface* 209–215. ISSN 07135424.

Zakharov, S.; Shugurov, I.; and Ilic, S. 2019. Dpod: 6d pose object detector and refiner. In *Proceedings of the IEEE International Conference on Computer Vision*, 1941–1950.