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]

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

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

To overcome the computation resource limitation to produce denser visual object descriptors, we propose a novel framework to train and extract dense visual object descriptors produced by DON, which is computationally efficient.

2 Related Work

We are solely interested in computing dense visual object descriptors of an object. The DON training strategy in [11] relies on the depth information for computing correspondences in an image pair using camera intrinsics and pose information [18]. However, when employing consumer-grade depth cameras for capturing the depth information, the depth cameras capture noisy depth in cases of tiny, reflecting objects, which are common in industrial environments. In the meantime, Kupcsik et al. [16] used Laplacian Eigenmaps [19] to embed a 3D object model into an optimally generated embedding space acting as a target to train DON in a supervised fashion. The optimal embeddings bring more domain knowledge by associating the 3D object model with image views. Kupcsik et al. [16] efficiently apply it to smaller, texture-less and reflective objects by eliminating the need for depth information. Kupcsik et al. [16] further, compare training strategies for producing 6D grasps for industrial objects and show that a unique supervised training approach increases pick-and-place resilience in industry-relevant tasks.

Florence [20] has found that the pixelwise contrastive loss function used to train DON might not perform well if a computed correspondence is spatially inconsistent (analogously to the case of noisy depth information). Further highlighting that the precision of contrastive-trained models can be sensitive to the relative weighting between positive-negative sampled pixels. Instead, the Florence [20] introduces a new continuous sampling-based loss function called "Pixelwise Distribution Loss". The pixelwise distribution loss is much more effective as it is a smooth continuous pixel space sampling method compared to the discrete pixel space sampling method based on pixelwise contrastive loss. The pixelwise distribution loss regresses probability distribution heatmaps to minimize the divergence between the predicted heatmap and the ground truth heatmap mitigating errors in correspondences. Furthermore, the pixelwise distribution loss does not need non-matching correspondences compared to the pixelwise contrastive loss. Differently, Hadjivelichkov and Kanoulas [21] extends the DON training using semantic correspondences between objects in multi-object or cluttered scenes overcoming the limitations of [18, 19]. The authors, Hadjivelichkov and Kanoulas [21] employ offline unsupervised clustering based on confidence in object similarities to generate hard and soft correspondence labels. The computed hard and soft labels lead DON in learning class-aware dense object descriptors, introducing hard and soft margin constraints in the proposed pixelwise contrastive loss to train DON. Further eliminating the need for camera pose and intrinsic information along

with depth information to compute correspondences in an image pair, Yen-Chen et al. [22] used NeRF [23] to train DON. The NeRF [23] recreates a 3D scene from a sequence of images captured by the smartphone camera. The correspondences are extracted from the synthetically reconstructed scene to train DON. Recently, based on SIMCLR inspired frameworks [24, 25], Adrian et al. [17] introduced similar architecture and another novel loss function called Pixelwise NTXent loss to train DON more robustly. The Pixelwise NTxent loss consumes synthetic correspondences independent of depth cameras computed from image augmentations to train DON. Adrian et al.'s experiments show that the novel loss function is invariant to the batch size. Additionally adopted "$PCK@k$" metric has been adopted as in proceedings [26, 27] to evaluate and benchmark DON on cluttered scenes previously not benchmarked.

In the proposed framework, we do not use any loss functions in [11, 20, 16, 17, 21, 22] to train DON however we adopt the network architecture from [11] and train on the task of the "KeypointNet"[28] with adaption of the loss functions proposed in [28, 29].

3 Methodology

3.1 Dataset Engineering

We have chosen the cap object for creating a synthetic dataset as the cap mesh models are readily available in the "Shapenet" library [30] as it contains rich object information, including textures. Furthermore, we choose five cap models from the Shapenet library and use Blenderproc [31] to generate the synthetic dataset. For each cap model, we save one RGB image, mask and depth from the synthetic scene. Additionally, we employ synthetic augmentations as proposed in [17] to synthetically spatial augment the cap's position and rotation in an image, including background randomization using Torchvision [32] library. To generate camera poses for different viewpoints, an augmented image-pair is sampled randomly, and 25 image-pair correspondences are computed [1] as illustrated in the Figure 1. Using depth information, we project the computed correspondences to the camera frame and compute the relative transformation between two camera-frame coordinates of the correspondences using Kabsch's transformation [33].

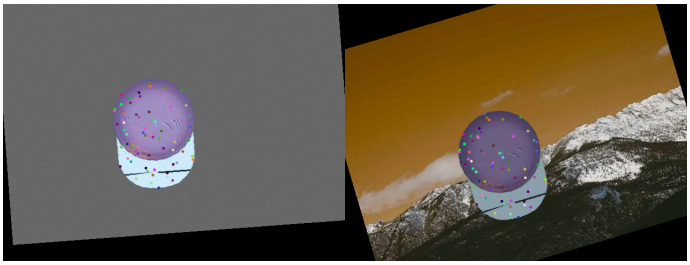

Figure 1: Depiction of image synthetic spatial augmentation and correspondences mapping in an image-pair. The colored encoded dots in the figure represents correspondences in an image-pair.

3.2 Framework & Mining Strategy

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

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

The upsampled dense features are extracted and treated as dense visual local descriptors produced from the DON. In otherwords we extract or mine the representations from the backbone. Similarly as

[1] GitHub Link:
`https://github.com/KanishkNavale/Mapping-Synthetic-Correspondences-in-an-Image-Pair`

in [28], we stack spatial-probability regressing layer and depth regressing layer on top of the identity layer to predict $N \in \mathbb{N}^+$ number of keypoint's spatial-probability as follows:

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

and depth as follows:

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

We incorporate continuous sampling method f_E from [20, 28] to convert the upsampled predicted spatial-probability and depth of a keypoint to spatial-depth expectation as follows:

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

Furthermore, we train the framework in a twin architecture fashion as proposed in [24, 25, 11, 20, 16, 17, 21, 22] on the KeypointNet task.

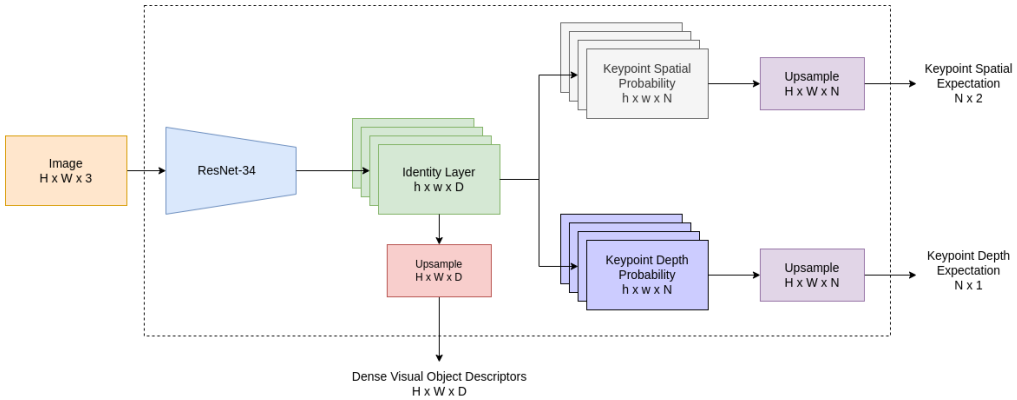

Figure 2: Illustration of the novel framework designed to compute and seamlessly extract dense visual object descriptors efficiently. During inference, we extract dense visual object descriptors directly from the network and ignore predicted spatial-depth expectations of the keypoints.

3.3 Loss Functions

For training, we directly adopt silhoutte consistency loss ($\mathcal{L}_{obj}$), variance loss ($\mathcal{L}_{var}$) and separation loss ($\mathcal{L}_{sep}$) functions from [28] to train the network on the keypoint prediction task. However, we modify the multi-view consistent loss and relative pose estimation loss. In the case of multi-view consistency loss we project the predicted spatial-depth expectation using camera intrinsics as follows:

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

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

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

In Equation 6, $\mathcal{T}_{A \rightarrow B} \in SE(3) \in \mathbb{R}^{4 \times 4}$ is a Special Euclidean Group [35] which is relative transformation from camera-frame A to camera-frame B. We use Huber loss $\mathcal{H}$ as it produces smoother gradients for framework optimization. Furthermore, we do not discard the relative transformation information to calculate the relative pose loss as suggested in [28]. Moreover, being influenced from [29] we modified the relative pose loss as follows:

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

3.4 Robot Grasping Pipeline

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

$$\mathbb{E}[u^*, v^*]_d = \underset{u,v}{\operatorname{argmin}} \; exp - \left(\frac{\|I_D[u, v] - d\|}{exp(t)} \right)^2 \tag{8}$$

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

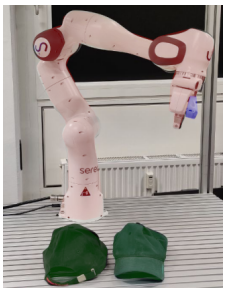

Figure 3: Illustration of the robot grasping pipeline setup. In the image, the robot is highlighted in red, the caps in green, and the camera in blue.

4 Experiments & Results

4.1 Dense Object Nets

We implemented training and benchmarking using "PyTorch-Lightning"[36] and "PyTorch"[37] libraries. Furthermore, we employ ADAM[38] optimizer to optimize the model for 2500 epochs 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 benchmark the DON with Pixelwise NT-Xent loss as in [17] with a fixed batch size of 1 and 128 image-pair correspondences. As per the benchmarking results in Table 2, the descriptor's robustness increases as the descriptor's dimension gets longer.

Table 2: Benchmark of DON framework for GPU consumption and AUC 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

The AUC for $PCK@k, \forall k \in [1, 100]$ is computed with 256 image-pair correspondences and the metrics mean and std. deviation is calculated from benchmarking 3 DON models trained for a single descriptor dimension. We could not train the descriptor dimension of 64 and 128 due to the limited VRAM. Furthermore, to inspect the results of trained DON, an interface is built using the PyGame library [39] to visualize the results of the trained DON. The mouse pointer in the image space is mapped to the pixel, and the descriptor at that pixel is queried in another image-descriptor space. We further use the spatial probability of the descriptor to visualize the queried descriptor in the image space using Equation 8 Identify if there are any multi-modal spatial activations in the descriptor spaces and none, as shown in Figure 4.

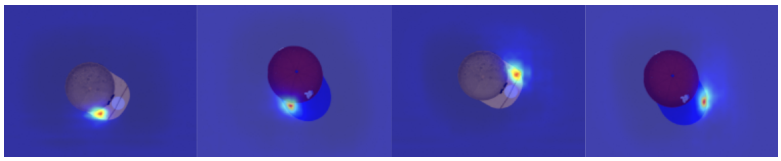

Figure 4: Depiction of the spatial probability heatmaps of the descriptor in the image space. We set the temperature in the Equation 8 to 1.1 and render the spatial probability heatmaps in the interface. The first and second image from the left and the right highlights the semantically equivalent descriptors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

4.2 Our Framework

To train our framework, we employ an ADAM optimizer to optimize the model for 2500 epochs with 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 fixed batch size of 1 and the StepLR scheduler with a step size 2500 and a gamma of 0.9 to train the model with all the loss weights to 1.0 except variance loss weight to $1e-3$. At first, we trained our model with 16 keypoints with a margin of 10 pixels as a hyperparameter for the separation loss, and later, we trained the models with 128 keypoints with a margin of 2 pixels. Later, we decreased the backbone dimension to 3, 16, 32 and 64 to benchmark the framework with 16 and 128 keypoints.

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

Our framework with 16 keypoints				
Descriptor Size (D)	64	128	256	512
AUC 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 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

4.3 Robot Grasping Pipeline

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

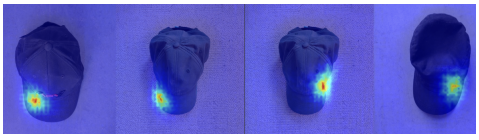

Figure 5: Visual inspection of the dense visual descriptors space of the real caps.

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

As our framework inertly regresses keypoints on the object, we could use it as an alternative approach to grasp the caps by computing the pose generated by the keypoints considering the actual depth information instead of network-regressed depth information. We extract the spatial probability of

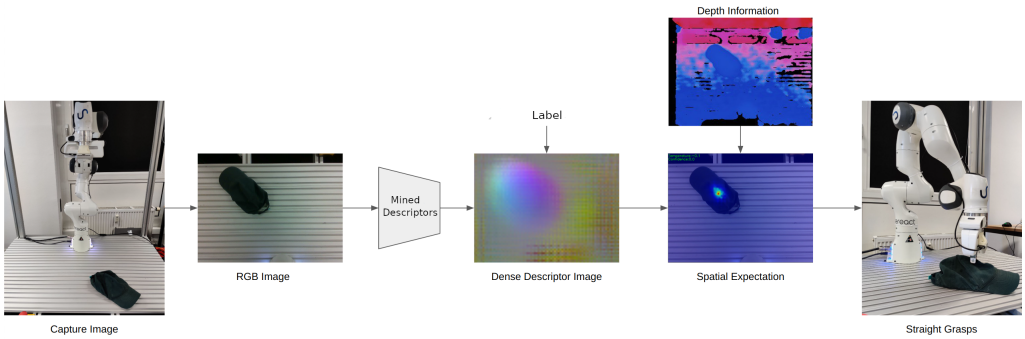

Figure 6: Depiction of the straight robot grasping pipeline.

each keypoint from the framework and deactivate spatial probabilities where the depth information is missing, as the depth image from the camera is noisy. Furthermore, the spatial expectations of the keypoints are projected to the camera frame to calculate a 6D pose in the camera frame. The 6D pose is transformed in the robot frame to perform an aligned grasp, as shown in Figure 7.

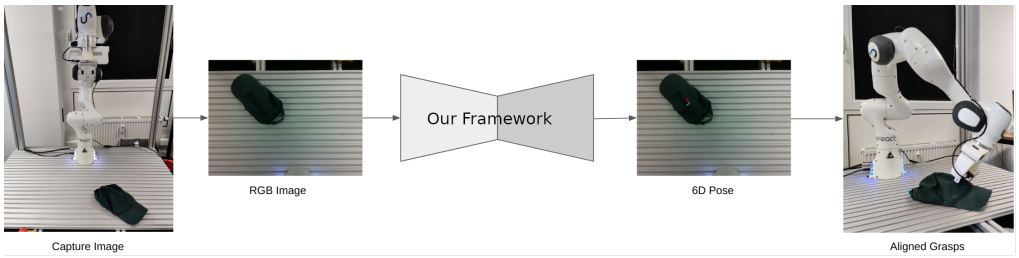

Figure 7: Illustration of the aligned robot grasping pipeline.

We did not evaluate the robot grasping pipeline.

5 Conclusion

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

6 Future Work

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

References

[1] N. Blomkamp, H. Zimmer, and S. Kinberg. *Chappie*. Sony Pictures Home Entertainment, 2015.

[2] G. Lucas and S. Ulstein. *Star wars*. 20th Century-Fox, 1977.

[3] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”. In: *Science* 362.6419 (2018), pp. 1140–1144.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), pp. 484–489.

[5] B. Entertainment. *World of warcraft*. Insight Editions, Division Of Palac, 2013.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[7] J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[9] R. A. Güler, N. Neverova, and I. Kokkinos. "Densepose: Dense human pose estimation in the wild". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7297–7306.

[10] *sereact*. "AI Robots for Production. Today". `https://sereact.ai/en`.

[11] P. R. Florence, L. Manuelli, and R. Tedrake. "Dense object nets: Learning dense visual object descriptors by and for robotic manipulation". In: *arXiv preprint arXiv:1806.08756* (2018).

[12] P. Sundaresan, J. Grannen, B. Thananjeyan, A. Balakrishna, M. Laskey, K. Stone, J. E. Gonzalez, and K. Goldberg. "Learning Rope Manipulation Policies Using Dense Object Descriptors Trained on Synthetic Depth Data". In: *CoRR* abs/2003.01835 (2020). arXiv: `2003.01835`.

[13] C.-Y. Chai, K.-F. Hsu, and S.-L. Tsao. "Multi-step Pick-and-Place Tasks Using Object-centric Dense Correspondences". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 4004–4011. DOI: `10.1109/IROS40897.2019.8968294`.

[14] P. Florence, L. Manuelli, and R. Tedrake. "Self-supervised correspondence in visuomotor policy learning". In: *IEEE Robotics and Automation Letters* 5.2 (2019), pp. 492–499.

[15] A. Ganapathi et al. "Learning Dense Visual Correspondences in Simulation to Smooth and Fold Real Fabrics". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 11515–11522. DOI: `10.1109/ICRA48506.2021.9561980`.

[16] A. Kupcsik, M. Spies, A. Klein, M. Todescato, N. Waniek, P. Schillinger, and M. Bürger. "Supervised Training of Dense Object Nets using Optimal Descriptors for Industrial Robotic Applications". In: *arXiv preprint arXiv:2102.08096* (2021).

[17] D. B. Adrian, A. G. Kupcsik, M. Spies, and H. Neumann. "Efficient and Robust Training of Dense Object Nets for Multi-Object Robot Manipulation". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 1562–1568.

[18] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[19] M. Belkin and P. Niyogi. "Laplacian eigenmaps for dimensionality reduction and data representation". In: *Neural computation* 15.6 (2003), pp. 1373–1396.

[20] P. R. Florence. "Dense visual learning for robot manipulation". PhD thesis. Massachusetts Institute of Technology, 2020.

[21] D. Hadjivelichkov and D. Kanoulas. "Fully Self-Supervised Class Awareness in Dense Object Descriptors". In: *5th Annual Conference on Robot Learning*. 2021.

[22] L. Yen-Chen, P. Florence, J. T. Barron, T.-Y. Lin, A. Rodriguez, and P. Isola. *NeRF-Supervision: Learning Dense Object Descriptors from Neural Radiance Fields*. 2022. DOI: `10.48550/ARXIV.2203.01913`.

[23] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. "Nerf: Representing scenes as neural radiance fields for view synthesis". In: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[24] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[25] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. "Barlow twins: Self-supervised learning via redundancy reduction". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12310–12320.

[26] C.-Y. Chai, K.-F. Hsu, and S.-L. Tsao. "Multi-step pick-and-place tasks using object-centric dense correspondences". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 4004–4011.

[27] M. E. Fathy, Q.-H. Tran, M. Z. Zia, P. Vernaza, and M. Chandraker. "Hierarchical metric learning and matching for 2d and 3d geometric correspondences". In: *Proceedings of the european conference on computer vision (ECCV)*. 2018, pp. 803–819.

[28] S. Suwajanakorn, N. Snavely, J. J. Tompson, and M. Norouzi. "Discovery of latent 3d keypoints via end-to-end geometric reasoning". In: *Advances in neural information processing systems* 31 (2018).

[29] W. Zhao, S. Zhang, Z. Guan, W. Zhao, J. Peng, and J. Fan. "Learning deep network for detecting 3d object keypoints and 6d poses". In: *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*. 2020, pp. 14134–14142.

[30] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. "Shapenet: An information-rich 3d model repository". In: *arXiv preprint arXiv:1512.03012* (2015).

[31] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam. "Blenderproc". In: *arXiv preprint arXiv:1911.01911* (2019).

[32] S. Marcel and Y. Rodriguez. "Torchvision the machine-vision package of torch". In: *Proceedings of the 18th ACM international conference on Multimedia*. 2010, pp. 1485–1488.

[33] W. Kabsch. "A solution for the best rotation to relate two sets of vectors". In: *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography* 32.5 (1976), pp. 922–923.

[34] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: (2016), pp. 770–778.

[35] W. P. Thurston. "Three-Dimensional Geometry and Topology, Volume 1". In: *Three-Dimensional Geometry and Topology, Volume 1*. Princeton university press, 2014.

[36] W. A. Falcon. "Pytorch lightning". In: *GitHub* 3 (2019).

[37] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[38] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[39] S. Kelly. "Basic introduction to pygame". In: *Python, PyGame and Raspberry Pi Game Development*. Springer, 2016, pp. 59–65.