
Training Dense Object Nets: A Novel Approach

Anonymous Author(s)

Affiliation
Address
email

Abstract

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

13

1 Introduction

14 Creating a general-purpose robotic system capable of performing practical tasks, such as those
15 portrayed in movies like Chappie [1] or C-3PO [2], is a significant challenge in robotics. While
16 advancements have been made in related domains, achieving this goal remains an ongoing endeavour.
17 Recent artificial intelligence (AI) breakthroughs, particularly in deep learning, have demonstrated
18 remarkable capabilities. For instance, AlphaGo [3], an AI system trained entirely on self-play,
19 defeated the world's best human Go player at the time. Subsequent algorithms, such as those
20 developed by Silver et al. [4], have mastered complex games like chess, Go, World of Warcraft [5],
21 and Shogi, surpassing human expertise. These achievements underscore the importance of visual
22 data in deep learning, as these algorithms learn directly from visual inputs like gameplay recordings
23 or online video streams. Additionally, the introduction of AlexNet [6] in 2012 has revolutionized
24 computer vision, leading to significant progress in tasks like semantic segmentation [7], object
25 identification and recognition [8], and human pose estimation [9].
26 Significant strides have also been made in robotics, including developing self-driving cars and
27 humanoid robots capable of complex tasks using cameras and vision sensors. Integration of AI
28 models, such as ChatGPT [10] and PaLM [11], has further enhanced the capabilities of robots.
29 ChatGPT, developed by OpenAI, improves human-robot interactions by generating coherent and
30 contextually relevant responses, enabling robots to engage in meaningful conversations and provide
31 informative answers. Palm-E, developed by Stanford University, combines computer vision and deep
32 reinforcement learning, enabling robots to perceive and manipulate objects in their environment.
33 These AI models hold significant potential to enhance the dexterity and adaptability of robots, opening
34 up new possibilities across various industries.
35 However, developing and deploying these sophisticated AI models, particularly Large Language
36 Models (LLMs) like ChatGPT and Palm-E, present their own challenges. LLMs' sheer size and
37 complexity demand vast computational resources, raising concerns about excessive energy consump-
38 tion and environmental impact. Researchers such as Bender et al. [12] and Strubell et al. [13] have

39 raised alarms about the substantial carbon emissions associated with training and operating LLMs at
40 scale, as well as the exacerbation of the digital divide due to the resource-intensive nature of LLMs.
41 Responsible resource allocation and addressing the ethical implications of prioritizing computational
42 power for LLMs are crucial for balancing technological progress and sustainability.

43 Currently, in robotics, typical industrial robots perform repetitive operations based on pre-programmed
44 instructions, finding the ideal object representation for grasping and manipulation tasks still needs
45 to be answered. Existing representations may be unable to understand an object's geometrical
46 and structural information, rendering them unsuitable for complex tasks In recent work, Florence
47 et al. [14] introduced a novel visual object representation termed "dense visual object descriptors" to
48 the robotics community. This representation, generated by the Dense Object Nets (DON) framework,
49 converts each pixel in an image ($I[u, v] \in \mathbb{R}^3$) into a higher-dimensional embedding ($I_D[u, v] \in \mathbb{R}^D$)
50 such that $D \in \mathbb{N}^+$, using image-pair correspondences as input. These dense visual object descriptors
51 provide a generalized representation of objects to a certain extent.

52 The DON framework has shown promise in various domains, including rope manipulation [15], block
53 manipulation [16], robot control [17], fabric manipulation [18], and robot grasp pose estimation [19,
54 20]. Adrian et al. [20] demonstrated that DON can be trained on synthetic data and still generalize
55 well to real-world objects. Furthermore, they highlighted the importance of the dimensionality of the
56 embedding in determining the quality of the descriptors produced by the DON framework.

57 In this paper, we address the challenges posed by the computationally intensive nature of AI and
58 propose a new framework for training DON in a computationally efficient manner. Our approach
59 aims to contribute to developing sustainable and efficient AI solutions for robotics, bridging the gap
60 between technological progress and environmental responsibility.

61 2 Related Work

62 The DON training strategy introduced in [14] relies on the depth information for computing cor-
63 respondences in an image-pair using camera intrinsics and pose information [21]. However, when
64 employing consumer-grade depth cameras for capturing the depth information, the depth cameras
65 capture noisy depth in cases of tiny, reflecting objects, which are common in industrial environments.
66 In the meantime, Kupcsik et al. [19] used Laplacian Eigenmaps [22] to embed a 3D object model
67 into an optimally generated embedding space acting as a target to train DON in a supervised fashion.
68 The optimal embeddings bring more domain knowledge by associating the 3D object model with
69 image views. Kupcsik et al. [19] efficiently apply it to smaller, texture-less and reflective objects by
70 eliminating the need for depth information. Kupcsik et al. [19] further, compare training strategies
71 for producing 6D grasps for industrial objects and show that a unique supervised training approach
72 increases pick-and-place resilience in industry-relevant tasks.

73 Florence [23] has found that the pixelwise contrastive loss function used to train DON might not
74 perform well if a computed correspondence is spatially inconsistent (analogously to the case of noisy
75 depth information). Further highlighting that the precision of contrastive-trained models can be sensi-
76 tive to the relative weighting between positive-negative sampled pixels. Instead, the Florence [23]
77 introduces a new continuous sampling-based loss function called "Pixelwise Distribution Loss". The
78 pixelwise distribution loss is much more effective as it is a smooth continuous pixel space sampling
79 method compared to the discrete pixel space sampling method based on pixelwise contrastive loss.
80 The pixelwise distribution loss regresses probability distribution heatmaps to minimize the divergence
81 between the predicted heatmap and the ground truth heatmap mitigating errors in correspondences.
82 Furthermore, the pixelwise distribution loss does not need non-matching correspondences com-
83 pared to the pixelwise contrastive loss. Differently, Hadjivelichkov and Kanoulas [24] extends the
84 DON training using semantic correspondences between objects in multi-object or cluttered scenes
85 overcoming the limitations of [21, 22]. The authors, Hadjivelichkov and Kanoulas [24] employ
86 offline unsupervised clustering based on confidence in object similarities to generate hard and soft
87 correspondence labels. The computed hard and soft labels lead DON in learning class-aware dense
88 object descriptors, introducing hard and soft margin constraints in the proposed pixelwise contrastive
89 loss to train DON. Further eliminating the need for camera pose and intrinsic information along
90 with depth information to compute correspondences in an image pair, Yen-Chen et al. [25] used
91 NeRF [26] to train DON. The NeRF [26] recreates a 3D scene from a sequence of images captured
92 by the smartphone camera. The correspondences are extracted from the synthetically reconstructed

93 scene to train DON. Recently, based on SIMCLR inspired frameworks [27, 28], Adrian et al. [20]
 94 introduced similar architecture and another novel loss function called Pixelwise NTXent loss to train
 95 DON more robustly. The Pixelwise NTXent loss consumes synthetic correspondences independent of
 96 depth cameras computed from image augmentations to train DON. Adrian et al.’s experiments show
 97 that the novel loss function is invariant to the batch size. Additionally adopted “*PCK@k*” metric
 98 has been adopted as in proceedings [29, 30] to evaluate and benchmark DON on cluttered scenes
 99 previously not benchmarked.

100 In the proposed framework, we do not use any loss functions in [14, 23, 19, 20, 24, 25] to train DON
 101 however we adopt the network architecture from [14] and train on the task of the KeypointNet[31,
 102 32].

103 3 Methodology

104 In this section, we outline the methodologies employed. Our approach encompasses synthetic dataset
 105 engineering, a novel framework, loss function modifications and a comprehensive grasping pipeline.
 106 Firstly, we focus on synthetic dataset engineering emphasizing spatial, colour and background
 107 augmentation. Secondly, we present a novel framework designed to reduce computational resource
 108 consumption with loss function modifications to optimize performance. Lastly, we introduce a
 109 comprehensive robot grasping pipeline, integrating all methodologies into a unified system.

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. We save one RGB image, mask, and depth for each cap model from the
 115 synthetic scene. Additionally, we employ synthetic augmentations as proposed in [20] to synthetically
 116 spatial augment the cap’s position and rotation in an image, including background randomization using
 117 Torchvision [35] library. An augmented image-pair is sampled randomly to generate camera poses for
 118 different viewpoints. Additionally, image-pair correspondences are computed ¹ as illustrated in the
 119 Figure 1. Using depth information, we project the computed correspondences to the camera frame and
 120 compute the relative transformation between two camera-frame coordinates of the correspondences
 121 using Kabsch’s transformation [36]. Moreover, mask and depth images are not used during inference.

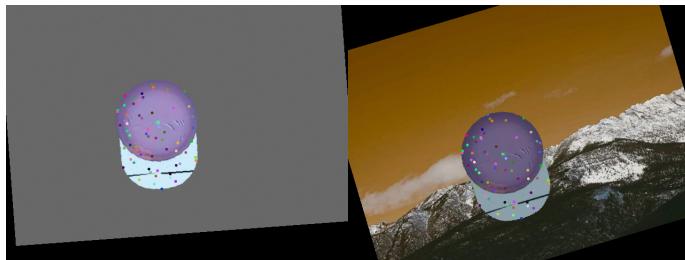


Figure 1: Depiction of image synthetic spatial augmentation and correspondences mapping in an image-pair. The colored encoded dots in the figure represents correspondences in an image-pair.

122 3.2 Framework & Mining Strategy

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

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

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

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

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

132 and depth as follows:

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

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

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

135 Furthermore, we train the framework in a twin architecture fashion as proposed in [27, 28, 14, 23, 19,
 136 20, 24, 25] on the modified KeypointNet task.

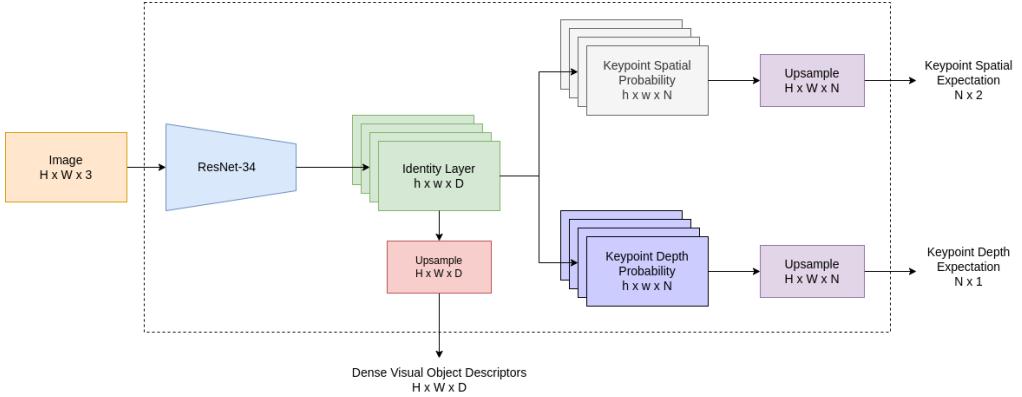


Figure 2: Illustration of the novel framework designed to compute and seamlessly extract dense visual object descriptors efficiently. During inference, we extract dense visual object descriptors directly from the network and ignore predicted spatial-depth expectations of the keypoints.

137 3.3 Loss Functions

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

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

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

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

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

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

151 **3.4 Robot Grasping Pipeline**

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

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

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

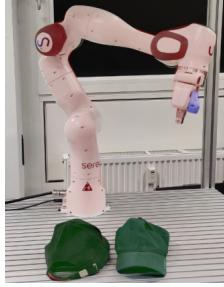


Figure 3: Illustration of the robot grasping pipeline setup. In the image, the robot is highlighted in red, the caps in green, and the camera in blue.

162 **4 Experiments & Results**

163 **4.1 Dense Object Nets**

164 We implemented training and benchmarking using “PyTorch-Lightning”[39] and “PyTorch”[40]
 165 libraries. Furthermore, we employ ADAM[41] optimizer to optimize the model for 2500 epochs
 166 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
 167 benchmark the DON with Pixelwise NT-Xent loss as in [20] with a fixed batch size of 1 and 128
 168 image-pair correspondences. As per the benchmarking results in Table 1, the descriptor’s robustness
 169 increases as the descriptor’s dimension gets longer.

Table 1: 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 ± 0.006	0.933 ± 0.011	0.948 ± 0.012	0.953 ± 0.008
VRAM Usage (GB)	9.377	13.717	20.479	30.067

170 The $AUC \pm \sigma$ for $PCK@k$, $\forall k \in [1, 100]$ is computed with 256 image-pair correspondences and
 171 the metrics mean and std. deviation is calculated from benchmarking 3 DON models trained for a
 172 single descriptor dimension. We could not train the descriptor dimension of 64 and 128 due to the
 173 limited VRAM. Furthermore, to inspect the results of trained DON, an interface is built using the
 174 PyGame library [42] to visualize the results of the trained DON. The mouse pointer in the image
 175 space is mapped to the pixel, and the descriptor at that pixel is queried in another image-descriptor
 176 space. We further use the spatial probability of the descriptor to visualize the queried descriptor
 177 in the image space using Equation 8 Identify if there are any multi-modal spatial activations in the
 178 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.

179 4.2 Our Framework

180 To train our framework, we employ an ADAM optimizer to optimize the model for 2500 epochs with
 181 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
 182 fixed batch size of 1 and the StepLR scheduler with a step size 2500 and a gamma of 0.9 to train the
 183 model with all the loss weights to 1.0 except variance loss weight to 1×10^{-3} . At first, we trained
 184 our model with 16 keypoints with a margin of 10 pixels as a hyperparameter for the separation loss,
 185 and later, we trained the models with 128 keypoints with a margin of 2 pixels.

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

Our framework with 16 keypoints				
Descriptor Size (D)	64	128	256	512
$AUC \pm \sigma$ for $PCK@k$	0.922 ± 0.006	0.933 ± 0.011	0.948 ± 0.012	0.953 ± 0.008
VRAM Usage (GB)	3.799	4.191	5.241	7.341
Our framework with 128 keypoints				
Descriptor Size (D)	64	128	256	512
$AUC \pm \sigma$ for $PCK@k$	0.922 ± 0.006	0.933 ± 0.011	0.948 ± 0.012	0.953 ± 0.008
VRAM Usage (GB)	4.913	5.409	6.551	7.915

186 4.3 Robot Grasping Pipeline

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



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

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

197 As our framework inertly regresses keypoints on the object, we could use it as an alternative approach
 198 to grasp the caps by computing the pose generated by the keypoints considering the actual depth
 199 information instead of network-regressed depth information. We extract the spatial probability of
 200 each keypoint from the framework and deactivate spatial probabilities where the depth information is

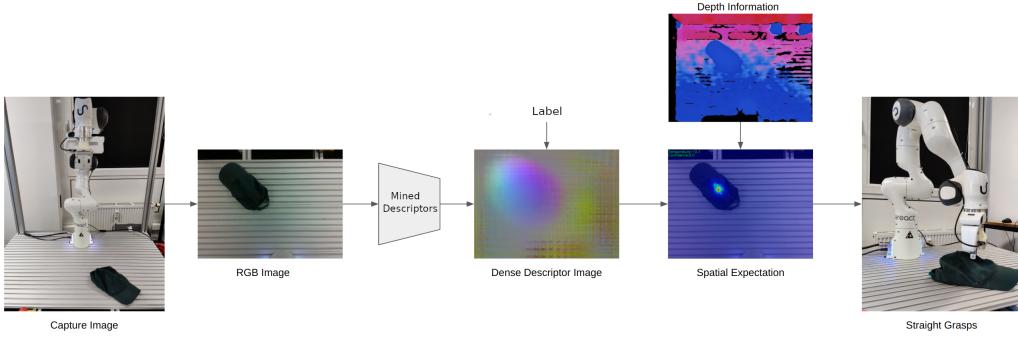


Figure 6: Depiction of the straight robot grasping pipeline.

201 missing, as the depth image from the camera is noisy. Furthermore, the spatial expectations of the
 202 keypoints are projected to the camera frame to calculate a 6D pose in the camera frame. The 6D pose
 203 is transformed in the robot frame to perform an aligned grasp, as shown in Figure 7.

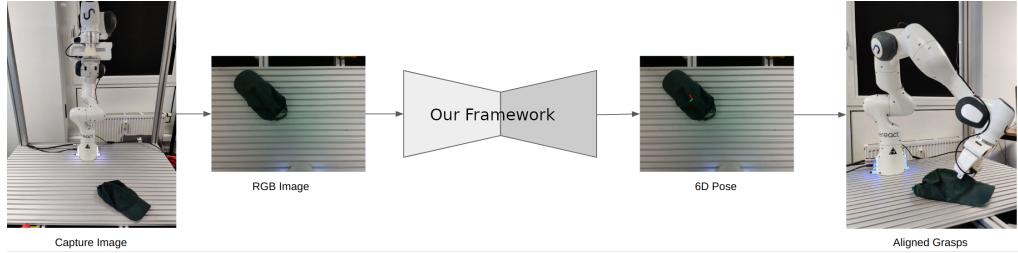


Figure 7: Illustration of the aligned robot grasping pipeline.

204 5 Conclusion

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

212 6 Future Work

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

216 References

- 217 [1] N. Blomkamp, H. Zimmer, and S. Kinberg. *Chappie*. Sony Pictures Home Entertainment, 2015.
- 218 [2] G. Lucas and S. Ulstein. *Star wars*. 20th Century-Fox, 1977.
- 219 [3] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran,
 220 T. Graepel, et al. “A general reinforcement learning algorithm that masters chess, shogi, and Go through
 221 self-play”. In: *Science* 362.6419 (2018), pp. 1140–1144.
- 222 [4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I.
 223 Antonoglou, V. Panneershelvam, M. Lanctot, et al. “Mastering the game of Go with deep neural networks
 224 and tree search”. In: *nature* 529.7587 (2016), pp. 484–489.

- 225 [5] B. Entertainment. *World of warcraft*. Insight Editions, Division Of Palac, 2013.
- 226 [6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural
227 networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90.
- 228 [7] J. Long, E. Shelhamer, and T. Darrell. “Fully convolutional networks for semantic segmentation”. In:
229 *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.
- 230 [8] K. He, G. Gkioxari, P. Dollár, and R. Girshick. “Mask r-cnn”. In: *Proceedings of the IEEE international
231 conference on computer vision*. 2017, pp. 2961–2969.
- 232 [9] R. A. Güler, N. Neverova, and I. Kokkinos. “Densepose: Dense human pose estimation in the wild”. In:
233 *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7297–7306.
- 234 [10] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL].
- 235 [11] A. Chowdhery et al. “PaLM: Scaling Language Modeling with Pathways”. In: *arxiv:2204.02311* (2022).
- 236 [12] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. “On the Dangers of Stochastic Par-
237 rots: Can Language Models Be Too Big?” In: *Proceedings of the 2021 ACM conference on fairness,
238 accountability, and transparency*. 2021, pp. 610–623.
- 239 [13] E. Strubell, A. Ganesh, and A. McCallum. “Energy and policy considerations for deep learning in NLP”.
240 In: *arXiv preprint arXiv:1906.02243* (2019).
- 241 [14] P. R. Florence, L. Manuelli, and R. Tedrake. “Dense object nets: Learning dense visual object descriptors
242 by and for robotic manipulation”. In: *arXiv preprint arXiv:1806.08756* (2018).
- 243 [15] P. Sundaresan, J. Grannen, B. Thananjeyan, A. Balakrishna, M. Laskey, K. Stone, J. E. Gonzalez, and K.
244 Goldberg. “Learning Rope Manipulation Policies Using Dense Object Descriptors Trained on Synthetic
245 Depth Data”. In: *CoRR* abs/2003.01835 (2020). arXiv: 2003.01835.
- 246 [16] C.-Y. Chai, K.-F. Hsu, and S.-L. Tsao. “Multi-step Pick-and-Place Tasks Using Object-centric Dense
247 Correspondences”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems
(IROS)*. 2019, pp. 4004–4011. DOI: 10.1109/IROS40897.2019.8968294.
- 249 [17] P. Florence, L. Manuelli, and R. Tedrake. “Self-supervised correspondence in visuomotor policy learning”.
250 In: *IEEE Robotics and Automation Letters* 5.2 (2019), pp. 492–499.
- 251 [18] A. Ganapathi et al. “Learning Dense Visual Correspondences in Simulation to Smooth and Fold Real
252 Fabrics”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 11515–
253 11522. DOI: 10.1109/ICRA48506.2021.9561980.
- 254 [19] A. Kupcsik, M. Spies, A. Klein, M. Todescato, N. Waniek, P. Schillinger, and M. Bürger. “Supervised
255 Training of Dense Object Nets using Optimal Descriptors for Industrial Robotic Applications”. In: *arXiv
256 preprint arXiv:2102.08096* (2021).
- 257 [20] D. B. Adrian, A. G. Kupcsik, M. Spies, and H. Neumann. “Efficient and Robust Training of Dense
258 Object Nets for Multi-Object Robot Manipulation”. In: *2022 International Conference on Robotics and
259 Automation (ICRA)*. IEEE. 2022, pp. 1562–1568.
- 260 [21] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press,
261 2003.
- 262 [22] M. Belkin and P. Niyogi. “Laplacian eigenmaps for dimensionality reduction and data representation”. In:
263 *Neural computation* 15.6 (2003), pp. 1373–1396.
- 264 [23] P. R. Florence. “Dense visual learning for robot manipulation”. PhD thesis. Massachusetts Institute of
265 Technology, 2020.
- 266 [24] D. Hadjiveliouchkov and D. Kanoulas. “Fully Self-Supervised Class Awareness in Dense Object Descrip-
267 tors”. In: *5th Annual Conference on Robot Learning*. 2021.
- 268 [25] L. Yen-Chen, P. Florence, J. T. Barron, T.-Y. Lin, A. Rodriguez, and P. Isola. *NeRF-Supervision: Learning
269 Dense Object Descriptors from Neural Radiance Fields*. 2022. DOI: 10.48550/ARXIV.2203.01913.
- 270 [26] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. “Nerf: Representing
271 scenes as neural radiance fields for view synthesis”. In: *Communications of the ACM* 65.1 (2021), pp. 99–
272 106.
- 273 [27] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. “A simple framework for contrastive learning of visual
274 representations”. In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.
- 275 [28] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. “Barlow twins: Self-supervised learning via
276 redundancy reduction”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12310–
277 12320.
- 278 [29] C.-Y. Chai, K.-F. Hsu, and S.-L. Tsao. “Multi-step pick-and-place tasks using object-centric dense
279 correspondences”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
280 IEEE. 2019, pp. 4004–4011.
- 281 [30] M. E. Fathy, Q.-H. Tran, M. Z. Zia, P. Vernaza, and M. Chandraker. “Hierarchical metric learning and
282 matching for 2d and 3d geometric correspondences”. In: *Proceedings of the european conference on
283 computer vision (ECCV)*. 2018, pp. 803–819.

- 284 [31] S. Suwajanakorn, N. Snavely, J. J. Tompson, and M. Norouzi. “Discovery of latent 3d keypoints via
285 end-to-end geometric reasoning”. In: *Advances in neural information processing systems* 31 (2018).
- 286 [32] W. Zhao, S. Zhang, Z. Guan, W. Zhao, J. Peng, and J. Fan. “Learning deep network for detecting 3d
287 object keypoints and 6d poses”. In: *Proceedings of the IEEE/CVF Conference on computer vision and*
288 *pattern recognition*. 2020, pp. 14134–14142.
- 289 [33] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song,
290 H. Su, et al. “Shapenet: An information-rich 3d model repository”. In: *arXiv preprint arXiv:1512.03012*
291 (2015).
- 292 [34] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and
293 H. Katam. “Blenderproc”. In: *arXiv preprint arXiv:1911.01911* (2019).
- 294 [35] S. Marcel and Y. Rodriguez. “Torchvision the machine-vision package of torch”. In: *Proceedings of the*
295 *18th ACM international conference on Multimedia*. 2010, pp. 1485–1488.
- 296 [36] W. Kabsch. “A solution for the best rotation to relate two sets of vectors”. In: *Acta Crystallographica*
297 *Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography* 32.5 (1976), pp. 922–
298 923.
- 299 [37] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: (2016), pp. 770–
300 778.
- 301 [38] W. P. Thurston. “Three-Dimensional Geometry and Topology, Volume 1”. In: *Three-Dimensional Geome-*
302 *try and Topology, Volume 1*. Princeton university press, 2014.
- 303 [39] W. A. Falcon. “Pytorch lightning”. In: *Github* 3 (2019).
- 304 [40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein,
305 L. Antiga, et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in*
306 *neural information processing systems* 32 (2019).
- 307 [41] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint*
308 *arXiv:1412.6980* (2014).
- 309 [42] S. Kelly. “Basic introduction to pygame”. In: *Python, PyGame and Raspberry Pi Game Development*.
310 Springer, 2016, pp. 59–65.