DINOBot: Robot Manipulation via Retrieval and Alignment with Vision Foundation Models

Norman Di Palo and Edward Johns *

Abstract

We propose DINOBot, a novel imitation learning framework for robot manipulation, which leverages the image-level and pixel-level capabilities of features extracted from Vision Transformers trained with DINO. When interacting with a novel object, DINOBot first uses these features to retrieve the most visually similar object experienced during human demonstrations, and then uses this object to align its end-effector with the novel object to enable effective interaction. Through a series of real-world experiments on everyday tasks, we show that exploiting both the image-level and pixel-level properties of vision foundation models enables unprecedented learning efficiency and generalisation. Videos and code are available at https://sites.google.com/view/dinobot.

1 Introduction

The recent major successes in Deep Learning all had two main common ingredients: enormous, web-scale datasets, and substantial computational power to train increasingly large neural networks. However no such dataset of comparable size is available for robotics, where sensory perception must be coupled with actions.

To take advantage of these large, pre-trained networks, the robotics community has often used them as backbone representations, to then train a neural network to predict actions on top of the extracted representations. However, imitation learning (IL) using these foundation models often still requires a considerable number of demonstrations for generalisation to emerge [22, 30, 27, 28].

We argue that, rather than integrating foundation models into existing imitation learning methods as a backbone representation, we can design new imitation learning frameworks around these new capabilities of foundation models. To this end, we introduce **DINOBot**, a new imitation learning framework for robot manipulation tasks, which leverages the key capabilities of Vision Transformers (ViTs) trained through DINO [5] (which we call DINO-ViTs), a self-supervised method for training vision networks.

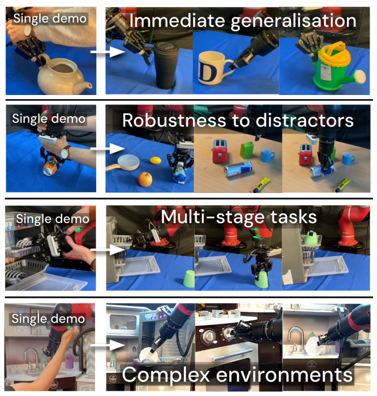


Figure 1: From a **single demo**, DINOBot can learn to adapt to new objects, be robust to distractors, execute multi-stage tasks, and interact with complex environments.

As described by DINO's authors, DINO-ViTs extract "*universal features suitable for image-level visual tasks as well as pixel-level visual tasks*" [24]. Inspired by these two capabilities, we designed DINOBot around two distinct modes of reasoning. Firstly, image-level *semantic* reasoning to generalise learned behaviours to novel objects. Secondly, pixel-level *geometric* reasoning to generalise

*The Robot Learning Lab at Imperial College London. Email to: `n.di-palo20@imperial.ac.uk`

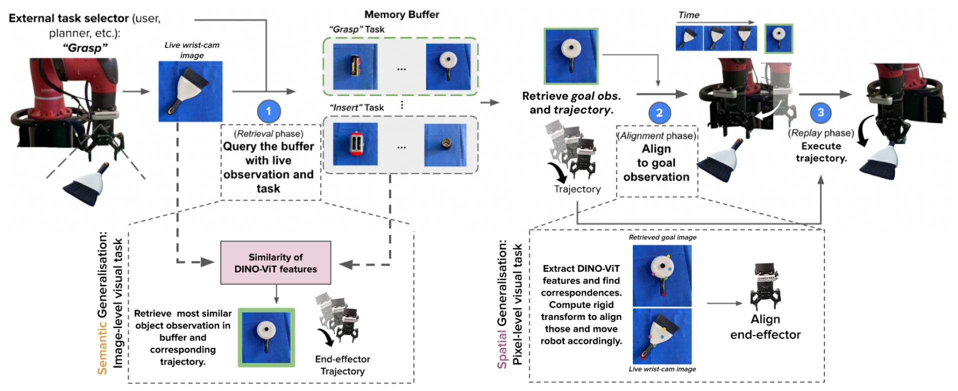

External task selector (user, planner, etc.):
"Grasp"
Live wrist-cam image
Memory Buffer
"Grasp" Task
"Insert" Task
1
(Retrieval phase)
Query the buffer with live observation and task
Semantic Generalisation:
Image-level visual task
Similarity of DINO-ViT features
Retrieve most similar object observation in buffer and corresponding trajectory.
End-effector Trajectory
Retrieve goal obs. and trajectory.
Trajectory
2
(Alignment phase)
Align to goal observation
Time
3
(Replay phase)
Execute trajectory.
Spatial Generalisation:
Pixel-level visual task
Retrieved goal image
Extract DINO-ViT features and find correspondences. Compute rigid transform to align those and move robot accordingly.
Live wrist-cam image
Align end-effector

Figure 2: Overall illustration of our framework. Upon observing a new object, the robot visually compares it other objects observed during demonstrations to find the most similar object (*semantic, image-level reasoning*), and retrieve both its image and the trajectory executed on that object. Then, the robot aligns its end-effector with this image (*spatial, pixel-level reasoning*), before then executing that trajectory. These two phases of reasoning are both based on extracting and matching DINO-ViT features.

learned behaviours to novel object poses. We integrate these two modes by modelling a manipulation task as a semantic image *retrieval* task, followed by a geometric *alignment* task (Fig. 2).

Through a series of real-world experiments, studying a range of tasks such as grasping, pouring, and inserting objects, we show not only that DINOBot achieves one-shot imitation learning on tasks where existing methods require many demonstrations, but it also achieves very efficient generalisation to novel objects. Our conclusion, and the key takeaway message, is clear: **designing an imitation learning framework around image-level (retrieval) and pixel-level (alignment) visual tasks allows us to leverage the remarkable capabilities of DINO-ViTs, leading to unprecedented learning efficiency compared to alternative paradigms**.

2 Method

Figure 2 illustrates our framework. During deployment, DINOBot executes manipulation tasks via a semantic *retrieval* and an spatial *alignment* phase, followed by a demonstration *replay* phase. During training, the operator provides demonstrations, and the collected data fills a memory buffer. Our method only needs an RGB-D wrist camera, rigidly mounted to the robot's end-effector. No prior knowledge of objects or tasks is needed, and no external camera is needed.

Spatial Generalisation through Alignment and Replay. To teach the robot how to interact with an object, the human operator manoeuvres the end-effector to provide a demonstration, e.g. with kinesthetic teaching. The end-effector starts from the *bottleneck pose*, B^O, which is a pose arbitrarily chosen by the operator to start the demonstration from, and from where the object must be visible from the wrist camera. The demonstration is recorded as a sequence of 3D linear and 3D angular velocities of the end-effector, expressed in the end-effector frame E. This sequence is the trajectory s, such that $s = [\mathcal{V}_1^E, \dots, \mathcal{V}_T^E]$, where $\mathcal{V} = [v_x, v_y, v_z, w_x, w_y, w_z]$. When starting the demonstration from pose B^O, the robot also records the wrist-camera observation of the object from that pose, which we call the *bottleneck observation*. For each demonstration, a *task* name is also specified by the operator, e.g. "Grasp", "Insert", or "Pour". In summary, each time a new demonstration is recorded by the operator, the framework adds the following data to the memory buffer: the bottleneck observation recorded before starting the demonstration, the trajectory of velocities $s = [\mathcal{V}_1^E, \dots, \mathcal{V}_T^E]$, and the task name.

We can imagine the bottleneck pose being rigidly attached to the virtual (since we do not assume object models) object frame. With W being the world frame, when we move the object, the global bottleneck pose B^W moves rigidly with the object, whilst the local bottleneck pose B^O stays constant. If the end-effector is re-aligned to B^O when the object is moved, replaying the end-effector velocities of the recorded trajectory would suffice in solving the task [18]. Thus, the robot only needs to reach B^O again for novel poses of this object, which is now expressed as a different B^W.

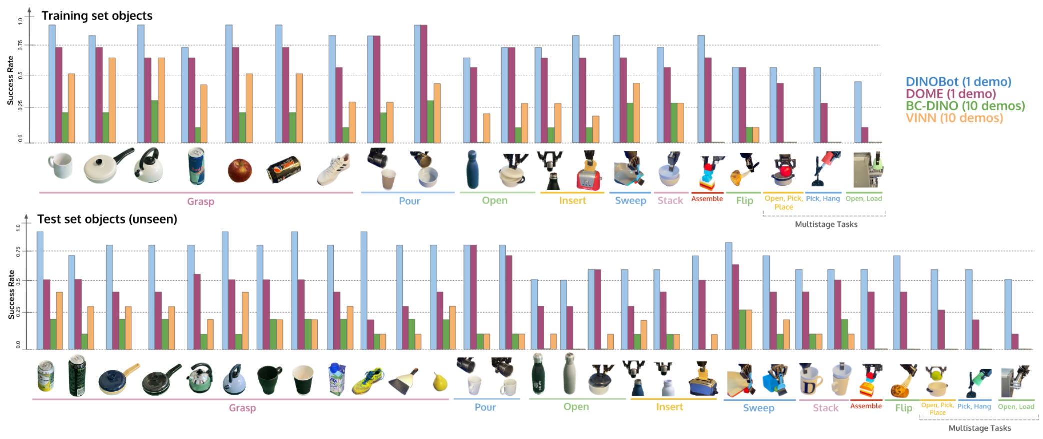


Figure 4: Success rates on each object for all methods.

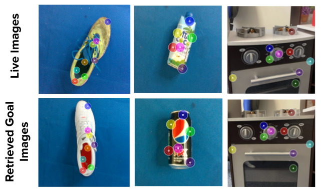


Figure 3: In each column, given a live image (top) DINOBot retrieves from the buffer the most similar image in the buffer (bottom), and finds correspondences between the two.

When a new object is observed during testing, how can the robot align itself with the object to reach B^W again? Given the observation recorded at the beginning of some demonstration stored in the buffer, and the current live observation recorded from the wrist-camera, DINOBot performs visual servoing to align the two images. Specifically, we extract deep patch features [8] from the DINO-ViT and use the method described in [2] to find correspondences through a Best Buddies Nearest Neighbour matching phase 3). This procedure generates two lists of keypoints, defined as 3D coordinates $C_1 = \{x_{1,i}, y_{1,i}, z_{1,i}\}$, $C_2 = \{x_{2,i}, y_{2,i}, z_{2,i}\}$ (we extract the depth of each RGB pixel correspondence through the RGB-D wrist camera). We then compute the least-squares rigid transformation that aligns the two lists of corresponding keypoints [34, 23] and move the robot accordingly. We repeat this process until the alignment is precise enough, i.e. the norm of the distance of the two lists of correspondences is smaller than a threshold. Once the alignment is completed, replaying the recorded trajectory allows the robot to successfully interact again with the object. We provide additional info and code on the website.

Semantic Generalisation through Retrieval. In the previous section, we described how we formulate the interaction with an object as visual alignment with a goal image, followed by replay of a trajectory. But given a new object to interact with, how does the robot select the best goal image and trajectory from its memory buffer? First, we assume that the robot receives, either by the human operator or by an external planner, the *task* to execute with the object, e.g. “Grasp" or “Open". The robot then records a live observation of the object from its wrist-camera. It then *retrieves* from its memory buffer the *bottleneck observation* most similar to the live observation, from the subset of the memory buffer for that task (e.g. if the task is to “Grasp", then all demonstrations which were also “Grasp" tasks would be considered). DINOBot performs retrieval by extracting features for each observation in the buffer and the live observation through the DINO-ViT. In our case, we extract the CLS token, a 1×768 vector [8]. It then performs a nearest neighbour search by computing the cosine similarity between the extracted features, as described in [24], to find the closest match in the buffer. The best match is retrieved together with its recorded trajectory: these are then used as goal observation for the *alignment* phase and actions to be replayed during the *replay* phase. As such, DINOBot is able to generalise observed demonstrations to novel objects, by independently using both the image-level and pixel-level capabilities of vision foundation models.

3 Experiments

In this section, we empirically measure the ability of our method to learn behaviours efficiently and transfer them effectively to new objects, and we structure our experiments around several important questions which we present in the following pages. These experiments were conducted on two different real-world environments, on a total of **15 tasks** with **53 objects** (a task, in this work, can be intuitively described as a verb, like *grasp* or *open*, and be applied to many different objects). First, a tabletop environment with 49 everyday objects (20 train, 29 test) and 8 types of tasks: **grasping**, **pouring**, **opening**, **inserting**, **sweeping**, **stacking**, **assembling**, **flipping** and 3 *multi-stage tasks* following the demonstration procedure in [6]: **open+pick+place**, **hanging cups** and **opening and loading a dishwasher**. Second, a toy kitchen environment where we teach 4 tasks: **opening** the microwave, **inserting** a plate into a dishwasher, **placing** a plate in the sink, and **turning** an oven knob.

A more detailed description of the tasks, the baselines, the setup, and a substantially larger set of experiments and ablations are included in the Supplementary Material. We invite you to watch the videos on our website at https://sites.google.com/view/dinobot.

Can DINOBot learn everyday-like tasks efficiently, and transfer those skills to novel objects? How does it compare to baselines from recent literature? To answer this, we train each method (baselines described in the Supplementary Material) with demonstrations of how to interact with the objects in the training set on the tabletop environment (Fig. 4, top). We then test each method using the objects depicted in the figure: both the training objects (top), and the unseen test objects (bottom). At test time, we position each object randomly on the table, and inform the robot what the task to perform is (e.g. “Grasp", “Open", etc). We sample a position inside a 40cm $\times$ 40cm area, and an angle between -45° and 45° relative to the original demonstration angle. We run 10 trials per object, sampling a new object pose each time. In this tabletop scenario, the alignment phase is 4-DOF, with the robot always aligned with the vertical axis and only rotating around this axis, while the *replay phase* trajectory execution is 6-DOF. In the Supplementary Material, we also explore 6-DOF alignment in the kitchen environment.

The performance of each method on each object is shown in Fig. 4. We group the results into training set objects (for which the robot received demonstrations, to study one-shot IL), and test set objects (to study generalisation to novel objects).

The results show that not only can DINOBot obtain remarkable one-shot performance on the training objects, but that it can generalise to unseen objects: performance stays considerably close to the training set performance.The performance of the baselines is noticeably lower, illustrating the benefits of our combination of retrieval and alignment to leverage the capabilities of DINO-ViTs. Using the extracted features as input to train a BC network, perhaps the most typical paradigm for using pre-trained networks in IL [22, 27, 30], clearly performs worse, even with 10 times the number of demonstrations. DOME performs well on training set objects, but performance degrades when trying to generalise to unseen objects, as the method was not designed to do so.

4 Conclusion

We have introduced DINOBot, an imitation learning framework designed around image-level (retrieval) and pixel-level (alignment) visual tasks, to take full advantage of the abilities of DINO-ViT foundation models. Our extensive experimental investigation (mostly in the Supplementary Material) demonstrated how this framework surpasses other DINO-based baselines, that do not harness the full capabilities of these vision models.

References

[1] K. Alton and M. van de Panne. Learning to steer on winding tracks using semi-parametric control policies. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 4588–4593, 2005.

[2] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *arXiv preprint arXiv:2112.05814*, 2(3):4, 2021.

[3] Max Argus, Lukas Hermann, Jon Long, and Thomas Brox. Flowcontrol: Optical flow based visual servoing. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7534–7541. IEEE, 2020.

[4] Rinu Boney and Alexander Ilin. Semi-supervised and active few-shot learning with prototypical networks. *arXiv preprint arXiv:1711.10856*, 2017.

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[6] Norman Di Palo and Edward Johns. Learning multi-stage tasks with one demonstration via self-replay. In *Conference on Robot Learning*, pages 1180–1189. PMLR, 2022.

[7] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. Tapir: Tracking any point with per-frame initialization and temporal refinement. *arXiv preprint arXiv:2306.08637*, 2023.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[9] Maximilian Du, Suraj Nair, Dorsa Sadigh, and Chelsea Finn. Behavior retrieval: Few-shot imitation learning by querying unlabeled datasets. *arXiv preprint arXiv:2304.08742*, 2023.

[10] Haoshu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. Graspnet-1billion: A large-scale benchmark for general object grasping. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11441–11450, 2020.

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

[12] Walter Goodwin, Ioannis Havoutis, and Ingmar Posner. You only look at one: Category-level object representations for pose estimation from a single example. In *6th Annual Conference on Robot Learning*, 2022.

[13] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

[14] Denis Hadjivelichkov, Sicelukwanda Zwane, Lourdes Agapito, Marc Peter Deisenroth, and Dimitrios Kanoulas. One-shot transfer of affordance regions? affcorrs! In *Conference on Robot Learning*, pages 550–560. PMLR, 2023.

[15] Siddhant Haldar, Jyothish Pari, Anant Rai, and Lerrel Pinto. Teach a robot to fish: Versatile imitation from one minute of demonstrations. *arXiv preprint arXiv:2303.01497*, 2023.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Sergio Izquierdo, Max Argus, and Thomas Brox. Conditional visual servoing for multi-step tasks. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2190–2196, 2022.

[18] Edward Johns. Coarse-to-fine imitation learning: Robot manipulation from a single demonstration. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4613–4619. IEEE, 2021.

[19] Jiang Lu, Pinghua Gong, Jieping Ye, and Changshui Zhang. Learning from very few samples: A survey. *arXiv preprint arXiv:2009.02653*, 2020.

[20] Elman Mansimov and Kyunghyun Cho. Simple nearest neighbor policy method for continuous control tasks, 2018.

[21] Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. In *Robotics Research: The 19th International Symposium ISRR*, pages 132–157. Springer, 2022.

[22] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[23] Shinji Oomori, Takeshi Nishida, and Shuichi Kurogi. Point cloud matching using singular value decomposition. *Artificial Life and Robotics*, 21:149–154, 06 2016.

[24] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[25] Jyothish Pari, Nur Muhammad Shafiullah, Sridhar Pandian Arunachalam, and Lerrel Pinto. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021.

[26] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[27] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. In *Conference on Robot Learning*, pages 416–426. PMLR, 2023.

[28] Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. In *Conference on Robot Learning*, pages 1332–1344. PMLR, 2023.

[29] Devavrat Shah and Qiaomin Xie. Q-learning with nearest neighbors. *Advances in Neural Information Processing Systems*, 31, 2018.

[30] Rutav Shah and Vikash Kumar. Rrl: Resnet as representation for reinforcement learning. *arXiv preprint arXiv:2107.03380*, 2021.

[31] Dana Sharon and Michiel van de Panne. Synthesis of controllers for stylized planar bipedal walking. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2387–2392, 2005.

[32] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022.

[33] Anthony Simeonov, Yilun Du, Yen-Chen Lin, Alberto Rodriguez Garcia, Leslie Pack Kaelbling, Tomás Lozano-Pérez, and Pulkit Agrawal. Se (3)-equivariant relational rearrangement with neural descriptor fields. In *Conference on Robot Learning*, pages 835–846. PMLR, 2023.

[34] Olga Sorkine-Hornung and Michael Rabinovich. Least-squares rigid motion using svd. *Computing*, 1(1):1–5, 2017.

[35] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020.

[36] Eugene Valassakis, Georgios Papagiannis, Norman Di Palo, and Edward Johns. Demonstrate once, imitate immediately (dome): Learning visual servoing for one-shot imitation learning. *arXiv preprint arXiv:2204.02863*, 2022.

[37] Mel Vecerik, Carl Doersch, Yi Yang, Todor Davchev, Yusuf Aytar, Guangyao Zhou, Raia Hadsell, Lourdes Agapito, and Jon Scholz. Robotap: Tracking arbitrary points for few-shot visual imitation. *arXiv preprint arXiv:2308.15975*, 2023.

[38] Mel Vecerik, Jean-Baptiste Regli, Oleg Sushkov, David Barker, Rugile Pevceviciute, Thomas Rothörl, Raia Hadsell, Lourdes Agapito, and Jonathan Scholz. S3k: Self-supervised semantic keypoints for robotic manipulation via multi-view consistency. In *Conference on Robot Learning*, pages 449–460. PMLR, 2021.

Setup: We run our experiments on a Sawyer robot mounting a Robotiq 2F-85 gripper. We use a single wrist-mounted Intel RealSense D435 camera. The camera receives RGB-D images which we rescale to $224 \times 224 \times 4$. The robot starts learning *tabula rasa*: no previous knowledge of tasks or objects, such as CAD models, is used.

Baselines: We compare DINOBot to the following baselines. 1) **DOME** [36], a similar one-shot IL method based on a learned (through simulation), goal-conditioned visual servoing phase followed by a trajectory replay. However, DOME does not generalise to novel objects, and so we extended the original method with our retrieval phase to provide the goal image to align with. This baseline exists to compare DINO's pixel-level alignment abilities with DOME's simulation-trained visual servoing network. 2) **BC-DINO**, a Behaviour Cloning (BC) implementation that trains a network to output actions on top of features extracted through a DINO-ViT. This baseline exists to compare our framework to the more typical approach in recent works, where BC policies are trained upon features extracted from pre-trained models [22, 30, 27] on all the collected demonstrations. 3) **VINN**: Visual Imitation through Nearest Neighbours (VINN) [25], frames Learning from Demonstration entirely as a retrieval problem, where actions are computed by retrieving the k most similar demo observations and averaging their corresponding actions. We use DINO features to embed the observations and retrieve them.

All methods receive as input the task to execute and the wrist-camera observation. For DINOBot and DOME, we provide a single demonstration for each training object. As BC-DINO and VINN are not designed as one-shot IL methods, we provide 10 demonstrations per training object. More details on the implementation of these methods and on the experiments can be found on our website.

Tasks Selection: To prove the effectiveness and generality of DINOBot, we took the tasks from other recent imitation learning papers, and we therefore test on *all the tasks* from: 1) **Relational-NDF** [33], (where instead of *bottle in container* we have multiple, more precise insertion tasks) + 2) **FISH** [15], (non multifingered ones - as *door opening*, we have *open a microwave door* - for *key insertion*, we have several precise insertion tasks with a lower than 5mm error tolerance), + 3) **VINN** [25], (where instead of *pushing* directly, we sweep, i.e. push with a tool), + 4) **Relay Policy Learning** [13].

Can DINOBot learn to interact with a complex kitchen environment? In this section, we now study more complex 6-DOF tasks and environments. We provide a single demonstration to our robot for the following tasks: 1) open a microwave, 2) insert a plate into the dishwasher, 3) put a plate into the sink, and 4) turn an oven knob. After each demonstration, we test the ability of our method to replicate the task with the robot starting from a different starting position, with 10 trials for each task. As the only input the robot receives are wrist-camera observations and no proprioceptive data, moving the robot to a different initial state is akin to moving the kitchen to test for spatial generalisation.

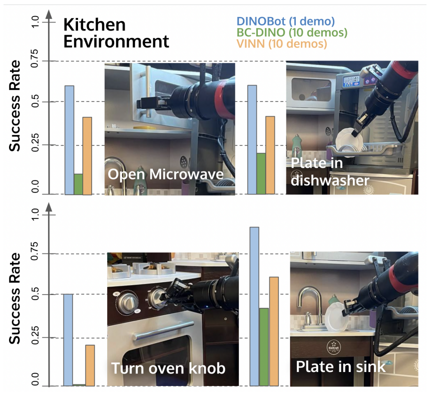


Figure 5: Results on kitchen tasks.

In this experiment, we skip the *retrieval* phase to study the *alignment* and *replay* phases' one-shot IL ability in isolation, such that the robot is asked to perform the same task it has just been shown from the demonstration. The robot now has to perform a challenging 6-DOF alignment by extracting correspondences between the *bottleneck observation*, recorded at the start of the demonstration, and its live wrist-camera observation (videos of execution and keypoints extractions on our website). We compare DINOBot with one demonstration to **BC-DINO** and **VINN** with 10 demonstrations. DOME is designed for 4-DOF tabletop settings and cannot be deployed for this task.

Results shown in Fig. 5 clearly show that our framework can successfully solve these everyday-like tasks, when receiving only a single demonstration. As with the previous experiments, the significantly better performance than BC-DINO and VINN again shows that explicitly using DINO's extracted visual features is superior to a more standard use of DINO as just backbone features.

Method	Without Distractors	With Distractors
DINOBot (1 demo)	**0.8**	**0.76**
DOME (1 demo)	0.67	0.5
BC-DINO (10 demos)	0.2	0.07
VINN (10 demos)	0.33	0.07

Table 1: Success rates with and without distractors.

Does DINOBot work even in the presence of distractor objects? In this experiment, we provide a single demonstration for each of the following tasks: 1) **grasping** a can, 2) **inserting** bread into a toaster, and 3) **pouring** from a cup into a mug. Here we study the *alignment* and *replay* phases robustness to distractors in isolation, hence we provide the goal image manually, skipping the *retrieval* phase. At test time, we use an unseen test set object (e.g. a different can or a different toaster), but we also vary the scene from the demonstration setup by adding a set of distractors from different classes (Fig. 6).

We compare DINOBot's performance against the baselines: DOME receives a single demonstration as well, while BC-DINO and VINN receive 10 demonstrations. We run 10 test trials for each task, with and without distractors. Results in Table 1 demonstrate that DINOBot not only surpasses all the baselines, but faces the smallest decrease in performance due to distractors. This demonstrates that DINO-ViT is very robust and able to extract correct correspondences between the goal observation object and the live observation, even in the presence of additional distractor objects(Fig. 6).

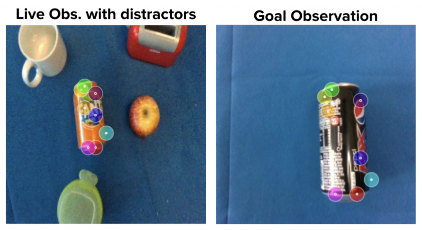


Figure 6: Despite the distractors, sensible correspondences can still be found.

What technique performs best in retrieving the most similar observation from the memory buffer? Being an essential part of our method (Sec. 2), we investigate the performance of different retrieval techniques to further motivate our design choice.

We compare retrieval based on features extracted from publicly-available ImageNet-trained ResNet-50 [16], CLIP [26], R3M [22] and DINO-ViT [5], each of which has been pre-trained on vast datasets of images (refer to the website for additional details). We measure retrieval accuracy by providing to each method an observation of an unseen object from the test set, composed of the 20 objects of Fig. 4 (bottom). We then measure how many times the method retrieves from the buffer an observation of the object belonging to the same class. We show results in Figure 7. The features extracted via a DINO-ViT achieve the best accuracy, confirming recent findings from the literature [5, 2] that such features effectively encode semantic, geometric, global and local information of the observation.

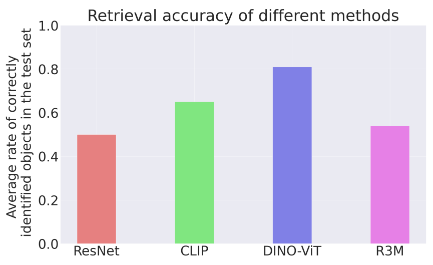


Figure 7: Comparison of retrieval accuracy of the baselines.

How accurate is the keypoints-based alignment phase of DINOBot? Here, we evaluate alignment precision in isolation, comparing it to two other alignment techniques from recent literature: **RoboTAP** [37], which uses keypoints extracted through **TAPIR** [7] for visual servoing and alignment, and **FlowControl** [3, 17], which uses optical flow to perform alignment; for the optical flow network, we use **RAFT** [35]. We use three pairs of objects: shoes, cans, and toasters. We generate a series of top-down observations of the objects where we manually translate and rotate the objects, in addition to artificially changing the background. Each method is then tested on its accuracy to compute the correct translation and rotation given an initial observation and a transformed observation. We test each method both when the pair of images depict the same object, and when the two images contain two different objects of the same class, e.g. two different shoes, to evaluate for semantic and geometric robustness.

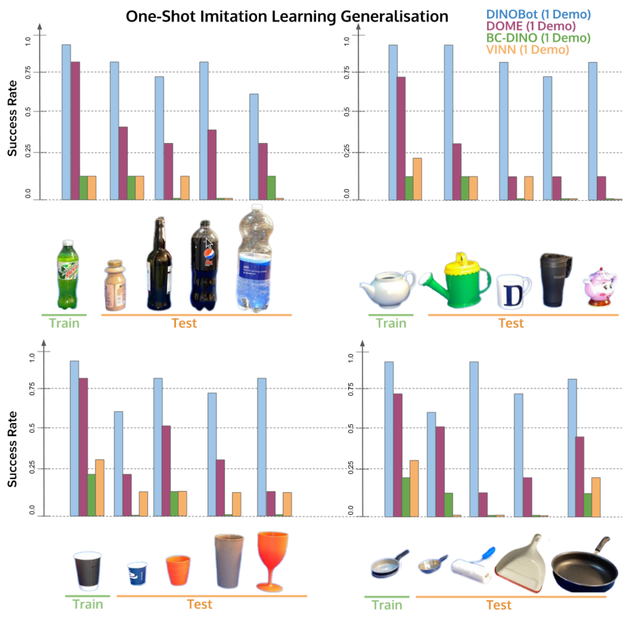


Figure 9: One-shot, intra-class IL generalisation experiments.

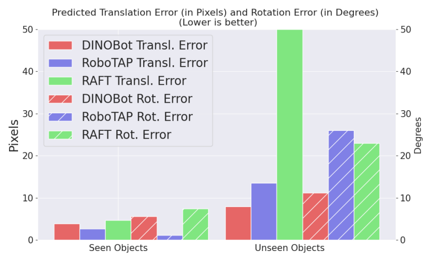


Figure 8: Alignment accuracy benchmark.

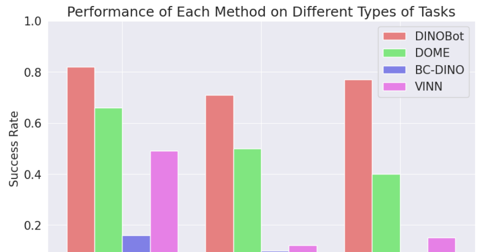
Results in Fig. 8 demonstrate that, while RoboTAP is extremely accurate when the same object appears in the pair of observations, performance degrades considerably when matching two different objects, albeit belonging to the same class. RAFT shows a similar trend, although with overall worse performance. DINOBot instead is able to generalise to unseen objects, maintaining strong performance.

How much can DINOBot generalise to objects with different sizes and appearance from a single demo? Here, we collected a new set of objects belonging to 4 classes - bottles, kettles, cups, and pans - where the object sizes and shapes vary significantly within each class. For each class, we then provide a demo for a single object, and then test on all the objects - 1 demo object, and 4 unseen objects - measuring the success rate over 10 test trials for each. As such, here we investigate the ability to generalise to novel objects that can vary substantially in size and shape, e.g. bottles more than double in size, mugs with handles of different shapes and position, etc. Results in Figure 9 show that DINOBot achieves significant generalisation and again outperforms baselines.

How does DINOBot compare to other methods across three challenges: adaptability, dexterity, and precision? To better highlight the different manipulation abilities expressed by our method, we cluster the results for the tasks in 4 into three groups, which cover some of the main abilities a robot should possess to tackle everyday-like tasks: **adaptability**, **dexterity**, and **precision**.

We cluster as follows. 1) All results from *grasp* experiments are grouped into **adaptability**, as grasping is the task that encounters the largest variety in objects [10], and thus requires significant adaptation to novel shapes based on visual observation. 2) All results from *pour, sweep, flip, open dishwasher* are grouped into **dexterity**, as these require non-trivial, often fast movements to succeed. 3) All results from *insert, assemble* are grouped into **precision**, as these have lower than 5mm positional tolerance.

Results, illustrated in Fig. 10, demonstrate that DINOBot surpasses all baselines on all these challenges. We observed that the use of retrieval helped DINOBot



adapt to unseen objects of various shape, while alignment and trajectory replay helped tackling precise and dexterous tasks, strongly surpassing the end-to-end baselines.

5 Related Work

Imitation Learning on top of Foundation Models. Many recent imitation learning methods for robot manipulation train end-to-end policies, built upon features extracted from pre-trained models. However, such methods still require tens or hundreds of demonstrations per task [30, 22, 27]. In our work, we show that an explicit decomposition into an image-level retrieval phase and a pixel-level alignment phase followed by a trajectory replay, results in a substantial improvement in data and time efficiency.

Retrieval for few-shot learning. Related to our use of DINO's image-level capabilities, retrieval for few-shot learning, especially in robotics, reinforcement learning and control, has also been investigated elsewhere [19, 4, 1, 31, 29, 20, 25, 9, 17]. Some approaches directly retrieve the actions to execute [20, 25, 31] or the data to train a policy [9]. [17] also retrieves the most similar demonstration from a buffer, and then aligns the robot by using optical flow between the current and goal observations, but is considerably worse at generalising to new objects as we demonstrate in Sec. 3.

Local correspondences for robot manipulation. Related to our use of DINO's pixel-level capabilities, extraction of dense or sparse correspondences through neural networks has been explored elsewhere for robot manipulation [11, 21, 38]. However, these methods require object-specific data collection to train networks. [37] uses a general keypoint-tracking network, that however does not generalise to unseen objects as we will illustrate in later sections. [12, 14, 2] demonstrate the capabilities of off-the-shelf DINO-ViT features for pose estimation. On top of these, we build a full IL framework based on retrieval and alignment.

Trajectory decomposition for robot manipulation. Our work is related to [18, 36], which also decompose robot manipulation into visual alignment and then replay of a trajectory. [18] trains an object-specific visual servoing policy with autonomously collected data, while [36] trains a general goal-conditioned alignment policy in simulation. However, neither of these methods enable generalisation to novel objects, and furthermore, with DINOBot we demonstrate that, by using DINO-ViTs, neither object-specific data collection nor additional simulation training is needed.

Semantic and spatial reasoning. CLIPort [32], like DINOBot, also phrases object manipulation as semantic reasoning combined with spatial reasoning. However, there are some fundamental differences: 1) their pipeline is more implicit, going from language and visual observations to affordance prediction through a single forward pass of a two-streams network, while we have explicit retrieval and alignment phases, that leads to DINOBot's better efficiency 2) their method is designed for top-down pick-and-place-like tasks, while we experiment with more complex tasks, also in a 6-DOF environment like a kitchen.