

Grounded Task Axes: Zero-Shot Semantic Skill Generalization via Task-Axis Controllers and Visual Foundation Models

M. Yunus Seker¹, Shobhit Aggarwal¹, and Oliver Kroemer¹

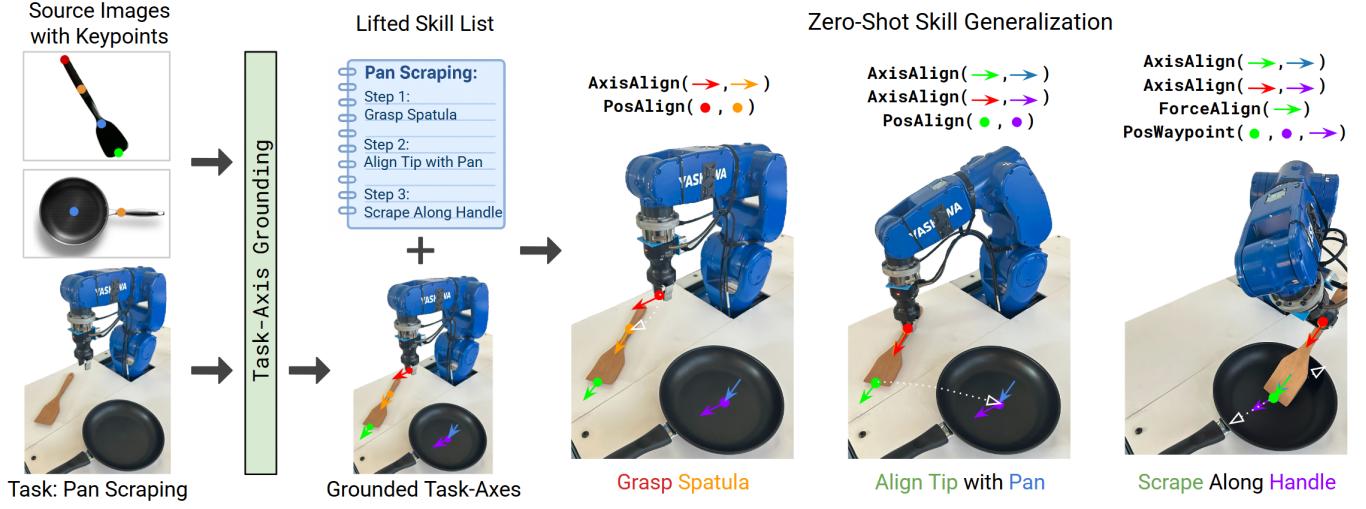


Fig. 1: Grounded Task-Axes (GTAs): Our system generalizes task-axis controllers zero-shot via vision-based keypoint correspondences.

Abstract—Transferring skills between different objects remains one of the core challenges of open-world robot manipulation. Generalization needs to take into account the high-level structural differences between distinct objects while still maintaining similar low-level interaction control. In this paper, we propose an example-based zero-shot approach to skill transfer. Rather than treating skills as atomic, we decompose skills into a prioritized list of grounded task-axis (GTA) controllers. Each GTAC defines an adaptable controller, such as a position or force controller, along an axis. Importantly, the GTACs are *grounded* in object key points and axes, e.g., the relative position of a screw head or the axis of its shaft. Zero-shot transfer is thus achieved by finding semantically-similar grounding features on novel target objects. We achieve this example-based grounding of the skills through the use of foundation models, such as SD-DINO, that can detect semantically similar keypoints of objects. We evaluate our framework on real-robot experiments, including screwing, pouring, and spatula scraping tasks, and demonstrate robust and versatile controller transfer for each.

I. INTRODUCTION

Transferring robot manipulation skills across diverse objects remains a fundamental challenge in open-world robotics. Humans intuitively generalize tasks across varied objects by identifying shared semantic and functional characteristics, even when confronted with substantial differences in appearance or geometry. Robotic systems, however, typically lack this intuitive adaptability, often failing to generalize when faced with even minor variations. Addressing this limitation requires robotic systems capable of abstracting tasks at a semantic level while still reliably performing low-level physical interactions.

A promising strategy to address this issue involves decomposing complex manipulation skills into modular components

defined relative to object-centric coordinate frames or task-specific axes. Early work introduced structured representations of robot manipulation tasks based on meaningful task frames and axes [1]–[3]. Building upon these concepts, Sharma et al. [4], [5] further advanced this approach by defining tasks as prioritized sequences of modular controllers. These modular frameworks significantly improved robustness, interpretability, and flexibility in robot manipulation. However, despite their effectiveness, existing task-axis methods still generally require extensive task-specific training, heavily annotated datasets or multiple demonstrations, and predefined sets of controllers, thus limiting their ability to generalize skills in zero-shot.

To overcome these limitations, we propose a novel zero-shot approach to semantic grounding of task-axis controllers. Our framework significantly expands the generalization capabilities of task-axis controllers by leveraging recent advancements in visual foundation models. Specifically, we propose a divide-and-conquer methodology that decomposes complex manipulation tasks into individual grounded task-axis (GTA) controllers. Each GTA controller specifies the low-level robot action—such as force application or positional adjustment—along clearly defined axes and keypoints grounded to the objects’ local geometry. Rather than relying on extensive retraining or demonstrations, we enable zero-shot generalization through example-based semantic grounding using powerful visual foundation models like SD-DINO [6].

Visual foundation models, such as SD-DINO, integrate semantic and geometric information, learned from large-scale pretraining, allowing for robust identification of semantically similar keypoints across highly varied object instances. In our framework, we exploit these capabilities to perform keypoint matching between reference and novel objects purely from

¹Carnegie Mellon University, The Robotics Institute, Pittsburgh, PA, USA
mseker@andrew.cmu.edu

RGB-D images. This matching facilitates accurate grounding of task-axis controllers in novel scenarios without task-specific demonstrations or object-specific training. Such a semantic and visual approach contrasts sharply with traditional geometric-based methods, offering enhanced robustness to significant variations in object shapes, textures, colors, and viewing angles. Our method makes four key contributions to skill generalization in robotic manipulation:

Modular Skill Representation: We represent complex skills as prioritized sets of grounded task-axis controllers, improving interpretability, providing task-tailored manipulation control, and enabling sub-skill controller reuse.

Semantic Grounding with Visual Foundation Models: We use SD-DINO to match keypoints across object instances, allowing accurate grounding in novel scenarios.

Zero-Shot Generalization: Our framework generalizes skills to novel objects without any training or demonstrations.

Real-World Evaluation: We validate our approach on diverse tasks, pan scraping, pouring, and screwing, demonstrating zero-shot execution across varied object geometries.

II. RELATED WORK

Task-Axis Controllers: Task-axis frames have long been used in robotic manipulation to define motions relative to meaningful reference frames. Early works [1]–[3] introduced constraint-based formulations for specifying manipulation along task-axes. Later methods further developed this idea, defining task-specific motions using fixed task-frames or axes for reliable execution [7]–[10].

More recently, task-axis controllers have been used as modular components for structuring complex behaviors. Sharma et al. [4] proposed composing such controllers via reinforcement learning, and later extended the framework to infer controller parameters from visual input via dense correspondence learning [5]. However, these methods depend on task-specific training. In contrast, our method performs inference in a zero-shot manner without retraining.

Imitation Learning and Trajectory Generation: Skill learning often involves training trajectory generators using imitation learning. Methods like DMPs [11]–[13] often require endpoints to be specified and a suitable coordinate frame to be selected for generalization. Rather than identifying a task-oriented coordinate frame, recent diffusion-based policies [14], [15], generate trajectories in global frames. Perceiver-Actor [15] directly predicts a target gripper frame and uses a motion planner for execution. By mapping scenes to compact keypoints and axes, our method can reduce input dimensionality and improve data efficiency. In the future, we plan to incorporate imitation learning approaches for trajectory generation into our controller library.

Visual Foundation Models for Correspondences: Vision foundation models have shown strong performance in establishing semantic and dense correspondences. Models like CLIP [16], DiFT [17], and DINO-ViT [18] learn semantic features from large-scale pretraining and enable pixel-level matching without supervision [19]. We leverage SD-DINO [6] to compute sparse semantic correspondences for

robust keypoint matching. Our system remains modular and can incorporate future vision models seamlessly.

Keypoint Correspondences for Manipulation: Recent works like DINOBOT [20], SKIL [21], and MAGIC [22] extract keypoints from pretrained vision models to enable generalization across object categories. However, these methods train task-specific policies from demonstrations or rely on real-to-sim transfer for planning [23]–[25]. By contrast, our method operates entirely in a zero-shot setting, without demonstrations or policy training.

III. METHOD

A. Task-Axis Controllers

Our proposed modular framework creates skills by combining multiple grounded task-axis controllers in parallel. This skill structure builds on the work of Sharma and Liang et al [4], [5], and we provide a more general formulation for these types of skills. It should be noted that our skills also build upon more classical task-frame formulations of robot manipulation [1]–[3], and our skill formulation is intended to be flexible enough to capture a wide range of skills.

In this section, we define the notation and structure of individual task-axis controllers. The following sections describe how they are combined to form skills (Sec III-B), how we ground them in the objects of new scenes (Sec III-C and III-D), and examples of specific controllers used in our experiments (Sec III-E).

A robot skill will consist of D grounded task-axis (GTA) controllers running in parallel. The i -th GTA controller has a set of $m_i \geq 0$ keypoints $\mathbf{g}_i = \{g_{i1}, \dots, g_{im}\}$ where $g_{ij} \in \mathbb{R}^3$, and a set of $n_i > 0$ axes $\alpha_i = \{\alpha_{i1}, \dots, \alpha_{in}\}$ where $\alpha_{ij} \in \mathbb{R}^3$. We refer to these keypoints and axes as the *grounding parameters* of the controller, as they form the basis for specifying the controller within the context of the current scene. The grounding process will anchor these features to the geometry of the objects in the scene, as well as the robot (e.g., a keypoint may be associated to the tip of a screw or a point on the gripper).

Each controller must specify at least one primary axis α_{i1} as the direction in which the controller is acting. For example, the primary axis α_{i1} will define the 3D world frame direction in which a force controller will apply a force.

The grounding parameters may vary over time, e.g., if the parameter reflects an object that is moving. We denote the time dependency with an additional subscript t , i.e., $\mathbf{g}_{it} = \{g_{i1t}, \dots, g_{imt}\}$. The variable α_{i10} thus specifies the controller's primary axis as defined at time step zero.

The action selection along the task axis α_{i0} is defined by a controller π_i and, sometimes, a trajectory generator τ_i . The trajectory generator is defined as $x_t^d = \tau_i(o, \mathbf{g}_i, \alpha_i, t; c_i^\tau, \theta_i^\tau)$, where o are sensor observations, c^τ are fixed parameters that specify the type of trajectory generator, and θ^τ specifies a set of tunable parameters for this type of generator. The output x^d specifies a desired value, which is passed to the controller.

We similarly define the controller policy as $u_t = \pi_i(o, \mathbf{g}_i, \alpha_i, x_t^d; c_i^\pi, \theta_i^\pi)$. To ease notation, we will often merge

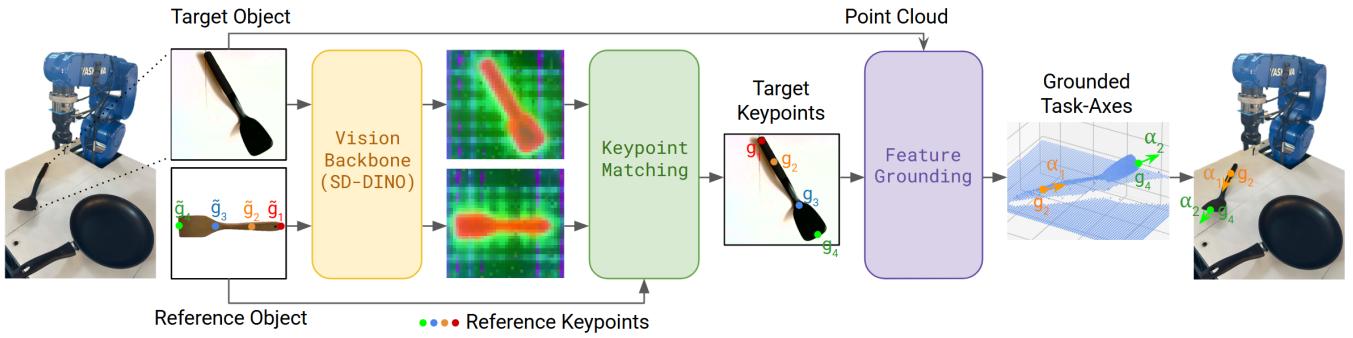


Fig. 2: Vision pipeline for grounding task-axes: Given a reference object annotated with keypoints and a target object, GTA uses SD-DINO as a vision backbone to extract visual features. We perform keypoint matching to find corresponding keypoints on the target object. Using these keypoints and 3D point cloud data, we ground the object-centered task-axis controllers in real-world.

the trajectory generator and controller jointly as a time-dependent controller $u_t = \pi_i(o, \mathbf{g}_i, \alpha_i, t; c_i, \theta_i)$. In practice, the controller may often be specified based on the initial set of grounding parameters $u_t = \pi_i(o, \mathbf{g}_{i0}, \alpha_{i0}, t; c_i, \theta_i)$ or the most recent values $u_t = \pi_i(o, \mathbf{g}_{it}, \alpha_{it}, t; c_i, \theta_i)$.

B. Task-Axis Controller Skills

A skill π is defined as a prioritized list of GTA controllers that are executed simultaneously. The index of the controller i indicates the ordering in the list, with lower indices indicating higher priority ($i=1$ is first priority, $i=2$ is second priority, etc.).

The robot needs to ensure that the controllers do not interfere with the execution of higher-priority controllers. Lower priority controllers are therefore projected into the null space of higher, resulting in a projected normalized axis $\hat{\alpha}$ of

$$\hat{\alpha}_{i1} = \text{Proj}(\alpha_{i1}, \{\alpha_{j1} | \forall j < i\}), \|\hat{\alpha}_{i1}\| = 1$$

with corresponding projected action commands \hat{u} . In practice, we use a separate list for translational and rotational controllers.

Three controller axes specify a full Cartesian coordinate frame. However, lower dimensional skills can also be defined using the framework. For example, to place a screw into a hole, the robot only needs to specify an axis α_{110} along the fixed central axis of the hole and a second α_{21t} radially from the hole to the screw tip. The skill formulation thus reflects the rotational invariant nature of the task. Similarly, for the orientation of the screw, we only need to specify a single control axis α_{21t} for aligning the hole and screw axes.

The resulting skills are defined in the Cartesian space, as is common for manipulation tasks. Ultimately, any robot skill needs to define a set of low-level joint commands. For completeness, we define this mapping as the *robot implementation* R of the GTA skill. The implementation function $R(\hat{\alpha}, \hat{u}, c)$ takes in the list of projected axes $\hat{\alpha}$, the corresponding list of projected actions \hat{u} , and the types of controllers c to compute the low-level joint commands. In our experiment, we only used a single type of robot. However, in the future GTA skills could be ported between robots by creating individual implementations for a shared library.

C. Grounding Task-Axis Controllers with Example Keypoints

The skill formulation in the previous sections allows us to define GTA controllers and skills in an abstract form with

unspecified variables for the keypoints and axes. Adopting nomenclature from symbolic planning, we refer to skills for which all of the other parameters (c, θ , controller order, etc.) have been specified but the grounding parameters g and α are unspecified as *lifted* task-axis skills. These skill definitions can thus specify manipulations, such as peg insertion, in the abstract. However, to execute the skill for a specific scenario and set of objects, we need to define the parameters through a grounding process. The resulting fully-defined skills then become *grounded* task-axis skills.

Determining the grounding parameters is a critical step in transferring a controller or skill to a new scenario. The grounding process *anchors* the parameters to objects in the scene. In this paper, we focus on example-based zero-shot grounding transfer based on RGB-D images \mathcal{I} . This approach allows us to leverage existing foundation models to provide broad transfer of skills between semantically similar objects.

We assume that the robot is provided with example grounding points \tilde{g}_{ij} for a previous source image \mathcal{I}^S . The goal is then to find the corresponding point g_{ij} for a new target image \mathcal{I} . To ease notation, we assume the camera is calibrated and that the depth value of each pixel in the image \mathcal{I} specifies a full 3D position in the world frame that the robot can index.

To transfer an example keypoint \tilde{g} and find a similar keypoint for grounding in the new scene, the robot uses feature maps ϕ computed by foundation model such as SD-DINO [6]. The feature map $\phi(\tilde{g}_{ij}, \mathcal{I}^S) \in \mathbb{R}^k$ maps the example point into a k dimensional embedding space. To find the corresponding grounding point in the current target scene \mathcal{I} , we define a mapping function M that finds the closest point in the embedding space

$$g_{ij} = M(\tilde{g}_{ij}, \mathcal{I}^S, \mathcal{I}) = \arg \max_g \phi(\tilde{g}_{ij}, \mathcal{I}^S)^T \phi(g, \mathcal{I})$$

Note that the range of candidate values for g to maximize over is limited to the 3D points corresponding to the pixels image \mathcal{I} . The arg max operator can be susceptible to noise depending on the quality of the image. We therefore also implemented a soft-max mapping

$$M(\tilde{g}_{ij}, \mathcal{I}^S, \mathcal{I}) = \mathcal{Z}^{-1} \sum_g \exp(\epsilon^{-1} \phi(\tilde{g}_{ij}, \mathcal{I}^S)^T \phi(g, \mathcal{I}))$$

where \mathcal{Z} is a normalization term for the softmax and ϵ is the temperature term.

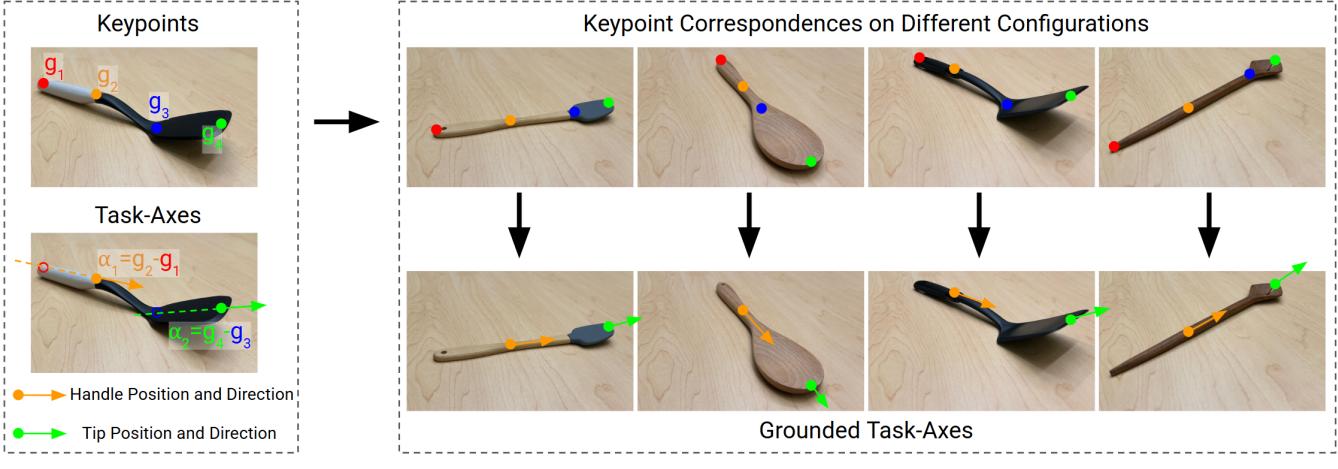


Fig. 3: (a) Example Keypoints and Task-Axes for a spatula object. (b) Zero-shot generalization to any object configuration.

To transfer an example axis $\tilde{\alpha}$ to a new scene, we consider a couple of different mappings depending on the type of axis. For global axes, such as the direction of gravity, we may use an identity mapping $\alpha_{ij} = \tilde{\alpha}_{ij}$. For axes defined by keypoints, we can directly use the transferred points to compute the axis $\alpha_{ij} = f_{ij}(\mathbf{g}_i)$, thus leveraging the keypoint transfer mechanism. Some axes are defined based on local object geometry, e.g., a surface normal or the direction of an edge. To capture these axes, we still require a reference keypoint \tilde{g}_{ik}^α to define the local region from which to extract the axis

$$\alpha_{ij} = f_{ij}(M(\tilde{g}_{ij}^\alpha, \mathcal{I}^S, \mathcal{I}), \mathcal{I})$$

where the mapping function M finds a suitable semantically similar point in the new image, and the function f extracts the axis based on the local geometry in the image I around this mapped point. Other forms of axis mappings could be specified in the future. For our experiments, and a wide range of manipulation tasks, the above types of axis mappings are sufficient for creating versatile skills.

D. Feature Maps

The feature map ϕ provides the basis of grounding lifted skills in new scenarios by finding similar points in the new scene to those in the reference example. The feature maps thus define the notion of similarity between points.

In manipulation tasks, we often need to consider the semantic similarity of keypoints for transfer. For example,

Cosine Similarity Maps using SD-DINO

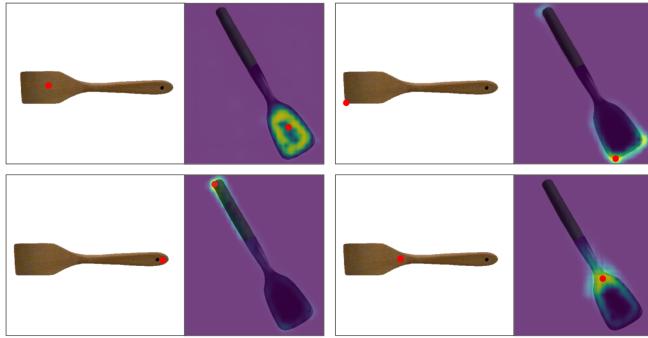


Fig. 4: Visualization of cosine similarity maps on target object image according to different pixel keypoints selected on the reference object.

tools often have tool tips, like the head of a screwdriver or the tip of a pen, that are critical to their successful usage. We therefore want a feature map that can capture such semantically similar points, despite small local geometric variations (e.g., a screwdriver with a different tip shape) or even larger global geometric variations (e.g., a hand screwdriver versus an L-shaped electrical one). We use the foundation model DINOv2 [26] as one of our feature maps, as it is well suited for capturing semantically similar points.

In some tasks, we also need to consider the spatial location of the points. For example, all edges of a plate may be semantically similar, but we want the robot to grasp the left side of a plate. We therefore want a feature map that can also capture more general spatial information. We achieve this using the Stable Diffusion (SD) [27] network.

To obtain the potential benefits of both approaches, we leverage a visual foundation model SD-DINO [6], which is a combination of both DINOv2 [26] and Stable-Diffusion [27]. We followed the procedure given in the SD-DINO paper to extract pixel-wise features for the given reference and target images. We first calculated the DINOv2 patch token features using DINOv2-b14, and we combined these features with the 3rd, 5th, and 11th decoder layer features of the Stable Diffusion at timestep 50. We upscaled all the features from both models to align their shapes with the input image dimensions.

To find the corresponding keypoints on the target image for each reference keypoint, we first took an average around each reference keypoint feature using a 3x3 window, and calculated a cosine similarity map using the features of each pixel in the target image. Finally, we selected the best corresponding keypoint by applying a soft-argmax to this cosine map with a temperature of 0.01. Figure 4 visualizes an example for the correspondence matching using the SD-DINO cosine similarity maps.

E. Example Controllers

In our experiments, we use a set of only 4 types of GTA controllers to create the various skills. This set can be expanded in the future. We describe our current library of GTA

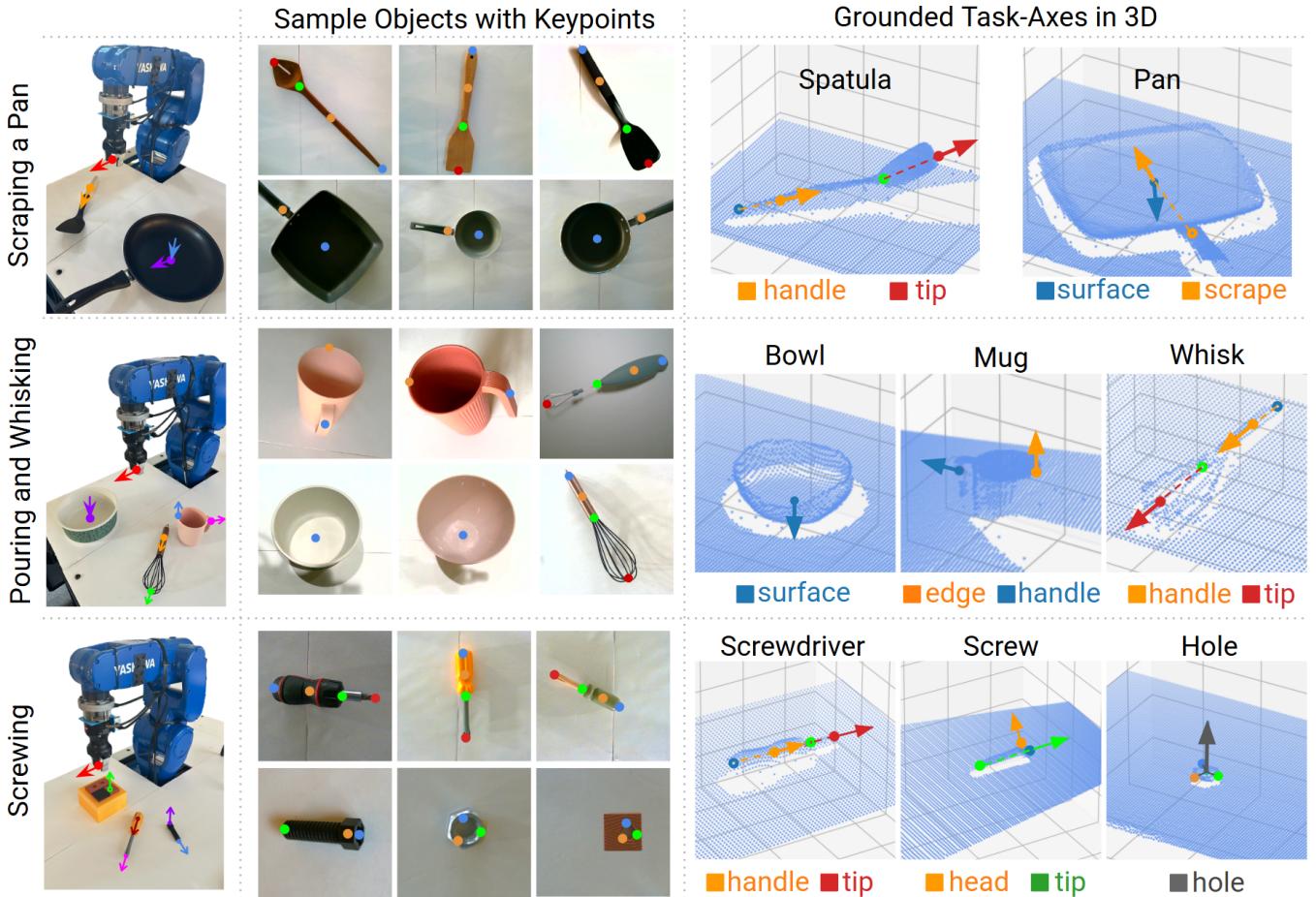


Fig. 5: (Left) Target tasks: Scraping a pan, pouring, and screwing. (Middle) Example object configurations with keypoint annotations from our in-lab data collection. (Right) Grounded Task-Axes derived from keypoints.

controllers below:

PosALIGN(g_{i1}, g_{i2}, θ_i) - defines a position controller that operates along $\alpha_{i1} = g_{i1} - g_{i2}$ to move keypoint g_{i1} on the robot's end-effector or tool to keypoint g_{i2} . The controller parameter $\theta_i \in \mathbb{R}^3$ defines a target offset between the two keypoints, such that $\theta_i=0$ moves the two points into alignment.

PosWAYPOINT($g_{i1}, g_{i2}, \alpha_{i2}, \theta_i$) - defines a position controller that moves keypoint g_{i1} on the robot's end-effector or tool to a set of waypoints along $g_{i2} + \alpha_{i1}$. θ_i specifies a list of offset waypoints.

AXISALIGN($\alpha_{i1}, \alpha_{i2}, \theta_i$) - defines a rotational controller that rotates axis α_{i1} on the robot's end-effector or tool to align with axis α_{i2} . Similar to PosAlign, $\theta_i \in \mathbb{R}^3$ defines a target offset angle between the axes.

FORCEALIGN(α_{i1}, θ_i) - defines a force controller that operates along α_{i1} to align with force value θ_i .

IV. EXPERIMENT SETUP

We evaluate our method on three diverse manipulation tasks—pan scraping, pouring and whisking, and screwing—each representing a distinct application domain: cleaning, cooking, and industrial assembly. Despite the variability across these domains, all tasks follow a unified formulation: we define a small set of functionally meaningful keypoints

for a set of reference objects, and derive task-axis controllers analytically using geometric relationships between keypoints and local surface geometry, allowing generalization without training.

In the pan scraping task, the robot grasps a spatula, aligns it with the given pan, and performs a simple scraping motion along the surface, following a direction projected from the pan handle to its center while maintaining contact force along the surface normal using force control. For the pouring and whisking task, the robot first grasps a mug, pours its contents into a bowl, and then grasps a whisk to stir the mixture inside. The screwing task involves grasping a screw, inserting it into a hole, and then grasping a screwdriver to drive it in while applying axial force along the screw's center line. Each of these tasks involves multi-step interactions with multiple objects, yet the keypoint-to-controller mappings are consistent and modular across tools. For instance, the screwdriver and spatula both need to be positioned and aligned before a forceful interaction, enabling controller reuse. Similarly, the pan and bowl share surface-normal-based controllers centered at their base. Figure 5 presents visual examples of the tasks, object variants, keypoints, and GTAs defined for each object.

To assess generalization, we collected an object data set covering a wide range of object appearances and poses. For each of the eight object classes, we selected three distinct

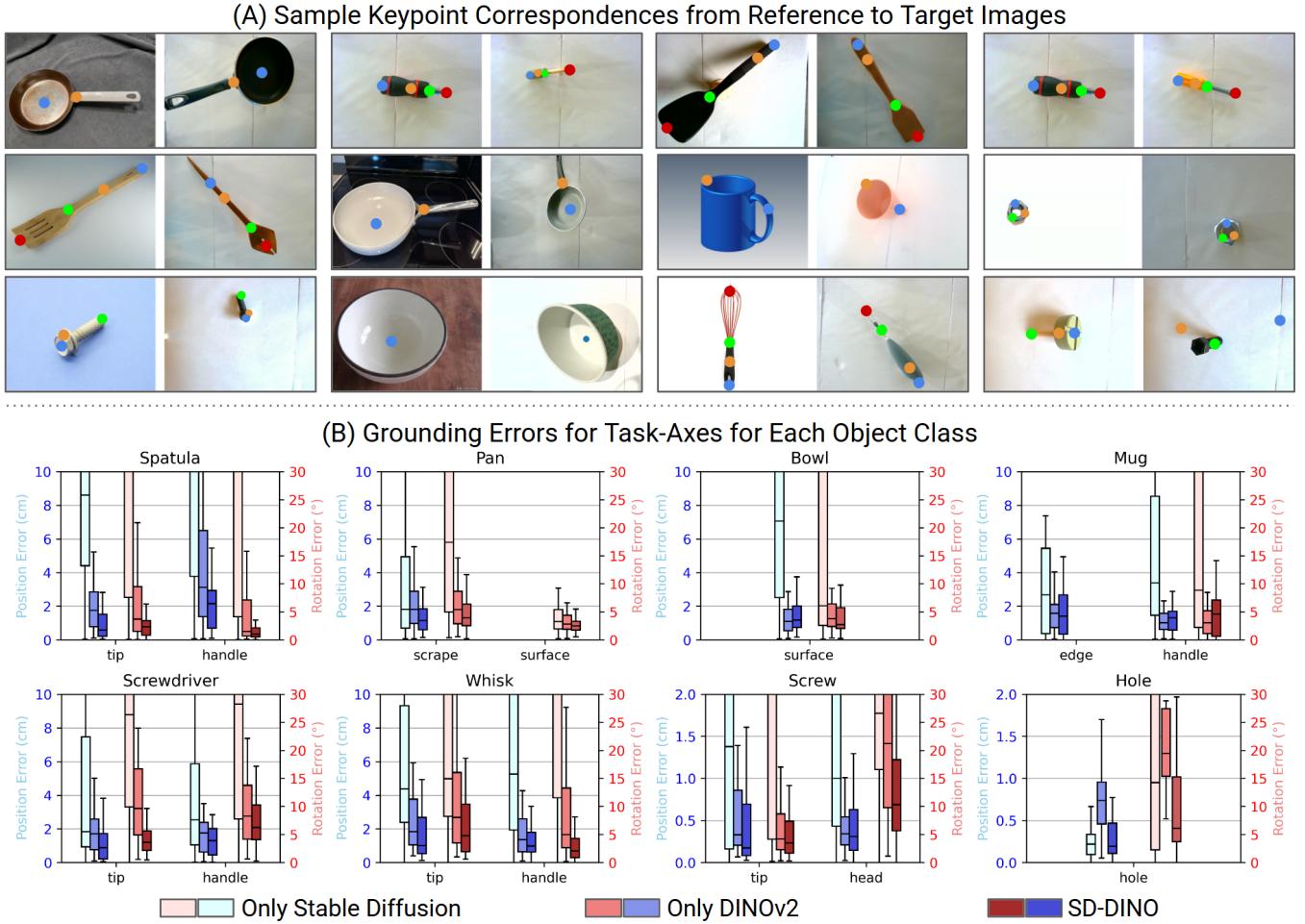


Fig. 6: (A) Examples for keypoint correspondences. Left side of the pairs show the reference keypoints, right side of the pairs show the corresponding keypoints using SD-DINO. Failure case in right-bottom. (B) Errors between ground-truth and grounded task-axes for each object class.

object variants and placed them in three unique configurations varying in position, orientation, and robot viewpoint. Additionally, we collected three internet images per object class. This yielded 54 reference-target pairs per object class, totaling 432 pairs. All local RGB-D data were captured using an Intel RealSense D405 stereo camera mounted on the wrist of a Yaskawa GP4 6-DOF robotic arm equipped with a Robotiq Hand-E two-finger gripper. Examples of object variants, annotated keypoints, and the resulting task-axis controllers overlaid on point clouds are illustrated in Fig. 5.

V. RESULTS

A. Evaluating Keypoint Correspondences

Before presenting quantitative results on grounding 3D task-axes, we first provide a qualitative evaluation of the predicted keypoint correspondences to illustrate the behavior and reliability of our keypoint matching. Figure 6A shows



Fig. 7: Grounding task-axes from a reference whisk to spatulas.

representative correspondence examples across a range of object categories from our dataset. In each image pair, the left image displays a reference example—either from our in-lab capture or collected from the internet—annotated with human-labeled keypoints. The right image shows the corresponding target object from our in-lab dataset, with keypoints predicted by our framework using SD-DINO features.

These examples highlight the model’s ability to accurately identify semantically aligned keypoints across significant variations in object configurations. While occasional mis-alignments arise in challenging settings, such as extreme viewpoint shifts, ambiguous geometry, or shadows (e.g., the screw example in the bottom right), the model generally demonstrates robust and consistent correspondence accuracy.

B. Evaluating Zero-Shot Task-Axis Groundings

We evaluate the accuracy of our task-axis grounding pipeline by comparing the grounded task-axis controllers against ground-truth values from manual keypoint annotations. Figure 6B summarizes the position and rotation errors across different object classes and GTA types. Overall, the results show high accuracy across the board: positional controller errors are typically below 1 cm, and rotational errors remain under 3 degrees for most objects and tasks, demonstrating the

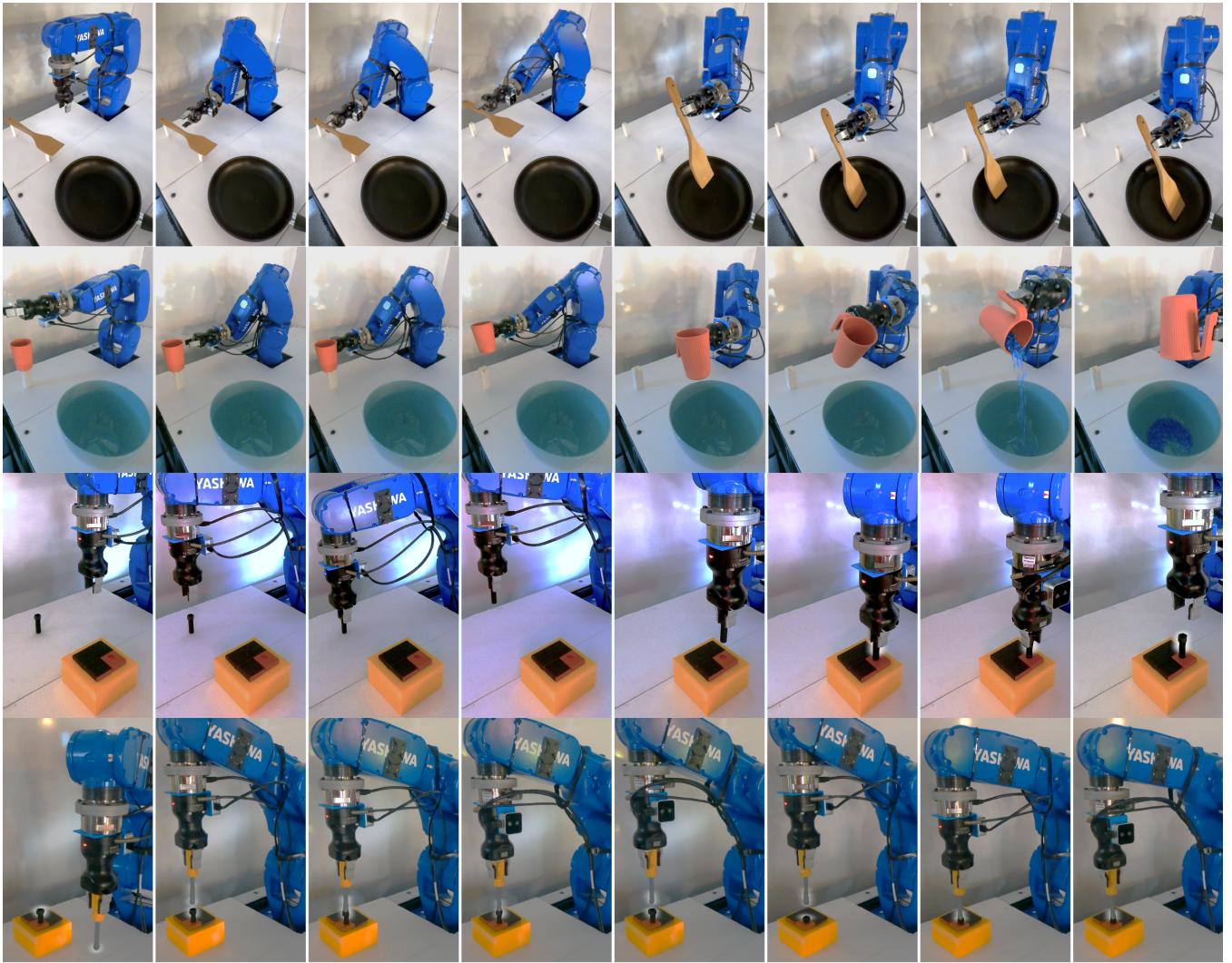


Fig. 8: Pan scraping, pouring, screw insertion, screwing, respectively. Screws and screwdriver tips are highlighted for better visualization.

effectiveness of our zero-shot controller grounding approach.

Notably, the apparent errors in some cases (e.g., the positional grasp controller on the spatula or screwdriver) reflect limitations of point-based evaluation rather than functional misalignment. The ground-truth grasp keypoint is a single manually labeled position, while in practice, a grasp may succeed anywhere along a functional region—such as the tool handle. Thus, small deviations in groundings do not necessarily translate to task failure. In fact, this tolerance to spatial variation suggests that our groundings are not only precise but also robust to annotation noise and ambiguity.

Figure 6B also provides insight into the impact of visual foundation model choice on controller accuracy. GTAs derived using only Stable Diffusion (light red and blue) show significantly higher variance, likely due to its limited semantic discrimination and reliance on low-level appearance features. DINOv2 (red and blue), which encodes functional and structural cues more effectively, performs better across all object classes. Our full model, SD-DINO (dark red and blue), consistently achieves the lowest error across all object classes, combining the semantic alignment of DINOv2 with the geometric refinement of Stable Diffusion. This confirms

that the hybrid architecture of SD-DINO offers the strongest foundation for keypoint matching and, consequently, for accurate controller grounding. These results demonstrate that our framework enables reliable zero-shot extraction of object-centric task-axis controllers across diverse objects, highlighting the value of semantic correspondence and geometry-aware grounding in manipulation tasks.

C. Transferring Semantically Similar Keypoints Across Distinct Objects

This experiment demonstrates that our framework can transfer semantically meaningful keypoint configurations across different object categories. Both whisks and spatulas are annotated with the same keypoint settings. While they serve different purposes—mixing versus scraping—they share structural similarities: both are elongated handheld tools operated primarily through the tip. Leveraging this shared functional geometry, we directly reused the keypoint annotations from a reference whisk to predict corresponding keypoints on a set of novel spatulas. Figure 7 (left) shows the annotated whisk, while the center image shows predicted keypoints on a sample spatula. From these correspondences, we grounded

task-axis controllers for each spatula in our dataset. Figure 7 (right) compares the task-axes errors when grounding spatula controllers from other spatulas versus the whisk. Although the whisk-based grounding yields slightly higher error, the performance remains within less than 1 cm difference for positions and 3 degrees of rotations—showcasing the robustness of our framework to cross-object generalization.

D. Executing Controllers to Generalize Zero-Shot Skills

We demonstrate the execution of multi-step manipulation skills using GTA controllers across three diverse tasks: pan scraping, pouring, and screw insertion and screwing. For each task, we defined a lifted skill composed of one or more GTA controllers based on the requirements of that specific manipulation and executed them in sequence after grounding on novel object instances.

Take the pan scraping task as an example. This skill involves several steps, including grasping the spatula, aligning it to the pan with a specific angle, and executing a scraping motion along the surface. Grasping using GTAs typically involves basic alignment of gripper with the object handles using PosALIGN and AxisALIGN controllers. Moving beyond grasping, the second step for aligning the tip of the spatula to the pan surface at a 45-degree angle from the surface normal and along the scraping direction, for example, we use the following three controllers in order of priority:

```
AxisAlign(spatulatip.dir, pansurface.dir, [0, 0, 45])
AxisAlign(grippery, panscrape.dir)
PosAlign(spatulatip.pos, panscrape.pos)
```

These controllers ensure that the spatula tip is correctly oriented with respect to the pan surface while aligning with the functional scraping direction. The priority structure allows the axis alignment of the surface angle to take precedence, while the scraping direction is projected into the null space, enabling compliant motion across the pan’s surface. In the following step, the robot maintains contact force during the scraping motion using a FORCEALIGN controller along the spatula tip direction and moves along the scraping direction with PosWAYPOINT.

The pouring and screwing tasks were implemented in a similar fashion, using appropriate combinations of the controllers in our controller library. For the pouring task, the robot first grasps the mug using PosALIGN and AxisALIGN controllers, moves it with PosALIGN to align with the bowl center, and then rotates the mug along its edge axis using an AxisALIGN controller to pour its contents. The screw insertion task involves grasping and aligning a screw above the hole using PosALIGN and AxisALIGN controllers, followed by pushing in using a FORCEALIGN controller along the insertion axis while rotating with AxisALIGN. The screwing task uses similar controllers for alignment and adds force and rotational controllers to drive the screw in along its axis.

Figure 8 shows the real-world execution of all three tasks with sample objects. These executions demonstrate that our modular, zero-shot framework enables accurate and generalizable manipulation of novel objects using semantically

grounded task-axis controllers. Notably, we were able to decompose all three multi-step tasks—despite their differences in tools, motions, and objectives—into low-level GTA controller skills instantiated from just four controller types. This compact yet expressive controller set highlights the versatility and reusability of our skill representation. In the future we plan to expand the set of controllers to support more precise manipulation control, e.g., controlled pouring.

VI. CONCLUSION

We presented a zero-shot framework for transferring manipulation skills by grounding modular task-axis controllers using semantically matched keypoints. By leveraging visual foundation models like SD-DINO, our method enables accurate grounding of controller parameters across novel objects without requiring retraining or demonstrations. We demonstrated reliable generalization of complex, multi-step tasks such as scraping, pouring, and screwing, across varied object categories and configurations. Our approach offers a modular, interpretable path toward scalable skill reuse and generalization in open-world robotic manipulation.

REFERENCES

- [1] M. T. Mason, “Compliance and force control for computer controlled manipulators,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 6, pp. 418–432, 1981.
- [2] D. H. Ballard, “Task frames in robot manipulation,” in *AAAI*, 1984.
- [3] M. H. Raibert and J. J. Craig, “Hybrid position/force control of manipulators,” 1981.
- [4] M. Sharma, J. Liang, J. Zhao, A. LaGrassa, and O. Kroemer, “Learning to compose hierarchical object-centric controllers for robotic manipulation,” *arXiv preprint arXiv:2011.04627*, 2020.
- [5] M. Sharma and O. Kroemer, “Generalizing object-centric task-axes controllers using keypoints,” in *IEEE ICRA*, 2021, pp. 7548–7554.
- [6] J. Zhang, C. Herrmann, J. Hur, L. Polania Cabrera, V. Jampani, D. Sun, and M.-H. Yang, “A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence,” *Advances in Neural Information Processing Systems*, vol. 36, 2023.
- [7] S. Manschitz, M. Gienger, J. Kober, and J. Peters, “Learning sequential force interaction skills,” *Robotics*, vol. 9, no. 2, p. 45, 2020.
- [8] T. Migimatsu and J. Bohg, “Object-centric task and motion planning in dynamic environments,” *IEEE RAL*, 2020.
- [9] J. E. King, M. Cognetti, and S. S. Srinivasa, “Rearrangement planning using object-centric and robot-centric action spaces,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016.
- [10] D. Berenson, S. Srinivasa, and J. Kuffner, “Task space regions: A framework for pose-constrained manipulation planning,” *IJRR*, 2011.
- [11] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, “Probabilistic movement primitives,” *NeurIPS*, 2013.
- [12] M. Y. Seker, M. Imre, J. H. Piater, and E. Ugur, “Conditional neural movement primitives,” in *Robotics: Science and Systems*, 2019.
- [13] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, “Dynamical movement primitives: learning attractor models for motor behaviors,” *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.
- [14] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, “Diffusion policy: Visuomotor policy learning via action diffusion,” *The International Journal of Robotics Research*, 2023.
- [15] M. Shridhar, L. Manuelli, and D. Fox, “Perceiver-actor: A multi-task transformer for robotic manipulation,” in *Conference on Robot Learning*. PMLR, 2023, pp. 785–799.
- [16] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, “Learning transferable visual models from natural language supervision,” 2021.
- [17] L. Tang, M. Jia, Q. Wang, C. P. Phoo, and B. Hariharan, “Emergent correspondence from image diffusion,” *Advances in Neural Information Processing Systems*, 2023.

- [18] S. Amir, Y. Gandelsman, S. Bagon, and T. Dekel, “Deep vit features as dense visual descriptors,” *arXiv preprint arXiv:2112.05814*, 2021.
- [19] J. Zhu, Y. Ju, J. Zhang, M. Wang, Z. Yuan, K. Hu, and H. Xu, “Densematcher: Learning 3d semantic correspondence for category-level manipulation from a single demo,” *International Conference on Learning Representations (ICLR) 2025*, 2024.
- [20] N. Di Palo and E. Johns, “Dinobot: Robot manipulation via retrieval and alignment with vision foundation models,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024.
- [21] S. Wang, J. You, Y. Hu, J. Li, and Y. Gao, “Skil: Semantic keypoint imitation learning for generalizable data-efficient manipulation,” *arXiv preprint arXiv:2501.14400*, 2025.
- [22] Y. Liu, J. Mao, J. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, “One-shot manipulation strategy learning by making contact analogies,” *arXiv preprint arXiv:2411.09627*, 2024.
- [23] X. Fang, B.-R. Huang, J. Mao, J. Shone, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, “Keypoint abstraction using large models for object-relative imitation learning,” *arXiv:2410.23254*, 2024.
- [24] Y. Ju, K. Hu, G. Zhang, G. Zhang, M. Jiang, and H. Xu, “Robo-abc: Affordance generalization beyond categories via semantic correspondence for robot manipulation,” in *ECCV*, 2024.
- [25] T. Zhang, Y. Hu, H. Cui, H. Zhao, and Y. Gao, “A universal semantic-geometric representation for robotic manipulation,” *arXiv preprint arXiv:2306.10474*, 2023.
- [26] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, *et al.*, “Dinov2: Learning robust visual features without supervision,” *arXiv preprint arXiv:2304.07193*, 2023.
- [27] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-resolution image synthesis with latent diffusion models,” 2022.