

---

# Training Dense Object Nets: A Novel Approach

---

Anonymous Author(s)

Affiliation  
Address  
email

## Abstract

Our work proposes a novel framework that addresses the computational resource limitations associated with training Dense Object Nets (DON) while achieving robust and dense visual object descriptors. DON’s descriptors are known for their robustness to viewpoint and configuration changes, but training them requires computationally expensive image pairs with correspondence mapping. This limitation hampers dimensionality and robustness, thereby restricting object generalization. To overcome this, we introduce a synthetic augmentation data generation procedure and a novel deep learning architecture that produces denser visual descriptors with reduced computational demands. Notably, our framework eliminates the need for image-pair correspondence mapping and showcases its application in a robot-grasping pipeline. Experimental results demonstrate that our approach yields descriptors as robust as those generated by DON.

## 1 Introduction

The objectives of long-standing robotics and robotic manipulation are to create a general-purpose robot capable of carrying out practical activities like Chappie [1] or C-3PO [2]. While advancements have been made recently in adjacent domains, achieving this goal remains a work in progress. For instance, AlphaGo [3], a game-playing artificial intelligence system trained entirely on self-play, defeated the world’s best human Go player at the time. Subsequently, Silver et al. [4] developed artificial intelligence algorithms that mastered the game of chess, Go, World of Warcraft [5], and Shogi, surpassing human playing expertise. Most of these algorithms learn directly from visual data, such as gameplay recordings or online video streams, emphasizing the importance of visual data in AI. Meanwhile, the launch of AlexNet [6] in 2012 transformed the field of computer vision. Other visual tasks, such as semantic segmentation [7], object identification and recognition [8], and human pose estimation [9], have also witnessed significant gains in recent years. Significant breakthroughs have been made in robotics, ranging from self-driving cars to humanoid robots capable of performing complex tasks using cameras and other vision sensors. Despite these advancements, the most frequently used robotic manipulation systems have evolved slightly in the previous 30 years. Typical auto-factory robots continue to perform repetitive operations such as welding and painting, following a pre-programmed course with no feedback from the surroundings. If we want to increase the utility of our robots, we must move away from highly controlled settings and robots that perform repetitive actions with little feedback or adaptability capabilities. Liberating ourselves from these constraints of controlled settings-based manufacturing would allow us to enter new markets, as witnessed by the proliferation of firms [10] competing in the logistics domain.

The ideal object representation for robot grasping and manipulation tasks remains to be engineered today. Existing representations may not be suitable for complex tasks due to limited capabilities of understanding an object’s geometrical and structural information. In 2018, Florence et al. [11] introduced a novel visual object representation to the robotics community, terming it “dense visual object descriptors”. DON, an artificial intelligence framework proposed by Florence et al. [11]

39 produces dense visual object descriptors. In detail, the DON converts every pixel in the image  
 40 ( $I[u, v] \in \mathbb{R}^3$ ) to a higher dimensional embedding ( $I_D[u, v] \in \mathbb{R}^D$ ) such that  $D \in \mathbb{N}^+$  consuming  
 41 image-pair correspondences as input yielding pixelwise embeddings which are nothing but dense  
 42 local descriptors. The dense visual object descriptor generalizes an object up to a certain extent  
 43 and has been recently applied to rope manipulation [12], block manipulation [13], robot control  
 44 [14], fabric manipulation [15] and robot grasp pose estimation [16, 17]. Adrian et al. [17] further  
 45 demonstrated that DON can be trained on synthetic data and still generalize to real-world objects.  
 46 Furthermore, Adrian et al. [17] demonstrated that the quality of descriptors produced by the DON  
 47 framework depends on the higher or longer embedding dimension. We tried training the DON on  
 48 a computation device equipped with NVIDIA RTX A6000 GPU with 48GB VRAM. However, we  
 49 could not train the DON to produce a higher embedding dimension due to the limited VRAM. The  
 50 DON framework is computationally expensive, as shown in Table 1, and limits the user to generalize  
 51 objects to a certain extent making it difficult to use as a robot grasping pipeline in real-world logistics  
 52 and warehouse automation scenarios.

Table 1: Benchmark of DON framework trained on GPU with 48GB VRAM with 128 image-pair correspondences, batch size of 1 and “Pixelwise NTXENT Loss” [17] as a loss function.

GPU VRAM consumption to train DON				
Descriptor Dimension	3	8	16	32
VRAM Usage (GB)	9.377	13.717	20.479	30.067

53 Moreover, today’s Large Language Models (LLMs) consume vast computational resources, posing  
 54 significant concerns for the future. LLMs’ sheer size and complexity require massive computing  
 55 power, leading to excessive energy consumption and environmental harm. Researchers and experts  
 56 have raised alarm about these issues. As Bender et al. [18] highlighted, the environmental impact  
 57 of training and operating LLMs at scale is substantial, contributing to increased carbon emissions.  
 58 Furthermore, in their paper, Strubell et al. [19] emphasized how the resource-intensive nature of  
 59 LLMs exacerbates the digital divide, limiting access to advanced AI technologies for underprivileged  
 60 populations. The ethical implications of prioritizing computational power for LLMs over binding  
 61 domains have also been acknowledged by researchers such as Amodei et al. [20], stressing the  
 62 need for responsible resource allocation. Striking a balance between technological progress and  
 63 sustainability is essential, as echoed by various experts in the field. Drawing motivation from LLMs  
 64 impact of consumption of vast computational resources, we propose a novel framework to train and  
 65 extract dense visual object descriptors produced by DON, which is computationally efficient.<sup>1</sup>

## 66 2 Related Work

67 We are solely interested in computing dense visual object descriptors of an object. The DON training  
 68 strategy in [11] relies on the depth information for computing correspondences in an image pair  
 69 using camera intrinsics and pose information [21]. However, when employing consumer-grade depth  
 70 cameras for capturing the depth information, the depth cameras capture noisy depth in cases of  
 71 tiny, reflecting objects, which are common in industrial environments. In the meantime, Kupcsik  
 72 et al. [16] used Laplacian Eigenmaps [22] to embed a 3D object model into an optimally generated  
 73 embedding space acting as a target to train DON in a supervised fashion. The optimal embeddings  
 74 bring more domain knowledge by associating the 3D object model with image views. Kupcsik  
 75 et al. [16] efficiently apply it to smaller, texture-less and reflective objects by eliminating the need for  
 76 depth information. Kupcsik et al. [16] further, compare training strategies for producing 6D grasps  
 77 for industrial objects and show that a unique supervised training approach increases pick-and-place  
 78 resilience in industry-relevant tasks.

79 Florence [23] has found that the pixelwise contrastive loss function used to train DON might not  
 80 perform well if a computed correspondence is spatially inconsistent (analogously to the case of noisy  
 81 depth information). Further highlighting that the precision of contrastive-trained models can be sensi-  
 82 tive to the relative weighting between positive-negative sampled pixels. Instead, the Florence [23]  
 83 introduces a new continuous sampling-based loss function called “Pixelwise Distribution Loss”. The

<sup>1</sup>Dataset and codebase link: [https://github.com/KanishkNavale/training\\_don\\_while\\_not\\_training\\_don](https://github.com/KanishkNavale/training_don_while_not_training_don)

84 pixelwise distribution loss is much more effective as it is a smooth continuous pixel space sampling  
85 method compared to the discrete pixel space sampling method based on pixelwise contrastive loss.  
86 The pixelwise distribution loss regresses probability distribution heatmaps to minimize the divergence  
87 between the predicted heatmap and the ground truth heatmap mitigating errors in correspondences.  
88 Furthermore, the pixelwise distribution loss does not need non-matching correspondences com-  
89 pared to the pixelwise contrastive loss. Differently, Hadjivelichkov and Kanoulas [24] extends the  
90 DON training using semantic correspondences between objects in multi-object or cluttered scenes  
91 overcoming the limitations of [21, 22]. The authors, Hadjivelichkov and Kanoulas [24] employ  
92 offline unsupervised clustering based on confidence in object similarities to generate hard and soft  
93 correspondence labels. The computed hard and soft labels lead DON in learning class-aware dense  
94 object descriptors, introducing hard and soft margin constraints in the proposed pixelwise contrastive  
95 loss to train DON. Further eliminating the need for camera pose and intrinsic information along  
96 with depth information to compute correspondences in an image pair, Yen-Chen et al. [25] used  
97 NeRF [26] to train DON. The NeRF [26] recreates a 3D scene from a sequence of images captured  
98 by the smartphone camera. The correspondences are extracted from the synthetically reconstructed  
99 scene to train DON. Recently, based on SIMCLR inspired frameworks [27, 28], Adrian et al. [17]  
100 introduced similar architecture and another novel loss function called Pixelwise NTXent loss to train  
101 DON more robustly. The Pixelwise NTXent loss consumes synthetic correspondences independent of  
102 depth cameras computed from image augmentations to train DON. Adrian et al.’s experiments show  
103 that the novel loss function is invariant to the batch size. Additionally adopted “*PCK@k*” metric  
104 has been adopted as in proceedings [29, 30] to evaluate and benchmark DON on cluttered scenes  
105 previously not benchmarked.

106 In the proposed framework, we do not use any loss functions in [11, 23, 16, 17, 24, 25] to train DON  
107 however we adopt the network architecture from [11] and train on the task of the KeypointNet[31,  
108 32].

### 109 **3 Methodology**

#### 110 **3.1 Dataset Engineering**

111 We have chosen the cap object for creating a synthetic dataset as the cap mesh models are readily  
112 available in the “Shapenet” library [33] as it contains rich object information, including textures.  
113 Furthermore, we choose five cap models from the Shapenet library and use Blenderproc [34] to  
114 generate the synthetic dataset. For each cap model, we save one RGB image, mask and depth  
115 from the synthetic scene. Additionally, we employ synthetic augmentations as proposed in [17]  
116 to synthetically spatial augment the cap’s position and rotation in an image, including background  
117 randomization using Torchvision [35] library. To generate camera poses for different viewpoints, an  
118 augmented image-pair is sampled randomly, additionally image-pair correspondences are computed<sup>1</sup>  
119 as illustrated in the Figure 1. Using depth information, we project the computed correspondences to  
120 the camera frame and compute the relative transformation between two camera-frame coordinates of  
121 the correspondences using Kabsch’s transformation [36]. Moreover, mask and depth images is not  
122 used during inference.



Figure 1: Depiction of image synthetic spatial augmentation and correspondences mapping in an image-pair. The colored encoded dots in the figure represents correspondences in an image-pair.

---

<sup>1</sup>GitHub Link:  
<https://github.com/KanishkNavale/Mapping-Synthetic-Correspondences-in-an-Image-Pair>

123 **3.2 Framework & Mining Strategy**

124 As a backbone, we employ ResNet-34 architecture [37]. We preserve the last convolution layer and  
 125 remove the pooling and linear layers. The backbone downsamples the RGB image  $I_{RGB} \in \mathbb{R}^{H \times W \times 3}$   
 126 to dense features  $I_d \in \mathbb{R}^{h \times w \times D}$  such that  $h \ll H, w \ll W$  and  $D \in \mathbb{N}^+$ . We upsample the dense  
 127 features from the identity layer (being identical to the last convolution layer in the backbone) as  
 128 illustrated in the Figure 2 in page 4 as follows:

$$f_U : I \in \mathbb{R}^{h \times w \times D} \rightarrow I_D \in \mathbb{R}^{H \times W \times D}. \quad (1)$$

129 The upsampled dense features are extracted and treated as dense visual local descriptors produced  
 130 from the DON. In otherwords we extract or mine the representations from the backbone. Similarly as  
 131 in [31], we stack spatial-probability regressing layer and depth regressing layer on top of the identity  
 132 layer to predict  $N \in \mathbb{N}^+$  number of keypoint's spatial-probability as follows:

$$f_S : I_d \in \mathbb{R}^{h \times w \times D} \rightarrow I_s^N \in \mathbb{R}^{h \times w \times N}, \quad (2)$$

133 and depth as follows:

$$f_D : I_d \in \mathbb{R}^{h \times w \times D} \rightarrow I_{\hat{d}} \in \mathbb{R}^{h \times w \times N}. \quad (3)$$

134 We incorporate continuous sampling method  $f_E$  from [23, 31] to convert the upsampled predicted  
 135 spatial-probability and depth of a keypoint to spatial-depth expectation as follows:

$$f_E \circ g_E : [I_s, I_{\hat{d}}] \rightarrow [u, v, d]^T \in \mathbb{R}^3, \text{ where } g_E : I \in \mathbb{R}^{h \times w \times N} \rightarrow I \in \mathbb{R}^{H \times W \times N}. \quad (4)$$

136 Furthermore, we train the framework in a twin architecture fashion as proposed in [27, 28, 11, 23, 16,  
 137 17, 24, 25] on the modified KeypointNet task.

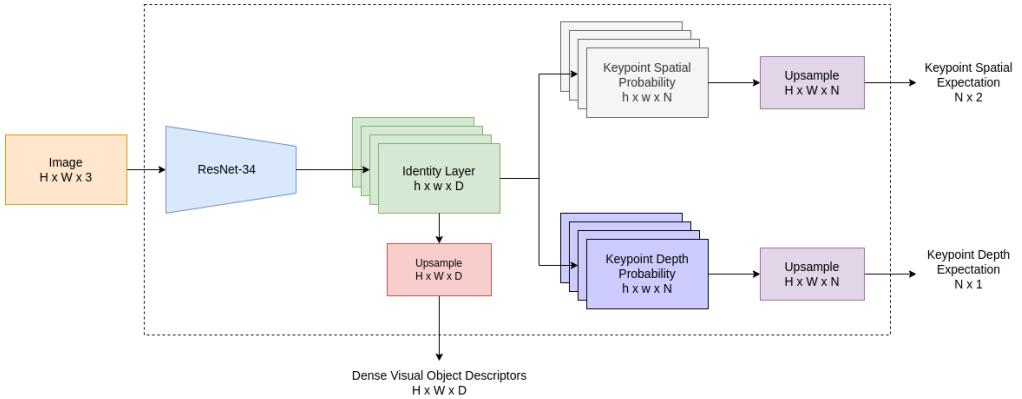


Figure 2: Illustration of the novel framework designed to compute and seamlessly extract dense visual object descriptors efficiently. During inference, we extract dense visual object descriptors directly from the network and ignore predicted spatial-depth expectations of the keypoints.

138 **3.3 Loss Functions**

139 For training, we directly adopt silhouette consistency loss ( $\mathcal{L}_{obj}$ ), variance loss ( $\mathcal{L}_{var}$ ) and separation  
 140 loss ( $\mathcal{L}_{sep}$ ) functions from [31] to train the network on the keypoint prediction task. However, we  
 141 modify the multi-view consistent loss and relative pose estimation loss. In the case of multi-view  
 142 consistency loss we project the predicted spatial-depth expectation using camera intrinsics as follows:  
 143

$$X_{cam} \in \mathbb{R}^{3 \times 1} = \mathcal{I}_{cam}^{-1} [u, v, 1.0]^T \times d, \text{ where } \mathcal{I}_{cam} \in \mathbb{R}^{3 \times 3} \text{ and } u, v, d \in \mathbb{R}^+. \quad (5)$$

144 Furthermore, we project the camera coordinates of the keypoints from one camera viewpoint to  
 145 another camera viewpoint using relative transformation supplied from the synthetic augmentation  
 146 procedure as follows:

$$\mathcal{L}_{mvc} \in \mathbb{R} = \mathcal{H}(\hat{X}_{cam}^B, \mathcal{T}_{A \rightarrow B} \hat{X}_{cam}^A), \text{ where } \hat{X}_{cam} = [X_{cam}, 1.0]^T \in \mathbb{R}^{4 \times 1}, \quad (6)$$

147 In Equation 6,  $\mathcal{T}_{A \rightarrow B} \in SE(3)$  is a Special Euclidean Group [38] which is relative transformation  
 148 from camera-frame  $A$  to camera-frame  $B$ . We use Huber loss  $\mathcal{H}$  as it produces smoother gradients  
 149 for framework optimization. Furthermore, we do not discard the relative transformation information  
 150 to calculate the relative pose loss as suggested in [31]. Moreover, being influenced from [32] we  
 151 modified the relative pose loss as follows:

$$\mathcal{L}_{pose} = \|log(\mathcal{T}_{truth}^\dagger \mathcal{T}_{pred})\|, \text{ where } log : SE(3) \rightarrow \mathfrak{se}(3) \text{ and } \mathcal{T}^\dagger = \begin{bmatrix} R^T & -R^T t \\ 0^T & 1 \end{bmatrix}. \quad (7)$$

### 152 3.4 Robot Grasping Pipeline

153 To use the proposed framework as a robot grasping pipeline, we extract dense visual object descriptors  
 154 from the network and store one single descriptor of objects in a database manually for now. During  
 155 inference, we extract dense visual object descriptors from the network and query the descriptor from  
 156 the database to find the closest match as follows:

$$\mathbb{E}[u^*, v^*]_d = \underset{u, v}{\operatorname{argmin}} \exp - \left( \frac{\|I_D[u, v] - d\|}{\exp(t)} \right)^2, \text{ where } \|I_D[u, v] - d\| \in \mathbb{R}^{H \times W}. \quad (8)$$

157 Where  $t \in \mathbb{R}$  controls the kernel width influencing the search space to compute the optimal spatial  
 158 expectation  $\mathbb{E}[u^*, v^*]_d$  of the query descriptor  $d \in \mathbb{R}^D$  in the descriptor image  $I_D \in \mathbb{R}^{H \times W \times D}$ .  
 159 The computed spatial expectation is projected to the robot frame using camera intrinsics and poses  
 160 to perform a pinch grasp. Furthermore, the Franka Emika 7-DOF robot manipulator with two jaw  
 161 gripper and wrist-mounted Intel Realsense D435 camera is used as a testing setup as illustrated in  
 162 Figure 3.



Figure 3: Illustration of the robot grasping pipeline setup. In the image, the robot is highlighted in red, the caps in green, and the camera in blue.

## 163 4 Experiments & Results

### 164 4.1 Dense Object Nets

165 We implemented training and benchmarking using “PyTorch-Lightning”[39] and “PyTorch”[40]  
 166 libraries. Furthermore, we employ ADAM[41] optimizer to optimize the model for 2500 epochs  
 167 with a learning rate of  $\alpha = 3 \times 10^{-4}$ ,  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$  with weight decay  $\eta = 10^{-4}$  to  
 168 benchmark the DON with Pixelwise NT-Xent loss as in [17] with a fixed batch size of 1 and 128  
 169 image-pair correspondences. As per the benchmarking results in Table 2, the descriptor’s robustness  
 170 increases as the descriptor’s dimension gets longer.

Table 2: Benchmark of DON framework for GPU consumption and  $AUC \pm \sigma$  for  $PCK@k$ ,  $\forall k \in [1, 100]$  metric.

DON benchmark				
Descriptor Size ( $D$ )	3	8	4	32
AUC for $PCK@k$	$0.922 \pm 0.006$	$0.933 \pm 0.011$	$0.948 \pm 0.012$	$0.953 \pm 0.008$
VRAM Usage (GB)	9.377	13.717	20.479	30.067

171 The  $AUC \pm \sigma$  for  $PCK@k, \forall k \in [1, 100]$  is computed with 256 image-pair correspondences and  
 172 the metrics mean and std. deviation is calculated from benchmarking 3 DON models trained for a  
 173 single descriptor dimension. We could not train the descriptor dimension of 64 and 128 due to the  
 174 limited VRAM. Furthermore, to inspect the results of trained DON, an interface is built using the  
 175 PyGame library [42] to visualize the results of the trained DON. The mouse pointer in the image  
 176 space is mapped to the pixel, and the descriptor at that pixel is queried in another image-descriptor  
 177 space. We further use the spatial probability of the descriptor to visualize the queried descriptor  
 178 in the image space using Equation 8 Identify if there are any multi-modal spatial activations in the  
 179 descriptor spaces and none, as shown in Figure 4.

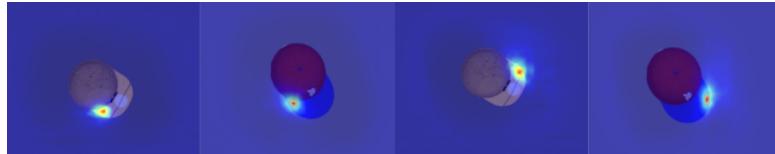


Figure 4: Depiction of the spatial probability heatmaps of the descriptor in the image space. We set the temperature in the Equation 8 to 1.1 and render the spatial probability heatmaps in the interface. The first and second image from the left and the right highlights the semantically equivalent descriptors in the image space.

## 180 4.2 Our Framework

181 To train our framework, we employ an ADAM optimizer to optimize the model for 2500 epochs with  
 182 a learning rate of  $\alpha = 1 \times 10^{-3}$ ,  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$  with no weight decay. We further use a  
 183 fixed batch size of 1 and the StepLR scheduler with a step size 2500 and a gamma of 0.9 to train the  
 184 model with all the loss weights to 1.0 except variance loss weight to  $1 \times 10^{-3}$ . At first, we trained our  
 185 model with 16 keypoints with a margin of 10 pixels as a hyperparameter for the separation loss, and  
 186 later, we trained the models with 128 keypoints with a margin of 2 pixels. We chose 128 keypoints  
 187 such that it is inline to the notion that DON is benchmarked with 128 image-pair correspondences.

Table 3: Benchmark of our framework for GPU consumption and  $AUC \pm \sigma$  for  $PCK@k, \forall k \in [1, 100]$  metric.

Our framework with 128 keypoints		
Descriptor Size ( $D$ )	$AUC \pm \sigma$ for $PCK@k$	VRAM Usage (GB)
3	$0.914 \pm 0.009$	4.763
8	4.913	4.785
16	4.913	4.832
32	4.913	4.872
64	4.913	4.913
128	4.913	5.409
256	4.913	6.551
512	4.913	7.915

## 188 4.3 Robot Grasping Pipeline

189 For the robot grasping pipeline, we trained our framework with actual caps. As the synthetic data  
 190 generation only needs mask and depth information, we could create a mask in no time. Additionally,  
 191 while training the framework, we do not need the actual real-world depth information as it computes  
 192 its own. We later extracted the dense visual local descriptors from the framework. We visually  
 193 inspected for any inconsistencies in the descriptor space, as shown in Figure 5, and found it consistent.  
 194 Furthermore, we did not use the models trained on the synthetic dataset, as the representations were  
 195 inconsistent with the real caps.

196 For robot grasping, a descriptor is picked from the descriptor space and queried in real-time such that  
 197 robot can pinch-grasp the object. We could successfully grasp the caps with the robot, as shown in  
 198 Figure 6.

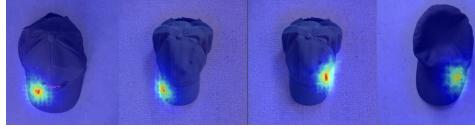


Figure 5: Visual inspection of the dense visual descriptors space of the real caps.

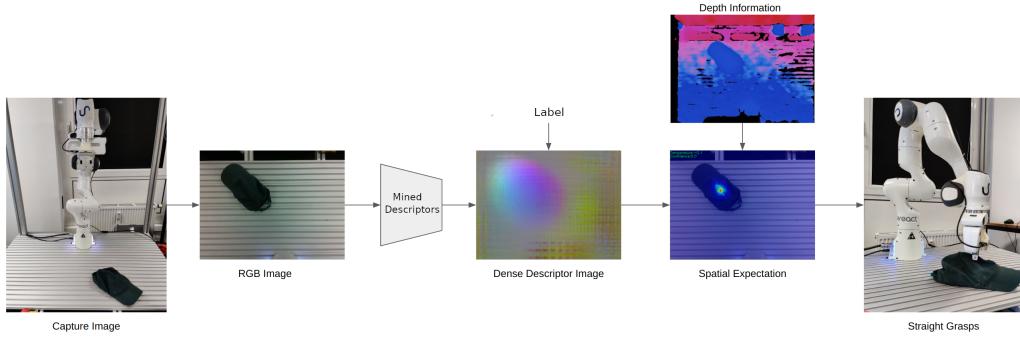


Figure 6: Depiction of the straight robot grasping pipeline.

199 As our framework inertly regresses keypoints on the object, we could use it as an alternative approach  
 200 to grasp the caps by computing the pose generated by the keypoints considering the actual depth  
 201 information instead of network-regressed depth information. We extract the spatial probability of  
 202 each keypoint from the framework and deactivate spatial probabilities where the depth information is  
 203 missing, as the depth image from the camera is noisy. Furthermore, the spatial expectations of the  
 204 keypoints are projected to the camera frame to calculate a 6D pose in the camera frame. The 6D pose  
 205 is transformed in the robot frame to perform an aligned grasp, as shown in Figure 7.

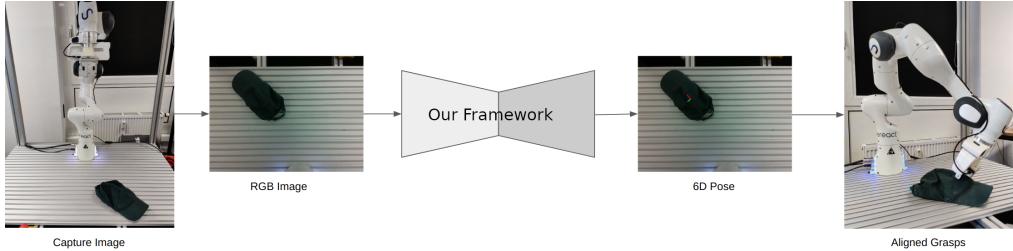


Figure 7: Illustration of the aligned robot grasping pipeline.

206 We did not evaluate the robot grasping pipeline.

## 207 5 Conclusion

208 We present a novel framework for mining dense visual object descriptors without explicitly training  
 209 DON. By leveraging synthetic augmentation data generation and a novel deep learning architecture,  
 210 our approach produces robust and denser visual local descriptors while consuming up to 86.67%  
 211 lesser computational resources than the originally proposed framework. Furthermore, it eliminates  
 212 the additional task of computing a large number of image-pair correspondences. We demonstrate the  
 213 application of our framework as a robot-grasping pipeline in two methodologies, one of which our  
 214 framework demonstrates its capabilities to produce object-specific 6D poses for robot grasping.

215 **6 Future Work**

216 The framework will be extended to produce multi-object dense visual descriptors in cluttered scenes.  
217 Furthermore, we will start incorporating ROI layers in the framework and adding additional object  
218 classification loss functions.

219 **References**

- 220 [1] N. Blomkamp, H. Zimmer, and S. Kinberg. *Chappie*. Sony Pictures Home Entertainment, 2015.
- 221 [2] G. Lucas and S. Ulstein. *Star wars*. 20th Century-Fox, 1977.
- 222 [3] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran,  
223 T. Graepel, et al. “A general reinforcement learning algorithm that masters chess, shogi, and Go through  
224 self-play”. In: *Science* 362.6419 (2018), pp. 1140–1144.
- 225 [4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I.  
226 Antonoglou, V. Panneershelvam, M. Lanctot, et al. “Mastering the game of Go with deep neural networks  
227 and tree search”. In: *nature* 529.7587 (2016), pp. 484–489.
- 228 [5] B. Entertainment. *World of warcraft*. Insight Editions, Division Of Palac, 2013.
- 229 [6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural  
230 networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90.
- 231 [7] J. Long, E. Shelhamer, and T. Darrell. “Fully convolutional networks for semantic segmentation”. In:  
232 *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.
- 233 [8] K. He, G. Gkioxari, P. Dollár, and R. Girshick. “Mask r-cnn”. In: *Proceedings of the IEEE international  
234 conference on computer vision*. 2017, pp. 2961–2969.
- 235 [9] R. A. Güler, N. Neverova, and I. Kokkinos. “Densepose: Dense human pose estimation in the wild”. In:  
236 *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7297–7306.
- 237 [10] sereact. “AI Robots for Production. Today”. <https://sereact.ai/en>.
- 238 [11] P. R. Florence, L. Manuelli, and R. Tedrake. “Dense object nets: Learning dense visual object descriptors  
239 by and for robotic manipulation”. In: *arXiv preprint arXiv:1806.08756* (2018).
- 240 [12] P. Sundaresan, J. Grannen, B. Thananjeyan, A. Balakrishna, M. Laskey, K. Stone, J. E. Gonzalez, and K.  
241 Goldberg. “Learning Rope Manipulation Policies Using Dense Object Descriptors Trained on Synthetic  
242 Depth Data”. In: *CoRR* abs/2003.01835 (2020). arXiv: 2003.01835.
- 243 [13] C.-Y. Chai, K.-F. Hsu, and S.-L. Tsao. “Multi-step Pick-and-Place Tasks Using Object-centric Dense  
244 Correspondences”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems  
245 (IROS)*. 2019, pp. 4004–4011. doi: 10.1109/IROS40897.2019.8968294.
- 246 [14] P. Florence, L. Manuelli, and R. Tedrake. “Self-supervised correspondence in visuomotor policy learning”.  
247 In: *IEEE Robotics and Automation Letters* 5.2 (2019), pp. 492–499.
- 248 [15] A. Ganapathi et al. “Learning Dense Visual Correspondences in Simulation to Smooth and Fold Real  
249 Fabrics”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 11515–  
250 11522. doi: 10.1109/ICRA48506.2021.9561980.
- 251 [16] A. Kupcsik, M. Spies, A. Klein, M. Todescato, N. Waniek, P. Schillinger, and M. Bürger. “Supervised  
252 Training of Dense Object Nets using Optimal Descriptors for Industrial Robotic Applications”. In: *arXiv  
253 preprint arXiv:2102.08096* (2021).
- 254 [17] D. B. Adrian, A. G. Kupcsik, M. Spies, and H. Neumann. “Efficient and Robust Training of Dense  
255 Object Nets for Multi-Object Robot Manipulation”. In: *2022 International Conference on Robotics and  
256 Automation (ICRA)*. IEEE. 2022, pp. 1562–1568.
- 257 [18] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. “On the Dangers of Stochastic Par-  
258rots: Can Language Models Be Too Big?” In: *Proceedings of the 2021 ACM conference on fairness,  
259 accountability, and transparency*. 2021, pp. 610–623.
- 260 [19] E. Strubell, A. Ganesh, and A. McCallum. “Energy and policy considerations for deep learning in NLP”.  
261 In: *arXiv preprint arXiv:1906.02243* (2019).
- 262 [20] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. “Concrete problems in AI  
263 safety”. In: *arXiv preprint arXiv:1606.06565* (2016).
- 264 [21] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press,  
265 2003.
- 266 [22] M. Belkin and P. Niyogi. “Laplacian eigenmaps for dimensionality reduction and data representation”. In:  
267 *Neural computation* 15.6 (2003), pp. 1373–1396.
- 268 [23] P. R. Florence. “Dense visual learning for robot manipulation”. PhD thesis. Massachusetts Institute of  
Technology, 2020.
- 269 [24] D. Hadjivelichkov and D. Kanoulas. “Fully Self-Supervised Class Awareness in Dense Object Descrip-  
270 tors”. In: *5th Annual Conference on Robot Learning*. 2021.

- 272 [25] L. Yen-Chen, P. Florence, J. T. Barron, T.-Y. Lin, A. Rodriguez, and P. Isola. *NeRF-Supervision: Learning*  
 273 *Dense Object Descriptors from Neural Radiance Fields*. 2022. DOI: 10.48550/ARXIV.2203.01913.
- 274 [26] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. “Nerf: Representing  
 275 scenes as neural radiance fields for view synthesis”. In: *Communications of the ACM* 65.1 (2021), pp. 99–  
 276 106.
- 277 [27] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. “A simple framework for contrastive learning of visual  
 278 representations”. In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.
- 279 [28] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. “Barlow twins: Self-supervised learning via  
 280 redundancy reduction”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12310–  
 281 12320.
- 282 [29] C.-Y. Chai, K.-F. Hsu, and S.-L. Tsao. “Multi-step pick-and-place tasks using object-centric dense  
 283 correspondences”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.  
 284 IEEE. 2019, pp. 4004–4011.
- 285 [30] M. E. Fathy, Q.-H. Tran, M. Z. Zia, P. Vernaza, and M. Chandraker. “Hierarchical metric learning and  
 286 matching for 2d and 3d geometric correspondences”. In: *Proceedings of the european conference on*  
 287 *computer vision (ECCV)*. 2018, pp. 803–819.
- 288 [31] S. Suwajanakorn, N. Snavely, J. J. Tompson, and M. Norouzi. “Discovery of latent 3d keypoints via  
 289 end-to-end geometric reasoning”. In: *Advances in neural information processing systems* 31 (2018).
- 290 [32] W. Zhao, S. Zhang, Z. Guan, W. Zhao, J. Peng, and J. Fan. “Learning deep network for detecting 3d  
 291 object keypoints and 6d poses”. In: *Proceedings of the IEEE/CVF Conference on computer vision and*  
 292 *pattern recognition*. 2020, pp. 14134–14142.
- 293 [33] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song,  
 294 H. Su, et al. “Shapenet: An information-rich 3d model repository”. In: *arXiv preprint arXiv:1512.03012*  
 295 (2015).
- 296 [34] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and  
 297 H. Katam. “Blenderproc”. In: *arXiv preprint arXiv:1911.01911* (2019).
- 298 [35] S. Marcel and Y. Rodriguez. “Torchvision the machine-vision package of torch”. In: *Proceedings of the*  
 299 *18th ACM international conference on Multimedia*. 2010, pp. 1485–1488.
- 300 [36] W. Kabsch. “A solution for the best rotation to relate two sets of vectors”. In: *Acta Crystallographica*  
 301 *Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography* 32.5 (1976), pp. 922–  
 302 923.
- 303 [37] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: (2016), pp. 770–  
 304 778.
- 305 [38] W. P. Thurston. “Three-Dimensional Geometry and Topology, Volume 1”. In: *Three-Dimensional Geome-*  
 306 *try and Topology, Volume 1*. Princeton university press, 2014.
- 307 [39] W. A. Falcon. “Pytorch lightning”. In: *GitHub* 3 (2019).
- 308 [40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein,  
 309 L. Antiga, et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in*  
 310 *neural information processing systems* 32 (2019).
- 311 [41] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint*  
 312 *arXiv:1412.6980* (2014).
- 313 [42] S. Kelly. “Basic introduction to pygame”. In: *Python, PyGame and Raspberry Pi Game Development*.  
 314 Springer, 2016, pp. 59–65.