# NovaFlow: Zero-Shot Manipulation via Actionable Flow from Generated Videos

Hongyu Li[1,2*], Lingfeng Sun[1*], Yafei Hu[1], Duy Ta[1], Jennifer Barry[1], George Konidaris[2], and Jiahui Fu[1]

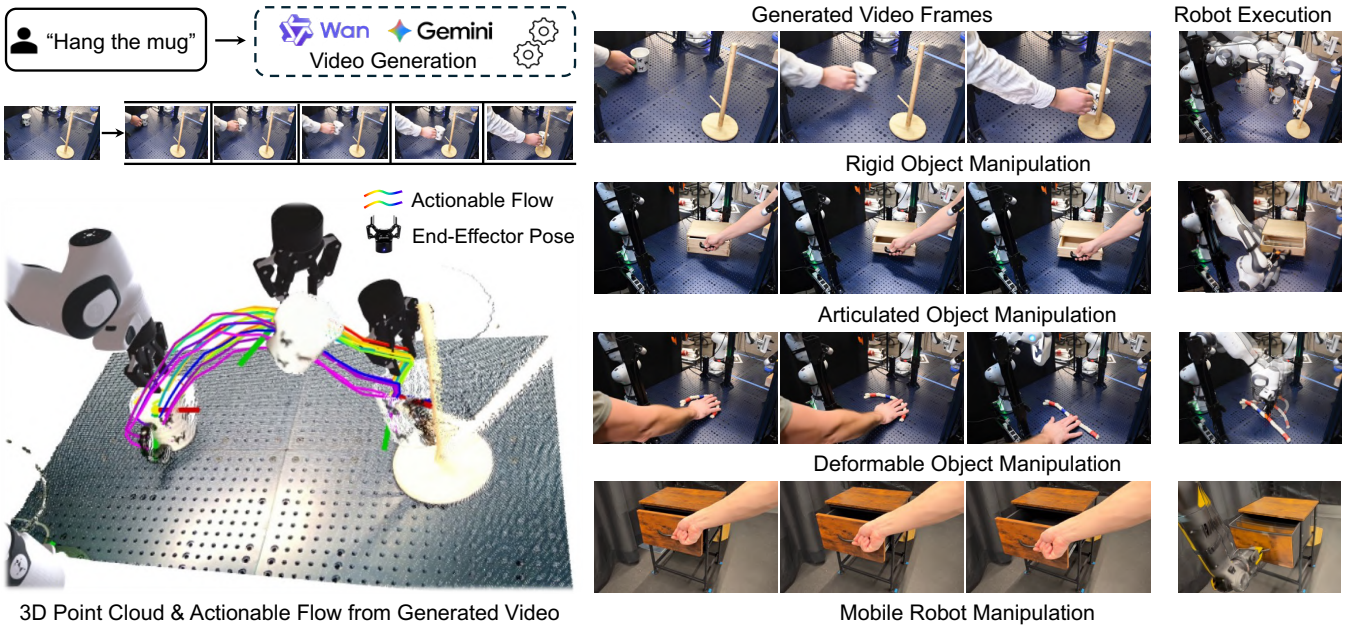


Fig. 1: **NovaFlow framework.** A generated task-solving video is distilled into a **3D actionable object flow** aligned with the robot's observation. From this flow, reference end-effector trajectories are computed and tracked, enabling robots to manipulate rigid, articulated, and deformable objects across different embodiments *without demonstrations*.

***Abstract*— Enabling robots to execute novel manipulation tasks zero-shot is a central goal in robotics. Most existing methods assume in-distribution tasks or rely on fine-tuning with embodiment-matched data, limiting transfer across platforms. We present NovaFlow, an autonomous manipulation framework that converts a task description into an actionable plan for a target robot without any demonstrations. Given a task description, NovaFlow synthesizes a video using a video generation model and distills it into 3D actionable object flow using off-the-shelf perception modules. From the object flow, it computes relative poses for rigid objects and realizes them as robot actions via grasp proposals and trajectory optimization. For deformable objects, this flow serves as a tracking objective for model-based planning with a particle-based dynamics model. By decoupling task understanding from low-level control, NovaFlow naturally transfers across embodiments. We validate on rigid, articulated, and deformable object manipulation tasks using a table-top Franka arm and a Spot quadrupedal mobile robot, and achieve effective zero-shot execution without demonstrations or embodiment-specific training. Project website: `https://novaflow.lhy.xyz/`.**

[1]Robotics and AI Institute (RAI) [2]Brown University.
*Hongyu Li and Lingfeng Sun contribute equally. Email: hli230@cs.brown.edu and lingfengsun@berkeley.edu. Hongyu Li and George Konidaris were supported by the Office of Naval Research (ONR) under REPRISM MURI N000142412603 and ONR grant N00014-22-1-2592. Partial funding was also provided by the RAI.

## I. Introduction

A long-standing goal in robotics is to build generalist robots capable of performing a wide variety of manipulation tasks in unstructured environments without task-specific training. Many believe that Vision-Language-Action (VLA) models [1–3] can achieve this generalization, following the success of Large Language Models (LLMs) [4, 5], Vision-Language Models (VLMs) [6, 7], and video generation models [8, 9] that learn from vast, internet-scale datasets. However, directly applying this paradigm to robotics creates a significant data bottleneck. VLA models, for their end-to-end training, require vast quantities of robot-specific vision-language-action data that is difficult and expensive to collect, a stark contrast to the readily available web-scale data used for LLMs and VLMs.

An alternative path towards generalist robots lies in creating modular systems that decompose the problem into task understanding and robot control. These systems leverage powerful pretrained models [10, 11] and traditional robotic engineering methods like inverse kinematics (IK) [12] or model predictive control [13] to bypass large-scale robot data collection, a promising strategy for closing the data gap [14]. For instance, some approaches use large language or vision-

language models to generate high-level plans, affordance maps, or semantic keypoints to guide the robot [15–18]. While these methods successfully offload semantic reasoning to large models, translating this understanding into physical actions remains an open problem. The control policy, for instance, relies on either predefined skill primitives (*e.g.*, opening a drawer) [11, 19] or learned skills from real-world demonstrations [10, 15, 16, 18, 20, 21]. This approach reintroduces the data bottleneck and limits generalizability and scalability.

To overcome these limitations, we propose **NovaFlow**, a novel framework that breaks the dependency on robot data to achieve autonomous manipulation. Our key insight is to *repurpose large-scale pretrained video generation models as a source of commonsense task understanding and implicit physical knowledge for deriving object motion*. We hypothesize that by training on internet-scale video data, these models have already captured a rich, generalizable understanding of task and object dynamics that can be leveraged for unseen objects, environments, and tasks. This separates our approach from prior work that relies on self-collected data to train smaller, specialized video models [22–25]. To translate this understanding from video to robot actions, we leverage *actionable 3D object flow*, a generalized atomic representation of object motion.

NovaFlow generates robot actions from a single visual observation and task description and consists of two components: a flow generator and a flow executor. The flow generator leverages large-scale video generation models to distill generalized knowledge of object motion into an *actionable 3D object flow*. This is achieved using a pipeline of pretrained perception modules for monocular depth estimation [26], 3D point tracking [27], and object grounding [28, 29]. The flow executor then translates this 3D flow into robot actions using IK and trajectory optimization, requiring no robot-specific data or task training. To handle diverse object types, the executor uses correspondence-based model-free tracking for rigid and articulated objects [30] and dynamic model-based planning with particle models for deformable objects [31, 32], using the flow as a tracking objective.

In summary, we present NovaFlow, an object-centric and embodiment-agnostic framework for autonomous manipulation that requires no task-specific tuning. We demonstrate its efficacy across both tabletop and mobile manipulator tasks involving rigid, articulated, and deformable objects. At its core, we introduce an actionable 3D object flow representation that is key to its generalizability and achieve state-of-the-art zero-shot performance on a range of real-world tasks, outperforming previous demonstration-free and data-dependent methods.

## II. Related Work

We define an approach as zero-shot or demonstration-free if it does not require collecting any robot-specific data or task-specific training. While LLMs and VLMs have shown promising zero-shot capabilities, their embodied successors, VLA models, have yet to achieve the same level of generalization. Recent VLAs [33, 34] still rely on data collection to generalize on novel embodiments or camera views. This is due to the data bottleneck created by the end-to-end training nature of VLAs. To address this, we decouple the task understanding (Sec. II-A) and robot control, bridged by an intermediate representation, the 3D object flow (Sec. II-B).

### A. Video-based Manipulation

Prior work has utilized video generation models for manipulation. Video can be a generalized representation of motion, which serves as a visual instruction for robots to execute tasks. Some work trains an inverse dynamics model [22, 35] or a policy [24, 36, 37] to convert the generated video into robot actions. Other work tracks the 6D pose of the end-effector [23] or the object [12]. While promising, these approaches require extensive robot-specific data to train a domain-specific model tailored to a particular embodiment, environment, or task [22–24, 35–37].

A key limitation of many video-based manipulation methods is their reliance on embodiment-dependent action generation, which hinders cross-embodiment generalization. To address this, object-centric approaches have been proposed. For example, a concurrent work [12] extracts 6D poses from the generated video for demonstration-free manipulation, which is object-centric and generalizes across embodiments. However, it is model-based and relies on a rigid-body assumption, limiting its applicability to a broader class of objects. To achieve greater object generalization, a shift towards model-free representations is essential, which then motivates the adoption of flow-based approaches.

### B. Flow-based Manipulation

Flow describes object motion by tracking the displacement of 2D pixels or 3D points between video frames. This offers a more generalizable representation of object dynamics compared to 6D pose, as it is inherently model-free and makes no assumptions about object rigidity. Recent work has shown success in using flow for manipulation [20, 25, 38–41]. However, these methods require robot data or task-specific training for either the flow generator or the executor [20, 25, 38–42]. To achieve greater generalization for zero-shot manipulation, Chen et al. [43] and Zhi et al. [44] train a flow generator on a collection of large-scale human egocentric datasets. While making a great step towards generalization, we empirically find that the generalizability of this approach [43] (understanding of in-the-wild object motion) is still not as good as the commonsense motion knowledge from pretrained video models.

## III. NovaFlow

NovaFlow enables robots to autonomously solve a wide variety of manipulation tasks by leveraging pretrained video generation models, thus eliminating the need for demonstrations or task-specific tuning. Since raw video pixels cannot be directly used by a robot's controller or model-based planner,

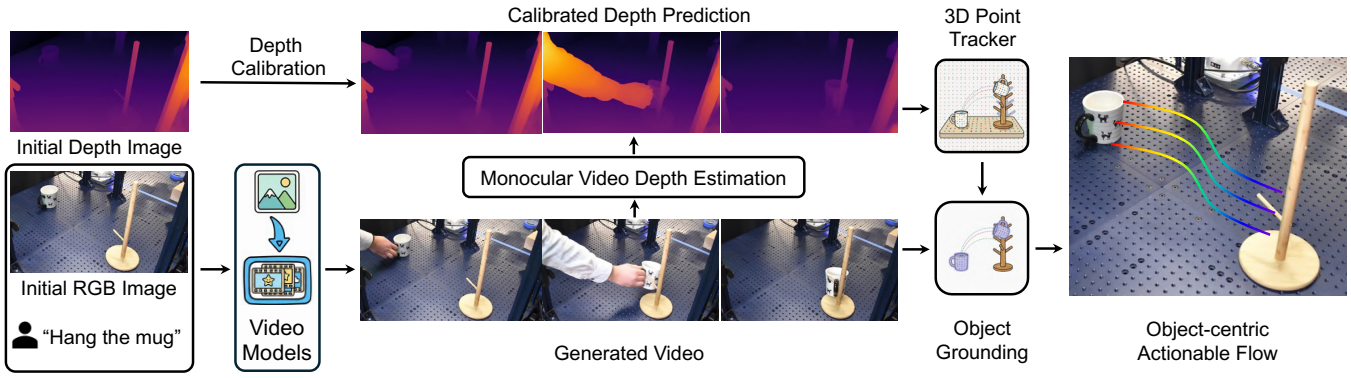


Fig. 2: **Flow generator pipeline.** Given an initial image and a task prompt, a video model is used to generate a video of the plausible object motion. This video is then processed by pretrained perception modules to distill an actionable 3D object flow. This involves (1) lifting the 2D video to 3D using monocular depth estimation, (2) calibrating the estimated depth against the initial depth, (3) tracking the dense per-point motion using 3D point tracking, and (4) extracting the object-centric 3D flow via object grounding.

NovaFlow handles this challenge by distilling the video's implicit commonsense knowledge of motion into a more actionable, intermediate representation: 3D object flow. The proposed pipeline consists of two core components: a *flow generator* (Fig. 2) that extracts the actionable 3D object flow from the generated video, and a *flow executor* (Fig. 3) that translates this flow into robot actions. The entire pipeline is demonstration-free and embodiment-agnostic, requiring no robot-specific data or training before task execution.

### A. Flow Generator

The primary objective of the flow generator is to translate a high-level task description into a structured, actionable flow for the robot. The standard input to the generator is the task description, which involves an initial RGB-D image pair $\{\mathbf{I}, \mathbf{D}\}$ captured from the robot's perspective (with known camera intrinsics) and a natural language instruction, $\mathbf{l}$, describing the desired task. For tasks requiring greater precision, an optional goal image, $\mathbf{I}_g$, can also be provided, either specified by the user [41, 45, 46] or generated by an image editing model with the standard input [47]. Based on the given input, the generator's goal is to produce a 3D object flow across $T$ frames for $M$ object keypoints, $\mathcal{F} \in \mathbb{R}^{T \times M \times 3}$.

The flow generator synthesizes a video using the task description and then distills an *actionable 3D object flow* from the generated video with a pipeline of pre-trained perception modules. The full process involves five main steps: (1) generating the video from the initial image and text prompt, (2) lifting the 2D video to 3D, (3) calibrating the estimated depth, (4) tracking dense per-point 3D motion, and (5) extracting the final object-centric 3D flow.

*1) Video Generation:* Given the initial image $\mathbf{I}$ and language prompt $\mathbf{l}$, a video generation model produces a video $\hat{\mathbf{V}} = \{\hat{\mathbf{I}}_1, \hat{\mathbf{I}}_2, \ldots, \hat{\mathbf{I}}_T\}$ of $T$ frames, known as image-to-video (I2V) generation. If a goal image $\mathbf{I}_g$ is provided, we use first-last-frame-to-video (FLF2V) generation instead.

*2) Monocular Depth Estimation:* To obtain 3D motion information, we lift the generated 2D video into 3D space. We apply a monocular video depth estimation model to $\hat{\mathbf{V}}$, which processes the video frame-by-frame to yield a sequence of estimated metric depth maps $\hat{\mathbf{D}} = \{\hat{\mathbf{D}}_1, \hat{\mathbf{D}}_2, \ldots, \hat{\mathbf{D}}_T\}$.

*3) Depth Calibration:* The depth maps $\hat{\mathbf{D}}$ obtained in the last step have a key limitation: the monocular depth estimation process is inherently ill-posed and often creates metric outputs with systematic scaling errors, especially on generated videos. This can hinder manipulation tasks that require accurate spatial alignment. To correct for this, we calibrate the entire estimated depth sequence $\hat{\mathbf{D}}$ by anchoring it to the initial ground-truth depth map. This calibration leverages the observation that estimated depth, while globally inaccurate, is often locally consistent. We compute a scaling factor between the median depth of the first estimated frame $\hat{\mathbf{D}}_1$ and the initial ground-truth depth map $\mathbf{D}$. While other methods exist, e.g., fitting an affine transformation [12], we find this median scaling factor method to be more stable.

*4) 3D Point Tracking:* With the generated video and the calibrated depth, we extract dense per-point 3D motion. We employ a 3D point tracking model, which takes the camera intrinsics, video $\hat{\mathbf{V}}$, the calibrated depth $\hat{\mathbf{D}}$, and a set of query points $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_M\}$ evenly sampled on the first frame as input. The model outputs a set of 3D trajectories $\mathcal{P} = \{\mathbf{p}_1^t, \ldots, \mathbf{p}_M^t\}_{t=1}^T$, where $\mathbf{p}_i^t$ is the 3D position of the $i$-th query point at timestep $t$.

*5) Object Grounding:* The dense 3D trajectories $\mathcal{P}$ capture the motion of the entire scene. To derive an actionable plan, we must now ground this motion by isolating only the trajectories belonging to the target objects. We achieve this by employing a pipeline that combines an open-vocabulary object detector with a video segmentation model, which produces a sequence of masks, $\mathcal{M} = \{\mathbf{m}_1, \ldots, \mathbf{m}_T\}$, that segment the object across the entire video. Lastly, by applying these masks, we filter the dense trajectories $\mathcal{P}$ to distill the actionable 3D object flow, $\mathcal{F} = \{\mathbf{f}_i^t \mid i = 1, \ldots, K;\ t = 1, \ldots, T\}$. This final output represents the $K$ keypoints that remain consistently tracked on the object's surface.

We conclude object grounding with a rejection sampling step to filter out hallucinations, such as generative artifacts

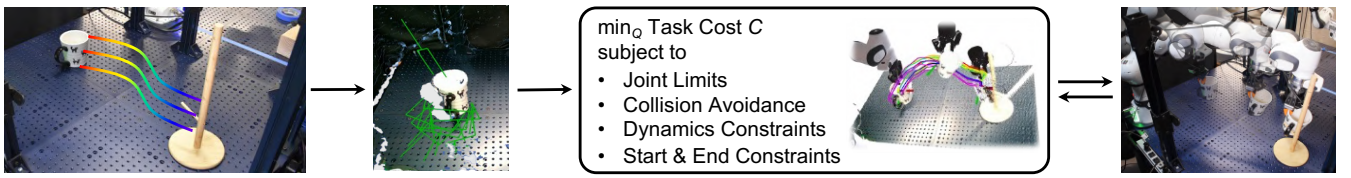


Fig. 3: **Flow executor pipeline.** The initial end-effector pose is determined from grasp proposal candidates. Robot trajectories are then planned based on the actionable flow considering costs and constraints, and subsequently tracked by the robots.

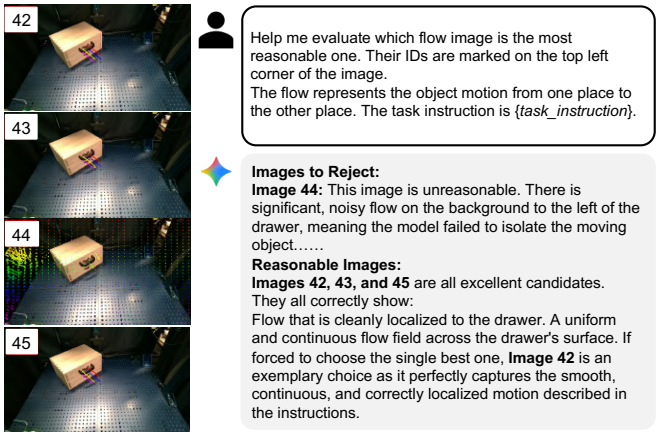


Fig. 4: **Rejection sampling for flow generator.** We generate multiple video candidates in parallel and create the object flow image for each by back-projecting its object flow, $\mathcal{F}$, onto the initial frame. A VLM (in our case, Google Gemini) evaluates all the flow images to select the most plausible video candidate.

and implausible motions, that may be unavoidably introduced by the video generation model (Fig. 4). Here, we use a VLM to validate and select the most plausible generated flow. Specifically, we generate $N$ video candidates simultaneously and obtain $N$ corresponding object flow images by back-projecting object flow $\mathcal{F}$ to the first frame of each video. We then mark each flow image using its ID and pass it into a VLM along with its task description to select the most plausible one. We empirically find that rejecting the flow image is more effective than rejecting the concatenated raw video images [12], since the flow images, explicitly visualizing motion, are a more grounded and direct input for VLMs to reason and understand.

### B. Flow Executor

The flow executor is responsible for translating the abstract 3D object flow $\mathcal{F}$ as planned trajectories into a sequence of executable robot actions $\mathcal{A} = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_T\}$ over $T$ timesteps. The input actionable flow serves as an intermediate representation that describes the desired trajectories of $K$ points on the target object over $T$ timesteps, carrying the high-level task understanding from pre-trained video models. Here, we present an open-loop planner that can be extended to a closed-loop tracking system by incorporating live object trackers. The current executor pipeline can handle two main classes of objects: rigid (including articulated objects, treated as part-wise rigid) and deformable.

*1) Rigid Object Manipulation:* For rigid body manipulation, the 3D flow of the keypoints, $\mathcal{F}$, can be used to estimate the rigid transforms, $(\mathbf{R}, \mathbf{t})$, of the object across frames in a model-free manner. In cases where the object is firmly grasped and moves rigidly with the end-effector (*e.g.*, no slippage), a common assumption in prior work [12, 40, 41], the end-effector pose can be calculated from the object pose. The object-specific firm grasps are selected from the object point cloud using a grasp proposal model, as shown in Fig. 3. At each timestep $t$, we find the rigid transformation $(\mathbf{R}^t, \mathbf{t}^t)$ that aligns the initial keypoints $\{\mathbf{f}_i^1\}_{i=1}^K$ to the current one $\{\mathbf{f}_i^t\}_{i=1}^K$. This is solved using the Kabsch algorithm [30], which finds the optimal rotation $\mathbf{R}^t$ that minimizes the sum of squared errors:

$$\mathbf{R}^t = \underset{\mathbf{R} \in SO(3)}{\operatorname{argmin}} \sum_{i=1}^{K} \|\mathbf{R}(\mathbf{f}_i^1 - \mathbf{c}^1) - (\mathbf{f}_i^t - \mathbf{c}^t)\|^2, \quad (1)$$

where $\mathbf{c}^1$ and $\mathbf{c}^t$ are the masked point cloud centroids at the first and current timesteps, respectively. This optimization can be solved efficiently using Singular Value Decomposition (SVD). Once the rotation is found, the translation is computed as $\mathbf{t}^t = \mathbf{c}^t - \mathbf{R}^t\mathbf{c}^1$. The object pose at timestep $t$ can be represented as a homogeneous transformation matrix $\mathbf{T}_{obj}^t \in SE(3)$, constructed from $\mathbf{R}^t$ and $\mathbf{t}^t$. The resulting sequence of 6D object poses is converted into an end-effector trajectory by applying a grasp transformation, $\mathbf{T}_{grasp}$, obtained from a grasping network [48, 49]. The target end-effector pose at each timestep is then:

$$\mathbf{T}_{ee}^t = \mathbf{T}_{obj}^t \cdot \mathbf{T}_{grasp}. \quad (2)$$

This Cartesian pose is converted to joint commands via trajectory optimization for execution by the robot's controller.

*2) Deformable Object Manipulation:* Unlike rigid objects, deformable objects have complex dynamics that cannot be described by a simple rigid transformation. NovaFlow can be naturally extended to handle deformable objects, with the 3D object flow $\mathcal{F}$ serving as a dense tracking objective for model-based planning. Specifically, we employ a particle-based dynamics model $f_\theta$ to predict the object's future state, where $\theta$ represents the learnable parameters of the model. The state of the object at time $t$ is represented by a set of $N_p$ particles $\mathcal{S}_t = \{\mathbf{s}_i^t\}_{i=1}^{N_p}$. The dynamics model predicts the next state based on the current state and a robot action $\mathbf{a}_t$: $\mathcal{S}_{t+1} = f_\theta(\mathcal{S}_t, \mathbf{a}_t)$.

Conventional methods for deformable manipulation often define a cost function using a correspondence-free metric,

like the Chamfer distance, to a single goal state $\mathcal{S}_{goal}$ [31, 32, 50–52]. Our actionable 3D object flow $\mathcal{F} = \{\mathcal{F}^t\}_{t=1}^{T}$, where $\mathcal{F}^t = \{\mathbf{f}_i^t\}_{i=1}^{N_p}$, allows us to define a cost function based on the sum of squared Euclidean distances, leveraging the explicit point-wise correspondences from the flow:

$$C(\mathcal{S}_t, \mathcal{F}^t) = \sum_{i=1}^{N_p} \|\mathbf{s}_i^t - \mathbf{f}_i^t\|^2. \tag{3}$$

This formulation has two potential advantages. First, using point correspondences may create a better-conditioned optimization landscape, as correspondence-free metrics can be susceptible to local minima. Second, tracking a dense flow provides intermediate targets along a desired motion path, rather than relying on only a final goal configuration.

We then frame the control problem as a Model Predictive Control (MPC) task. At each timestep $t$, we solve for an optimal sequence of actions $\mathbf{A}_t^* = \{\mathbf{a}_t^*, \ldots, \mathbf{a}_{t+H-1}^*\}$ over a planning horizon $H$ by minimizing the cumulative cost:

$$\mathbf{A}_t^* = \underset{\mathbf{A}_t}{\operatorname{argmin}} \sum_{j=t}^{j+H-1} C(\mathcal{S}_j, \mathcal{F}^j), \tag{4}$$

subject to the *dynamics constraints* $\mathcal{S}_{j+1} = f_\theta(\mathcal{S}_j, \mathbf{a}_j)$. We then execute the first action $\mathbf{a}_t^*$ and repeat the optimization at the next timestep.

**Trajectory optimization.** To enable smooth and collision-free motion, we additionally incorporate trajectory optimization to refine the sequence of actions. We formulate the trajectory generation as a non-linear least-squares problem. The goal is to find an optimal sequence of joint configurations $Q = \{q_0, q_1, \ldots, q_{T-1}\}$ that minimizes a sum-of-squares objective function. The trajectory is initialized by linearly interpolating between start and end configurations, $q_{\text{start,IK}}$ and $q_{\text{end,IK}}$, which are pre-calculated using an IK solver using the end-effector pose. The optimal trajectory $Q^*$ is found by solving the constrained non-linear optimization problem:

$$\begin{aligned} \min_{Q} \quad & w_s \mathcal{C}_{\text{smooth}} + w_r \mathcal{C}_{\text{rest}}, \quad \text{subject to} \\ & q_0 = q_{\text{start,IK}} \quad \text{and} \quad q_{T-1} = q_{\text{end,IK}}, \\ & q_{\min} \le q_t \le q_{\max}, \quad \forall t \in \{0, \ldots, T-1\}, \\ & d_s(q_t, q_{t+1}, O_j) \ge \epsilon_{\text{safe}}, \quad \forall t, \forall O_j \in \text{Obstacles}. \end{aligned} \tag{5}$$

In this formulation, the objective function seeks to minimize a weighted sum of the motion smoothness cost ($\mathcal{C}_{\text{smooth}}$) and the rest pose regularization cost ($\mathcal{C}_{\text{rest}}$). Constraints in the optimization include: (1) *start and end constraints*, meaning the trajectory's start and end configurations ($q_0$ and $q_{T-1}$) must exactly match the predefined goals ($q_{\text{start, IK}}$ and $q_{\text{end, IK}}$); (2) *collision avoidance*, by enforcing the signed distance, $d_s$, between the robot and any obstacle to remain greater than a safety margin, $\epsilon_{\text{safe}}$, at all times; (3) *joint limits*, ensuring the robot's physical joint position and velocity limits throughout the entire motion. We treat these constraints as cost terms and use the Levenberg-Marquardt solver to solve the non-linear least-squares problem.

## IV. Experiments

We aim to demonstrate the generalizability of NovaFlow across different object types and embodiments and to show the importance of each component in our framework. We evaluate the framework's ability to execute a broad range of manipulation tasks involving rigid, articulated, and deformable objects across embodiments without requiring task-specific demonstrations or additional fine-tuning.

### A. Implementation Details

We implement NovaFlow with modular, swappable components. For video generation, we use the open-source model Wan [9], which produces 41 frames per task (16 FPS, 1280×720). We estimate depth with MegaSaM [26] using calibrated intrinsics, track 3D points with TAPIP3D [27], and ground objects via Grounded-SAM2 [54] (Grounding DINO [28] + SAM2 [29]). We use a trained PhysTwin [31] model to predict particle dynamics for deformable objects. All modules are drop-in replaceable with newer models, improving speed and robustness, which is another benefit for our modular framework.

### B. Real-World Experiments and Evaluation Tasks

We evaluate NovaFlow on a Franka arm with a Robotiq-85 gripper for table-top manipulation and a Spot quadruped for mobile robot manipulation. For rigid and articulated objects, we use a single RealSense D455 depth camera as input. For deformable objects, we use three synchronized cameras (as required by PhysTwin [31]), though a single-view setup is also possible [52].

We categorize our tasks by the object type involved as rigid (**R**), articulated (**A**), and deformable (**D**):

- **Hanging a mug (R)**: hang a mug on a wooden rack, requiring accurate relative pose placement for the handle to pass through the wooden stick on the rack.
- **Inserting a block (R)**: insert a yellow block into a hole in a board, a task similar to peg-in-hole that requires accurate insertion skills.
- **Placing a cup on a saucer (R)**: place a cup on a saucer, a task demanding accurate placement skills.
- **Watering a plant (R)**: pour water from a green cup into a plant pot, requiring language understanding and manipulation skills.
- **Opening a drawer (A)**: open a drawer, requiring a precise understanding of its articulation.
- **Straightening a rope (D)**: straighten a curved rope, which requires understanding the dynamics of a deformable object.

During evaluation, we randomize the object placement after each trial. We report the quantitative and qualitative results in Fig. 5 and Fig. 6.

### C. Comparison with Baselines

We compare NovaFlow against two groups of baselines. (+) denotes methods requiring external training data, while (*) denotes baselines adapted to fit our pipeline.

**Demo-free, zero-shot baselines (similar to ours):**

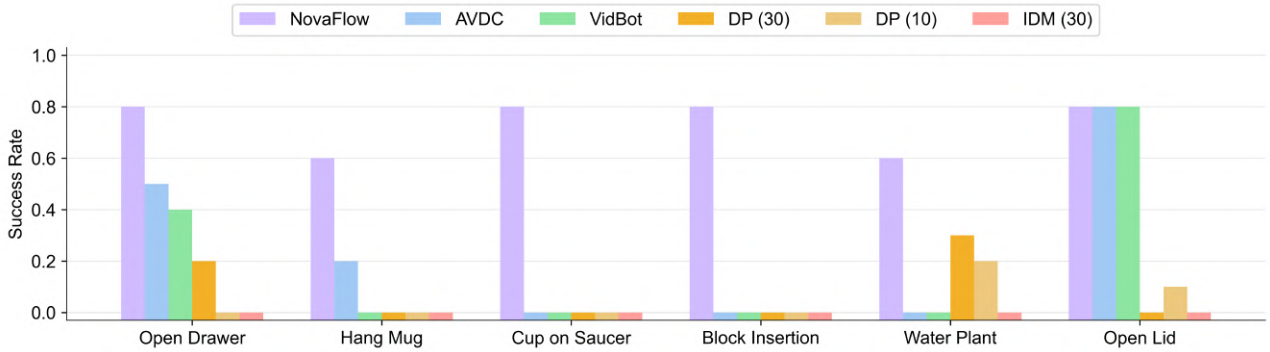


Fig. 5: **Experiment results.** We compare against Diffusion Policy (DP) [53] trained using 10 and 30 demonstrations, inverse dynamics model (IDM) from UniPi [22], AVDC [25], and VidBot [43] in real-world tabletop manipulation tasks.

- **AVDC [25] (*):** Extracts object-centric motion using optical flow. We adapt it to our pipeline by applying it directly to generated videos.
- **VidBot [43]:** Learns flow from large-scale human interaction datasets to model affordances.

**Data-dependent baselines (require demonstrations):**

- **Diffusion Policy (DP) [53] (+):** Diffusion policy serves as an imitation policy baseline trained under very few demos for a single task. We train with 10 and 30 demonstrations per task, using the same single-view camera RGB input as our approach.
- **Inverse Dynamics Model (IDM) [22] (+):** IDM was originally designed to train together with a fine-tuned video generation model using in-domain demonstrations. Since video fine-tuning is outside our scope, we trained the IDM model with the 30 demonstrations previously used in DP training to convert generated robot task-solving videos (from Wan2.1) into robot actions.

NovaFlow achieves the highest success rates across tasks among zero-shot methods and also surpasses data-dependent baselines trained with 10–30 demonstrations, as shown in Fig. 5 (with 10 trials for each task). **AVDC(*)** performs competitively on affordance-like tasks but struggles with precise, long-horizon placements. In our setup, it distills motion from 2D optical flow, lacking 3D awareness and long-term coherence under occlusion. These limitations, as also noted in the AVDC paper, cause the method to struggle with tasks requiring accurate placement and rotation-heavy motions. **VidBot** excels on affordance-centric, articulated interactions (e.g., "open drawer") but fails when tasks require object–object relations and precise relative pose placement. This matches our diagnosis that its training emphasizes object–affordance understanding rather than modeling multi-object constraints. For **DP(+)**, despite per-task training (an easier setting that bypasses language understanding), it still shows poor generalization from a few examples, especially because our evaluations are randomly sampled and not drawn from the training distribution. The main issue for **IDM(+)** is the domain shift between its training and test data. The inverse dynamics model learns from real-world robot demos, yet it must interpret generated videos whose motion is not always kinematically perfect or consistent. Consequently, the generated videos are out-of-distribution, causing the model to fail even if the video's high-level action seems semantically reasonable.

Overall, methods that (i) lack an actionable *3D* representation (AVDC, VidBot) or (ii) rely on small, task-specific robot datasets (DP, IDM) fail to provide zero-shot autonomous task-solving. Distilling a dense, actionable 3D object flow and decoupling understanding from control is critical for zero-shot generalization.

### *D. Ablation Studies and Failure Analysis*

Here we discuss some of the design choices in NovaFlow's submodules. For video generation, we compare the current Wan 2.1 model to the closed-source model Veo [55], which produces 8 s clips (24 FPS). Prompt extension is utilized for better controllability. For precise placement tasks (e.g., mug on rack and block insertion), we optionally condition on a goal image (FLF2V) instead of I2V.

We analyze the failure cases of NovaFlow in Fig. 7, identifying four primary failure modes. **Video failure** occurs when the generative model produces content that is not physically plausible, lacks 3D consistency, or violates the user's instructions; our rejection sampling with a VLM mitigates but does not eliminate this. **Tracking failure** results from inaccuracy in 3D point tracking, often caused by textureless surfaces, heavy occlusions, or accumulated inconsistencies inherited from the video model. **Grasp failure** happens when the robot fails to secure the object correctly (e.g., bad approach, missed grasp, and slip). Finally, **execution failure** encompasses errors during trajectory execution, such as collisions, joint limits, or an inability to follow the planned path accurately. Our analysis reveals that most failures occur in the *last mile*: grasp and execution are the most frequent, suggesting that while the upstream flow estimation is relatively robust, physical interaction remains the bottleneck. This is similar to the sim-to-real gap in simulation-based training. To address these limitations, future work could focus on integrating a closed-loop feedback system to enable dynamic replanning and refine the generated flow in response to observations.

We investigate the effect of a goal image on the block insertion task requiring millimeter-level precision (Tab. I). We use two metrics: Video Success Rate, the percentage of

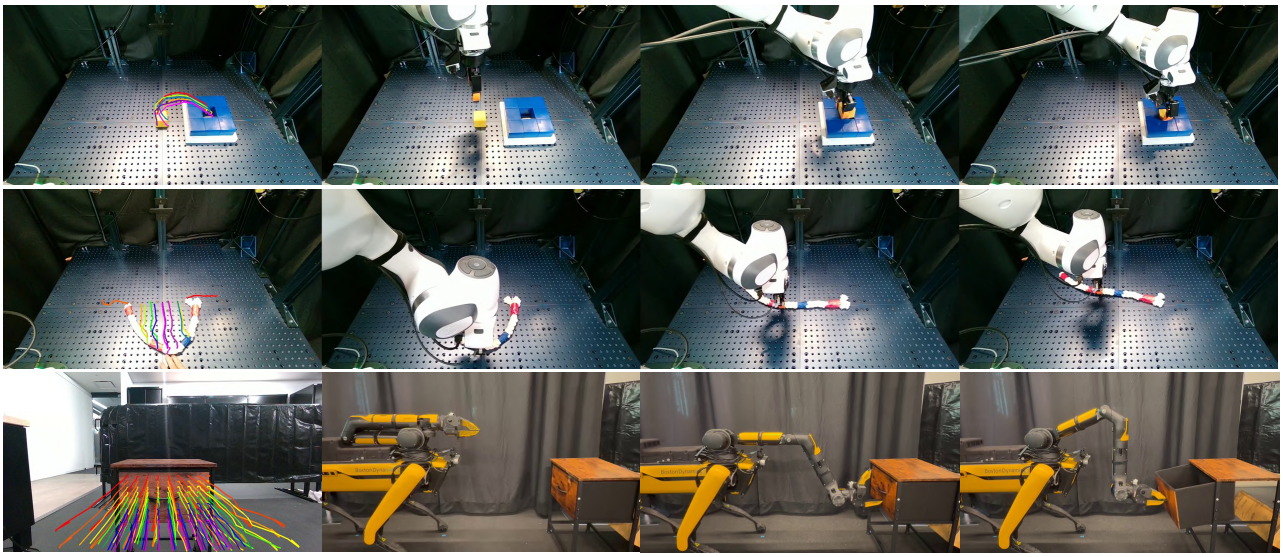

Fig. 6: **Real-world manipulation experiments.** NovaFlow is versatile and supports cross-embodiment manipulation, which we use to manipulate rigid, deformable, and articulated objects using tabletop and mobile manipulator.

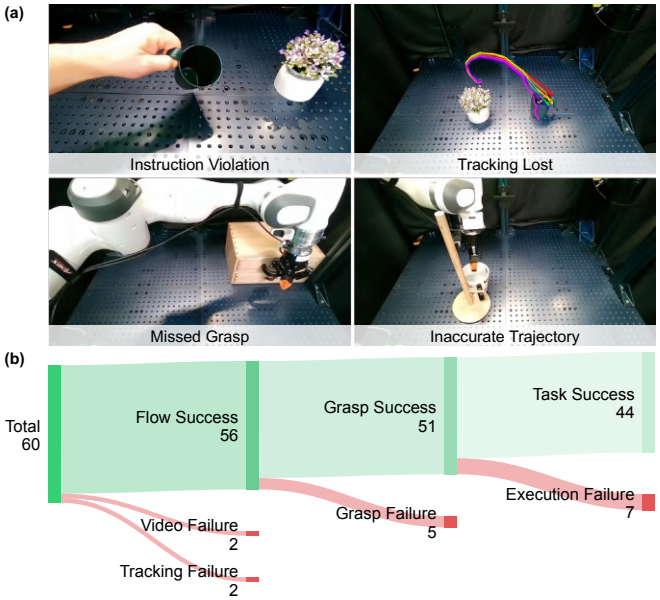


Fig. 7: **Failure analysis.** (a) Examples of video, tracking, grasp, and execution failures. (b) Failure cause distribution.

TABLE I: Effect of goal image on block insertion task.

| Condition | Video Success | Task Success | Time (s) |
|---|---|---|---|
| w/ Goal Image (Wan2.1) | 46% | 80% | 612 |
| w/o Goal Image (Wan2.1) | 15% | 40% | 612 |
| w/o Goal Image (Veo) | 75% | 80% | 20 |

generated videos with a valid actionable flow, and Task Success Rate, the execution success of a flow selected by a VLM after rejection sampling. For each trial, we synthesize eight videos, from which the VLM selects the best for execution. Our results show that omitting the goal image significantly impairs the performance of the open-source Wan2.1 model. While VLM rejection sampling improves the final success rate, the drop remains substantial. In contrast, the closed-source Veo model proves more robust, outperforming Wan2.1 and achieving a high task success rate even without a goal image.

TABLE II: Runtime analysis. Time is measured in seconds.

| | MegaSaM | TAPIP3D | SAM2 | **Total (Veo)** | Total (Wan) |
|---|---|---|---|---|---|
| **Time** | 100 | 5 | 8 | **133** | 725 |

### *E. Runtime Analysis*

We deploy NovaFlow on a single NVIDIA H100 GPU, and a complete flow generation takes around 2 minutes end-to-end (Veo). We report per-module timings in Tab. II to guide replacements and optimization. The dominant time-consuming modules are the video generation and 3D lifting modules. For video generation, closed-source models are usually much faster in time but more expensive in cost.

## V. Conclusion

We introduced NovaFlow, a demonstration-free framework for autonomous manipulation that translates natural language commands into robot actions by leveraging the commonsense knowledge embedded in large-scale video generation models. Our key insight is to distill generated task-solving videos into an actionable 3D object flow, an intermediate representation that decouples high-level task understanding from low-level robot control. This modular design enables NovaFlow to handle rigid, articulated, and deformable objects across different robot embodiments without requiring any task-specific training or demonstrations. Our real-world experiments show that NovaFlow not only outperforms other zero-shot methods but also surpasses imitation learning policies trained on dozens of demonstrations.

Despite its success, our failure analysis reveals that the primary bottleneck is the physical execution phase, particularly in grasping and handling unexpected dynamics. This highlights a gap between the open-loop plan generated from video and the complexity of real-world interaction. A promising direction for future work is to develop a closed-loop system

where real-time feedback from the environment is used to refine or replan the generated flow, making the system more adaptive and robust to unforeseen challenges [56].

## REFERENCES


[1] M. J. Kim *et al.*, "OpenVLA: An Open-Source Vision-Language-Action Model," in *Conference on Robot Learning*, Jan. 2025.

[2] A. Brohan *et al.*, "RT-1: Robotics Transformer for Real-World Control at Scale," Dec. 2022.

[3] T. L. Team *et al.*, "A Careful Examination of Large Behavior Models for Multitask Dexterous Manipulation," Jul. 2025.

[4] Qwen *et al.*, "Qwen2.5 Technical Report," Jan. 2025.

[5] DeepSeek-AI *et al.*, "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning," Jan. 2025.

[6] H. Liu *et al.*, "Visual Instruction Tuning," in *Advances in Neural Information Processing Systems*, vol. 36, Dec. 2023.

[7] OpenAI *et al.*, "GPT-4o System Card," Oct. 2024.

[8] W. Kong *et al.*, "HunyuanVideo: A Systematic Framework For Large Video Generative Models," Mar. 2025.

[9] T. Wan *et al.*, "Wan: Open and Advanced Large-Scale Video Generative Models," Apr. 2025.

[10] A. Stone *et al.*, "Open-World Object Manipulation using Pre-Trained Vision-Language Models," in *Conference on Robot Learning*, Dec. 2023.

[11] M. Dalal *et al.*, "Local Policies Enable Zero-Shot Long-Horizon Manipulation," in *2025 IEEE International Conference on Robotics and Automation*, May 2025, pp. 13 875–13 882.

[12] S. Patel *et al.*, "Robotic Manipulation by Imitating Generated Videos Without Physical Demonstrations," Jul. 2025.

[13] W. Yu *et al.*, "Language to Rewards for Robotic Skill Synthesis," in *Proceedings of The 7th Conference on Robot Learning*. PMLR, Dec. 2023, pp. 374–404.

[14] K. Goldberg, "Good old-fashioned engineering can close the 100,000-year "data gap" in robotics," *Science Robotics*, vol. 10, no. 105, p. eaea7390, Aug. 2025.

[15] K. Black *et al.*, "Zero-Shot Robotic Manipulation with Pre-Trained Image-Editing Diffusion Models," in *International Conference on Learning Representations*, Oct. 2023.

[16] H. Bharadhwaj *et al.*, "Towards Generalizable Zero-Shot Manipulation via Translating Human Interaction Plans," in *IEEE International Conference on Robotics and Automation*, May 2024.

[17] W. Huang *et al.*, "ReKep: Spatio-Temporal Reasoning of Relational Keypoint Constraints for Robotic Manipulation," in *Conference on Robot Learning*, Jan. 2025.

[18] G. Yin *et al.*, "CodeDiffuser: Attention-Enhanced Diffusion Policy via VLM-Generated Code for Instruction Ambiguity," in *Robotics: Science and Systems XXI*. arXiv, Jun. 2025.

[19] J. Liang *et al.*, "Code as Policies: Language Model Programs for Embodied Control," in *IEEE International Conference on Robotics and Automation*, May 2023.

[20] M. Xu *et al.*, "Flow as the Cross-domain Manipulation Interface," in *Conference on Robot Learning*, Jan. 2025.

[21] H. Huang *et al.*, "IMAGINATION POLICY: Using Generative Point Cloud Models for Learning Manipulation Policies," in *Conference on Robot Learning*, Sep. 2024.

[22] Y. Du *et al.*, "Learning Universal Policies via Text-Guided Video Generation," in *Advances in Neural Information Processing Systems*, Dec. 2023.

[23] J. Liang *et al.*, "Dreamitate: Real-World Visuomotor Policy Learning via Video Generation," in *Conference on Robot Learning*, Jan. 2025.

[24] S. Li *et al.*, "Unified Video Action Model," in *Robotics: Science and Systems XXI*, vol. 21, Jun. 2025.

[25] P.-C. Ko *et al.*, "Learning to Act from Actionless Videos through Dense Correspondences," in *International Conference on Learning Representations*, Oct. 2023.

[26] Z. Li *et al.*, "MegaSaM: Accurate, Fast and Robust Structure and Motion from Casual Dynamic Videos," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[27] B. Zhang *et al.*, "TAPIP3D: Tracking Any Point in Persistent 3D Geometry," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, Oct. 2025.

[28] S. Liu *et al.*, "Grounding DINO: Marrying DINO with Grounded Pre-training for Open-Set Object Detection," in *European Conference on Computer Vision*, 2025.

[29] N. Ravi *et al.*, "SAM 2: Segment Anything in Images and Videos," in *International Conference on Learning Representations*, Oct. 2024.

[30] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, vol. 32, no. 5, pp. 922–923, Sep. 1976.

[31] H. Jiang *et al.*, "PhysTwin: Physics-Informed Reconstruction and Simulation of Deformable Objects from Videos," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025, pp. 7219–7230.

[32] K. Zhang *et al.*, "Particle-Grid Neural Dynamics for Learning Deformable Object Models from RGB-D Videos," in *Robotics: Science and Systems*, Jun. 2025.

[33] J. Lee *et al.*, "MolmoAct: Action Reasoning Models that can Reason in Space," Aug. 2025.

[34] K. Black *et al.*, "$\pi_0$: A Vision-Language-Action Flow Model for General Robot Control," Nov. 2024.

[35] A. Ajay *et al.*, "Compositional Foundation Models for Hierarchical Planning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 22 304–22 325, Dec. 2023.

[36] H. Bharadhwaj *et al.*, "Gen2Act: Human Video Generation in Novel Scenarios enables Generalizable Robot Manipulation," Sep. 2024.

[37] J. Liang *et al.*, "Video Generators are Robot Policies," Aug. 2025.

[38] C. Yuan *et al.*, "General Flow as Foundation Affordance for Scalable Robot Learning," in *Proceedings of The 8th Conference on Robot Learning*. PMLR, Jan. 2025, pp. 1541–1566.

[39] B. Eisner *et al.*, "FlowBot3D: Learning 3D Articulation Flow to Manipulate Articulated Objects," in *Robotics: Science and Systems XVIII*, vol. 18, Jun. 2022.

[40] Z.-H. Yin *et al.*, "Object-centric 3D Motion Field for Robot Learning from Human Videos," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, Oct. 2025.

[41] H. Bharadhwaj *et al.*, "Track2Act: Predicting Point Tracks from Internet Videos Enables Generalizable Robot Manipulation," in *European Conference on Computer Vision*, 2024.

[42] H. Zhang *et al.*, "FlowBot++: Learning Generalized Articulated Objects Manipulation via Articulation Projection," in *7th Annual Conference on Robot Learning*, Aug. 2023.

[43] H. Chen *et al.*, "VidBot: Learning Generalizable 3D Actions from In-the-Wild 2D Human Videos for Zero-Shot Robotic Manipulation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[44] H. Zhi *et al.*, "3DFlowAction: Learning Cross-Embodiment Manipulation from 3D Flow World Model," Jun. 2025.

[45] J. Shi *et al.*, "ZeroMimic: Distilling Robotic Manipulation Skills from Web Videos," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, May 2025, pp. 16 939–16 947.

[46] G. Zhou *et al.*, "DINO-WM: World Models on Pre-trained Visual Features enable Zero-shot Planning," in *International Conference on Machine Learning*, Jun. 2025.

[47] B. F. Labs *et al.*, "FLUX.1 Kontext: Flow Matching for In-Context Image Generation and Editing in Latent Space," Jun. 2025.

[48] A. Murali *et al.*, "GraspGen: A Diffusion-based Framework for 6-DOF Grasping with On-Generator Training," Jul. 2025.

[49] H.-S. Fang *et al.*, "AnyGrasp: Robust and Efficient Grasp Perception in Spatial and Temporal Domains," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3929–3945, Oct. 2023.

[50] K. Zhang *et al.*, "AdaptiGraph: Material-Adaptive Graph-Based Neural Dynamics for Robotic Manipulation," in *Robotics: Science and Systems*, vol. 20, Jul. 2024.

[51] Y. Wang *et al.*, "Dynamic-Resolution Model Learning for Object Pile Manipulation," in *Robotics: Science and Systems*, Jul. 2023.

[52] S. Huang *et al.*, "ParticleFormer: A 3D Point Cloud World Model for Multi-Object, Multi-Material Robotic Manipulation," in *Conference on Robot Learning*, Jul. 2025.

[53] C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," in *Robotics: Science and Systems XIX*, vol. 19, Jul. 2023.

[54] T. Ren *et al.*, "Grounded SAM: Assembling Open-World Models for Diverse Visual Tasks," Jan. 2024.

[55] T. Wiedemer *et al.*, "Video models are zero-shot learners and reasoners," Sep. 2025.

[56] L. Sun *et al.*, "Interactive Planning Using Large Language Models for Partially Observable Robotic Tasks," in *IEEE International Conference on Robotics and Automation*, May 2024.