CASTER: Formalizing Zero-Shot Manipulation from Synthetic Videos as Constrained Optimization

***Abstract*— Prior research on learning from humans often requires real human demonstrations. But advances in video generation models offer a new paradigm: we can now prompt these generators to make *synthetic videos* of the task. On the plus side, these synthetic videos take place in the robot's current environment, and do not require actual human teaching. However, while the generated videos appear to show plausible motions, they lack both physical consistency and robot grounding, and thus the robot cannot directly roll out the trajectory shown by the video. In this paper we seek to formalize the process of going from synthetic video generation to real-world robot trajectories for zero-shot manipulation tasks. We recognize that this problem is an instance of *constrained optimization*. The object trajectories extracted from the synthetic video provide an initial guess at the correct motion, and the robot must refine these trajectories (for instance, within a digital twin) while enforcing physical feasibility and semantic consistency. Our optimization-based formalism provides insights into the video generation properties necessary for success. Specifically, we show that the video generation must probabilistically seed the search within a semantic basin of attraction where the trajectory can be locally modified without losing its task alignment. Our analysis further reveals that state-of-the-art approaches can be viewed as instances of our unified formalism with key simplifications. In our experiments we remove these simplifications to more closely match the underlying optimization, leading to more robust zero-shot trajectories from synthetic videos and improving average task success by 34.1% compared to the best-performing baseline. See videos at our project website: https://vidgenrobotics2026.github.io**

I. Introduction

Collecting robot-centric data is expensive [1], [2]. Videos of humans completing tasks are widely available, but these in-the-wild videos are often unstructured, and occur in settings that are unlike the robot's context. Video generation techniques can help bridge this gap. By prompting video generators to create *synthetic* videos, we gather "human" data without actually requiring human demonstrations — plus, the resulting videos are set in the robot's current environment.

Consider the example in Figure 1. The user prompts the system with a desired task. Given this prompt and an initial image, recent video generation models can output a synthetic video of the task being performed by a human. Ideally, the robot would be able to directly apply this video to guide its own motion. *But there are two key limitations.* First, while the synthetic video may appear visually plausible, it is not necessarily physically feasible. Second, the generated video features a human doing the task, introducing kinematic and visual discrepancies between what the video shows and what the robot arm can actually do.

Our work focuses specifically on using pretrained video generation models *at inference time* to guide robot behavior. That is, rather than using generated videos as offline data for policy training or fine-tuning a video generator on robot-specific data, we consider a setting where a general-purpose, frozen video generator produces task demonstrations during deployment. Existing research searches for ways to address the above problems and unlock the promise of video generation models in this setting. Common approaches typically fall along a spectrum in how much they post-process the video data. At one extreme, the system simply learns to project the object's motion in the synthetic video to real-world robot trajectories [3]. At the other extreme, robots construct a digital twin to physically test and refine the proposed video behavior before applying the results in the real world [4]. We view these methods as effective *ad hoc* solutions, but it remains unclear *how to define the underlying problem*. For example: what should the robot optimize for when trying to map the synthetic video to real-world task performance? And what characteristics does the video generation model need to have in order for this zero-shot transfer to be possible?

In this paper we seek to formalize the process of converting synthetic videos to real-world robot trajectories for zero-shot manipulation tasks. Our key insight is that:

Transferring video generation is a constrained optimization problem. The synthetic video seeds the search, which should maximize trajectory feature alignment while maintaining semantic and feasibility constraints.

Building on this formalism, we introduce **CASTER**, **C**onstraint-**A**ware **S**ynthetic **T**ask-feature **E**xt**R**action, a framework that extracts task-relevant features from generated videos and optimizes robot trajectories within a digital twin. The extracted trajectories serve as initial conditions: within a digital twin, the robot then fine-tunes those trajectories to ensure that its resulting behavior performs the intended object motions without unintended collisions or physically infeasible behaviors. We find that the objective function itself is particularly important. While prior works assume that the robot should match *every* waypoint of the synthetic video, we generalize so that the robot's objective function assigns higher weights to *task-critical* waypoints or features.

Formalizing this optimization provides three fundamental benefits. First, we can unify existing approaches as instances of our optimization problem under different simplifying assumptions. Second, we can use the optimization structure to define the characteristics of the video generation that are necessary for success. Finally, we can develop new methods for zero-shot manipulation using video generation models by more closely approximating the underlying problem. Overall, we make the following contributions:

Defining a Unified Formalism. We define a constrained optimization problem that inputs synthetic task videos and

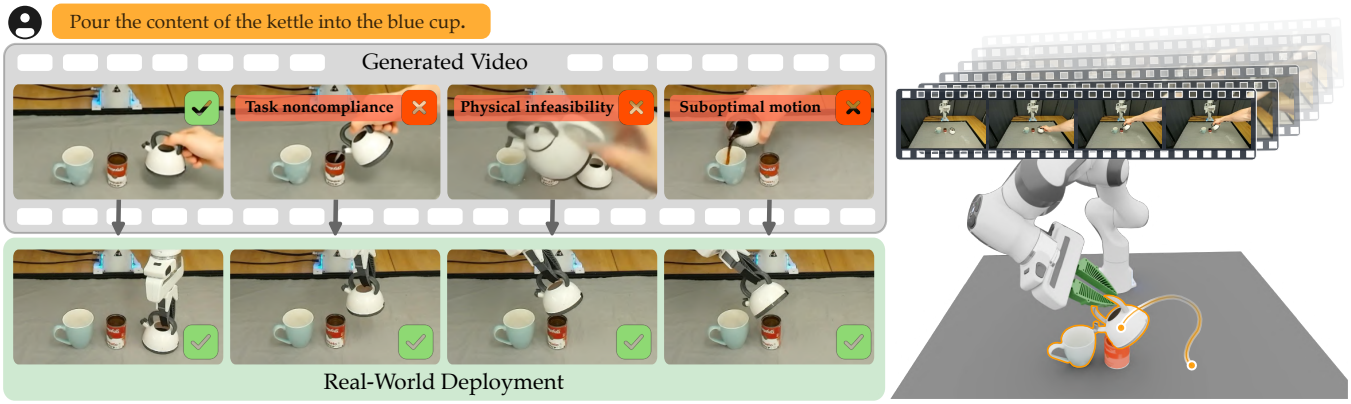

Fig. 1. Using video generation models at inference time to guide real-world robot trajectories. The generated video looks plausible, but may make mistakes that prevent directly transferring this video to the real robot's motion. We bridge the gap between generation and deployment by developing an optimization-based formalism. Under this formalism the robot uses multiple synthetic videos to seed its search and identify task-critical features. By matching these features while enforcing semantic consistency and physical feasibility, we can transfer erroneous videos to successful zero-shot trajectories.

outputs trajectories for real-world execution. This optimization must satisfy physical feasibility (i.e., the motion must be executable without accidental collisions) and semantic consistency (i.e., the motion must be aligned with the task).

Characterizing Video Generation. Building on our theoretical framework, we derive the properties and requirements for the video generation model. We show that simultaneously enforcing feasibility and semantic constraints depends on generating videos that can be locally modified without losing their semantic meaning. Intuitively, tasks with a larger basin of attraction around feasible motions (i.e., many initial guesses converge towards the same equilibrium) are more suitable for synthetic video frameworks.

Developing Approximate Solutions. We demonstrate that state-of-the-art approaches can be viewed as instances of our unified framework. We then derive a new approach that more closely matches the underlying optimization problem by weighting trajectory features instead of matching every waypoint. Our experiments show the pros and cons of these different solution methods.

II. Related Work

Robot Video Generation for Manipulation. Recent works have explored generating videos of robots performing tasks for learning manipulation policies and planning robot actions. Generating the robot directly reduces the differences in embodiment between the video demonstration and robot execution. UniPi [5] fine-tunes a pretrained video generator on robot data to predict language-conditioned visual plans, and uses an inverse-dynamics model to recover robot actions. Video Policy [6] combines video and action diffusion, conditioning the action network on intermediate video features at each denoising step. VAG [7] jointly generates aligned video-action pairs, which can be used as synthetic training data or for learning world-action models.

Other works use generated robot videos to augment the data available for policy learning. DreamGen [8] generates synthetic trajectories and recovers pseudo-action labels for training robot policies. Ctrl-World [9] predicts action-conditioned, multi-view robot rollouts and uses successful trajectories to fine-tune policies. GE-Sim 2.0 [10] additionally predicts proprioceptive states and filters generated experience with a learned judge. For humanoid manipulation, GenHOI [11] recovers contact-aware geometric constraints from generated interaction videos and optimizes trajectories for execution without task-specific policy training.

However, generating consistent robot videos zero-shot remains challenging for general-purpose video models, as robot behavior is sparsely represented in their training data. *These models are trained on large amounts of human video and are better suited to generating human demonstrations in new scenes without fine-tuning* [12]. We therefore use human video generation to predict how a task can be performed, and translate the generated motion into robot actions.

Human Video Generation for Manipulation. Generated human videos can describe tasks across diverse objects and scenes, but do not directly specify executable robot actions. Prior works extract motion from these videos for trajectory optimization or policy learning, including approaches that generate object motion without a visible human.

One line of work recovers trajectories or keyframes and translates them into robot actions through retargeting or trajectory optimization [13]. Dreamitate [14] tracks tool motion, while RiGVid [3] recovers object poses; both use these trajectories to determine robot end-effector motion. NovaFlow [15] recovers 3D object flow, and NovaPlan [16] uses object and hand motion with closed-loop replanning. RoboReact [17] reconstructs interaction keyframes, retargets them to the robot, and refines execution using feedback.

Other works learn policies that use generated videos or recovered motion. Gen2Act [12] trains a closed-loop policy conditioned on generated human videos to perform tasks specified through video generation at deployment. GenMimic [18] trains a physics-aware reinforcement learning policy to track recovered human motion. Dream2Flow [19] reconstructs 3D object flow and uses a tracking objective for either trajectory optimization or reinforcement learning. PhysWorld [4] uses reinforcement learning in a digital twin to obtain executable robot actions that track generated object motions. We seek a generalized formulation that lets us study

these prior approaches within a common framework, analyze when generated motion can successfully transfer to a robot, and explicitly bridge semantic intent and physical feasibility.

III. Problem Statement

We consider the setting of zero-shot tabletop robot manipulation. An example of this setting is shown in Figure 1. The robot is given a natural-language task description τ and an initial visual observation o of a scene containing M objects. The setting is *zero-shot* in the sense that the robot must execute the task within a finite horizon without access to task-specific robot demonstrations (i.e., teleoperation) or any additional real-world interaction.

Trajectories. We assume a fixed tabletop workspace observed by a static camera where task-relevant objects remain at least partially observable during execution. Let s_o be the state of the M objects (e.g., the pose of each item) and let s_r be the robot's state (e.g., its joint and gripper positions). The system state $s = (s_o, s_r)$ includes both the objects and the robot; this state transitions based on robot actions a. In our experiments these actions are changes in joint position of the arm and commands to open or close the gripper. During an interaction of T timesteps the robot executes actions $a^{0:T-1}$, resulting in a sequence $s^{0:T} = (s_o^{0:T}, s_r^{0:T})$ of system states. We therefore define the *trajectories*:

$$\xi = \left(s^{0:T}, a^{0:T-1}\right), \quad \xi_o = s_o^{0:T}, \quad \xi_r = s_r^{0:T} \tag{1}$$

where ξ_o and ξ_r isolate the objects and robot motions. Similarly, we define $\xi_a = a^{0:T-1}$ as the trajectory of robot actions that induce these object and robot motions.

Task Success. For an interaction to be successful the system's trajectory ξ must complete the task τ. This includes both the *semantic* output (e.g., pouring water into the glass) and the *physical* requirements (e.g., avoiding collisions with that glass). We express these requirements as:

$$\mathcal{G}_\tau(\xi) = 1, \quad \xi \in \mathcal{F}_\mathcal{R} \tag{2}$$

where $\mathcal{G}_\tau$ is a task-dependent success predicate and $\mathcal{F}_\mathcal{R}$ is the set of trajectories permitted by the robot embodiment and environment. This feasible set captures constraints such as workspace reachability, joint and velocity limits, collision avoidance, and the ability to establish the contacts required to manipulate relevant objects.

Video Generation. In addition to the initial observation o and task description τ, we assume inference-only access to a pretrained image-and-text-conditioned *video generation model*. Given (o, τ), the video generation model q_{vid} defines a conditional distribution over possible future videos:

$$v \sim q_{\text{vid}}(v \mid o, \tau), \quad \xi_o^{\text{video}} = f_{\text{track}}(v) \tag{3}$$

In Equation (3) v is the raw RGB video and ξ_o^{video} is the motion trajectory of the M objects in that video. For now we assume access to a procedure f_{track} that accurately maps the synthetic video into these object trajectories.

The generator q_{vid} is treated as a *black box*: its parameters, training data, gradients, and internal representations are unavailable, and it cannot be fine-tuned using robot data. The generated videos do not necessarily depict the target robot or provide any explicit geometric, contact, or kinematic information. These videos may therefore describe futures that are semantically consistent with the task but infeasible for the robot, or physically plausible motions that do not accomplish the requested task. Our goal is to use only the robot's observations, its embodiment and control model, and samples obtained from this frozen video generator to perform a manipulation task that satisfies both the semantic success and physical feasibility from Equation (2).

IV. Introducing a Unified Formalism

Generated videos provide insights into the correct behavior through their visual content, but these videos do not directly provide executable robot actions. Prior work has explored several approaches for zero-shot robot manipulation based on synthetic video generation. Rather than starting with another solution, *we begin by focusing on the underlying problem*. More specifically, we formulate leveraging synthetic videos as constrained optimization. In Section IV-A we define the generalized components of this optimization, and in Section IV-B we derive the properties the video generator must satisfy in order for the optimization to converge. Finally, in Section IV-C we show how related approaches can be viewed as instances of our unified formalism with relaxed constraints or different cost functions.

A. Learning from Generated Videos as Constrained Optimization

Given a synthetic video v of task τ, our goal is to extract robot actions ξ_a that perform the same task with a real-world robotic system. When the robot takes these actions, they result in the objects, robot, and system trajectories from Equation (1). Ideally these trajectories produce task success. Consider Equation (2): the behavior resulting from ξ_a must be both semantically aligned and physically feasible. The synthetically generated video does not directly show what actions the robot should take, but in Equation (3) it does provide insight into what the object trajectories could be. Put another way, each synthetic video yields an *initial guess* at the optimal object motions $\xi_{0,v}$. The motion resulting from ξ_a should be *close* to that initial guess, while satisfying the *task success constraints*. We write this formally as a constrained optimization problem:

$$\begin{aligned} \xi_a^* \in \arg\min_{\xi_a} \quad & \left\| \phi(\xi_o) - \phi\big(f_{\text{track}}(v)\big) \right\| \\ \text{subject to:} \quad & \mathbb{P}\left[\mathcal{G}_\tau(\xi) = 1\right] \geq 1 - \epsilon_{\text{sem}} \\ & \mathbb{P}\left[\xi \in \mathcal{F}_\mathcal{R}\right] \geq 1 - \epsilon_{\text{phys}} \end{aligned} \tag{4}$$

where ξ_o and ξ are the trajectories induced by robot actions ξ_a. Below we separately discuss the cost function and constraints from Equation (4).

Cost Function. The optimization cost captures the distance between the object motions in the video and the object motions in the real-world trajectory. More specifically, we focus on the deviation in *task features*. Although task features can be instantiated in various ways, we generally represent

them as relative trajectories between objects. Let $\phi : \Xi_o \to \mathbb{R}^k$ map the M object trajectories into a k-length vector. In our problem setting, trajectory length is constrained by video generation, and we assume that corresponding timesteps across videos represent comparable stages of task execution. One straightforward choice for ϕ is the identity mapping. Here the features are simply the object poses, and minimizing the cost causes the robot's actions to try and recreate every single object motion in the synthetic video.

In practice, however, some aspects of the object motions are more task-critical than others. Consider the task "pour a glass of water." For this task, the exact path the robot follows to carry the water to a cup is not particularly important, provided the robot keeps the water upright. The features ϕ therefore only need to capture the *task-critical* parts of the object motions and inter-object geometric relationships. These features can discard noise and incidental variations in the motions and timing of the generated trajectories.

Semantic Consistency Constraint. Optimizing for the object features alone is insufficient: even if we faithfully reproduce the object trajectories in the video, these trajectories may not complete the intended task. We therefore enforce constraints in Equation (4). To understand how these constraints connect to the properties of the video generation model in Section IV-B, we *expand on their definition here*, starting with the semantic consistency constraint.

Given the task-dependent success predicate $\mathcal{G}_\tau$, we define the corresponding semantic solution manifold for task τ and initial observation o as:

$$\mathcal{M}_{\text{sem}}(\tau, o) = \{\xi_a : \mathcal{G}_\tau(\xi) = 1\} \tag{5}$$

This manifold contains all the action trajectories that achieve the intended task, regardless of their motions and timings. Because our cost function compares trajectories through the lens of object features, we also define the image of the semantic solution manifold in the feature space:

$$\mathcal{M}^{\phi}_{\text{sem}}(\tau, o) = \{\phi(\xi_o) : \mathcal{G}_\tau(\xi) = 1\} \tag{6}$$

where we emphasize that the objects' trajectory ξ_o is induced by robot actions ξ_a. In practice, our optimization procedure may not be able to satisfy the semantic constraint with absolute certainty. We therefore relax the success condition from Equation (2) to reach a more general constraint:

$$\mathbb{P}\left[\mathcal{G}_\tau(\xi) = 1\right] \geq 1 - \epsilon_{\text{sem}} \tag{7}$$

where $\epsilon_{\text{sem}} \in [0, 1)$ is the tolerated probability of semantic failure. In summary, we constrain the actions ξ_a such that the resulting object and robot trajectories complete the intended task τ with high probability.

Physical Feasibility Constraint. The physical feasibility constraint requires the state-action trajectory to remain within the set of behaviors the robot can actually execute. Similar to our semantic approach, we relax Equation (2) to define this constraint probabilistically:

$$\mathbb{P}\left[\xi \in \mathcal{F}_{\mathcal{R}}\right] \geq 1 - \epsilon_{\text{phys}} \tag{8}$$

where $\epsilon_{\text{phys}} \in [0, 1)$ is the tolerated probability of physical constraint violation. The feasible set $\mathcal{F}_{\mathcal{R}}$ is determined by the robot embodiment, the environment, and the interactions required by the task. Given task τ and initial image o let:

$$g_j(\xi; \tau, o) \leq 0, \qquad j = 1, \ldots, J \tag{9}$$

denote the individual physical constraints, such that:

$$\mathcal{F}_{\mathcal{R}} = \{\xi : g_j(\xi; \tau, o) \leq 0,\ j = 1, \ldots, J\} \tag{10}$$

is the feasibility set. In practice, the robot can determine if it satisfies Equation (8) by checking each of the individual constraints in Equation (10).

One standard approach for enforcing physical feasibility is utilizing a *digital twin*. Let $\hat{g}^{\text{sim}}_j$ denote the simulator's approximation of the real constraint g^{real}_j from Equation (9). The policy can then be optimized subject to:

$$\mathbb{P}\left[\hat{g}^{\text{sim}}_j(\xi; \tau, o) \leq -\delta_j,\ \forall j\right] \geq 1 - \epsilon_{\text{phys}} \tag{11}$$

where $\delta_j \geq 0$ is a safety margin accounting for mismatch between the digital twin and the real system. Intuitively, this safety margin means the robot must satisfy the physical constraint with an *additional tolerance* in the digital twin; increasing this tolerance improves sim-to-real robustness. If:

$$\left|g^{\text{real}}_j(\xi; \tau, o) - \hat{g}^{\text{sim}}_j(\xi; \tau, o)\right| \leq \delta_j \tag{12}$$

over the relevant trajectory set, then the added tolerance is sufficient and $\hat{g}^{\text{sim}}_j(\xi; \tau, o) \leq -\delta_j$ implies $g^{\text{real}}_j(\xi; \tau, o) \leq 0$. Put another way, if Equation (12) holds then satisfying the stronger constraint in the digital twin will also satisfy the original feasibility constraint from Equation (8).

B. Video Generation Requirements

Thus far we have framed zero-shot robot manipulation via synthetic video generation as a constrained optimization problem. Under this framework the video generator provides an *initial guess* at the solution, which is then refined by Equation (4). So what properties must the video generator satisfy? To answer this question we will define a basin of attraction, i.e., a set of initial guesses that all converge to the same equilibrium set. More specifically, we focus on *semantic basins of attraction*. These semantic basins provide local regions of the trajectory manifold that when constrained for physical feasibility converge to motions which are aligned with the task-dependent success predicate $\mathcal{G}_\tau$. Below we show how the success of the video generator depends on seeding the optimization in Equation (4) within one of these semantic basins of attraction.

Basin of Attraction. Recall from Equation (3) that ξ^{video}_o is the motion trajectory of objects in the synthetic video. We define z as the features extracted from these trajectories:

$$z = \phi\left(f_{\text{track}}\left(v\right)\right) \tag{13}$$

We next define the basin of attraction around these features. Intuitively, the generated features must be close to both semantic consistency and physical feasibility:

$$\mathcal{S}^{\phi}_{\rho}(\tau, o) = \left\{z : d_\phi\left(z, \mathcal{M}^{\phi}_{\text{sem}}(\tau, o)\right) \leq \rho(\tau, o)\right\} \tag{14}$$

Here $\mathcal{M}^{\phi}_{\text{sem}}$ from Equation (6) is the set of object trajectories that perform the intended task, z from Equation (13) is the generated features, and d_ϕ is the L2 distance in feature space. Equation (14) states that this distance must be less than or equal to the radius $\rho(\tau, o) \geq 0$. The radius characterizes how far the features of a video-derived trajectory may deviate from the semantic feature manifold while still allowing the given physical-constraint enforcement mechanism to preserve its semantic mode. This radius may depend on the task, the initial scene, the robot and environment, and the properties of the enforcement mechanism.

To summarize: we assume that Equation (14) defines a sufficient basin of attraction for the chosen optimization procedure. Under this assumption, synthetic videos which yield object trajectories ξ_o^{video} and features z within $\mathcal{S}^{\phi}_{\rho}(\tau, o)$ converge to trajectories that successfully complete the task.

Video Generation Bound. The probability that the synthetic video lies within the relevant basin of attraction depends on the quality of the video generation model. Because the generated videos v are stochastic, the extracted feature values z can be viewed as a random variable. Let $z_{\text{sem}} \in \mathcal{M}^{\phi}_{\text{sem}}(\tau, o)$ be a semantically valid feature realization. Write

$$z = z_{\text{sem}} + \eta \tag{15}$$

where η denotes the error in the desired feature values extracted from the generated videos. By definition, the expected squared feature error is:

$$\mathbb{E}\left[\|\eta\|_2^2 \mid \tau, o\right] = \sigma^2_{\text{vid}}(\tau, o) \tag{16}$$

For $\rho(\tau, o) > 0$ and finite $\sigma^2_{\text{vid}}(\tau, o)$, Markov's inequality therefore gives a probabilistic condition on video generation:

$$\mathbb{P}\left[z \in \mathcal{S}^{\phi}_{\rho}(\tau, o) \mid \tau, o\right] \geq 1 - \frac{\sigma^2_{\text{vid}}(\tau, o)}{\rho^2(\tau, o)} \tag{17}$$

Put intuitively: the probability of successfully seeding the optimization within a basin of attraction depends on the ratio between (a) the feature error in the synthetic video, σ_{vid}, and (b) the radius within which the physical constraint can be successfully applied, ρ. In the best-case scenario (*reducing* σ_{vid}) the generated video moves the objects to complete the commanded task, and (*increasing* ρ) the object trajectories can be locally modified for physical feasibility without losing their semantic meaning. This analysis does not require unbiased features. Systematic hallucinations from the video model are allowed, but can weaken the bound by increasing σ^2_{vid}.

Task-Selection Implications. The sufficient condition outlined by Equation (17) offers insights into which types of tasks are suitable for our synthetic video pipeline. *The more the object trajectories can be modified without violating their semantic meaning, the better suited for video generation.* Accordingly, pick-and-place tasks are favorable, since the exact motion used to go from the pick to the place can be highly variable (increasing ρ). At the other extreme, contact-rich tasks like pushing an object into a crowded box are unfavorable, since small changes in pose, timing, or grasp can alter the entire task (decreasing ρ).

C. Unifying Related Approaches

In addition to providing guidance on video generation properties, our unified optimization framework also lets us connect recent methods. We can view the state-of-the-art methods [3], [4], [15] as *ad hoc* instances of Equation (4) under simplifying assumptions:

PhysWorld [4]. PhysWorld extracts a single target object trajectory from a synthetic video, and then optimizes the robot's behavior in a digital twin to track this trajectory. This is equivalent to Equation (4) but with a more restrictive cost function:

$$\xi_a^* \in \arg\min_{\xi_a} \left\| \xi_{\text{target}} - \xi_{\text{target}}^{\text{video}} \right\| \tag{18}$$

where ξ_{target} is the trajectory of the target object. Related approaches such as [15] map generated rigid object motions recovered from a single video to end-effector poses, then optimize trajectories with collision and joint limit penalties.

RIGVid [3]. RIGVid directly replays the object trajectory extracted from the synthetic video by assuming a rigid grasp, and using kinematics to map from object motion to robot motion. No physical feasibility is explicitly enforced. This is equivalent to Equation (4) with two changes: (a) the more specific tracking cost from Equation (18) and (b) no physical feasibility constraint $\mathbb{P}\left[\xi \in \mathcal{F}_{\mathcal{R}}\right] \geq 1 - \epsilon_{\text{phys}}$.

We note that these simplifications are not necessarily a negative; they may be reasonable choices for some problem settings. We will explore how these simplifications compare to more closely following Equation (4) in our experiments.

V. Approximating the Underlying Solution

In Section IV we present zero-shot manipulation from synthetic videos as a constrained optimization problem. While current methods introduce simplifications, in this section we present **CASTER** (Figure 2), an approach that more closely follows Equation (4). Practically, this leads to two distinctions from previous work. First, instead of optimizing to exactly follow the object's video trajectory, we preserve *task features* ϕ. Second, we consider multiple sampled synthetic videos when extracting these features, enabling the robot to optimize over *expected behaviors*. We build a digital twin to reconstruct the scene, then perform constrained trajectory optimization in the digital twin while enforcing collision-avoidance and kinematic constraints. The aggregated trajectory features define a shared optimization target and each sampled video initializes a separate optimization run. The resulting robot trajectory can be executed open-loop in the real world.

A. Extracting the Task Features

We start with the cost function in Equation (4). Using the task instruction τ and a single RGB-D observation $o = [\mathbf{I}, \mathbf{d}] \in \mathbb{R}^{H \times W \times 4}$, the robot applies Equation (3) to sample multiple videos v from a pretrained video generator. Each video is then checked by a VLM to validate whether the

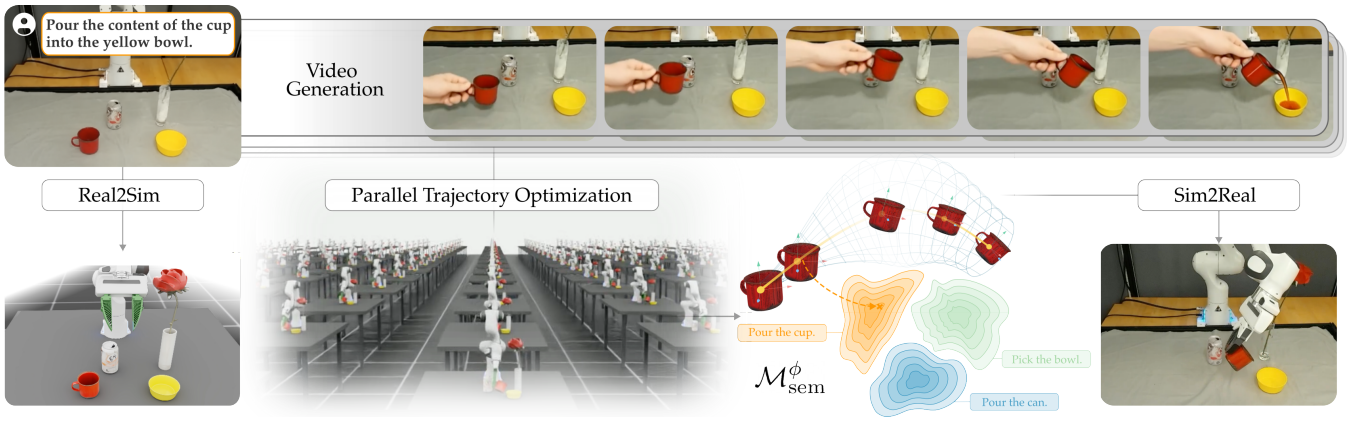

Fig. 2. Overview of **CASTER**. (Top Left) Given an initial RGB-D observation and a task prompt, we (Top) generate synthetic videos of a human performing that task in the current environment and (Bottom Left) build a digital twin of the scene. From the synthetic videos we extract the object trajectories, which then serve as initial guesses for trajectory optimization within the digital twin. (Bottom) The trajectories extracted from synthetic videos do not need to be perfect: instead, they must lie within a semantic basin of attraction where the optimization can locally modify the trajectory shape without removing its semantic meaning. (Bottom Right) We perform constrained optimization so that the robot's actions in the digital twin cause the objects to move similarly to their video trajectories, while enforcing kinematic constraints so that the output can be directly executed by the real robot.

behavior shown in that video performs the requested task. Videos which fail this check are discarded, and only the retained videos are used to extract task features.

Tracking Objects. Referring to Equation (3), we next need to instantiate f_{track} and extract the object trajectories ξ_o^{video} from the synthetic videos. We assume the planar mapping from the RGB image to the robot's coordinate frame is calibrated. We then use an off-the-shelf depth generation model [20] to estimate depth for each frame of the generated video, denoted as $\hat{\mathbf{d}}_{1:T}$. To align these estimates with the observed depth $\mathbf{d}$, we fit an affine transformation to the estimated depth of the first frame o and apply the same transformation to all subsequent frames. To compute the object orientation in the synthetic videos we apply TrackCraft3R [21] — this approach tracks a dense set of 3D points on the object surface, which can then be fit to a rigid body transformation. Put together, this position, depth, and orientation tracking produces ξ_o^{video}.

Extracting Features. In Section IV-A we argue that the robot should not try to *exactly reproduce* every object trajectory. Instead, the robot only needs to match the task-critical parts of the object motions in ξ_o^{video}. To identify these critical features we prompt an LLM with the semantic task τ and the object labels (e.g. the names of each object in the scene). This LLM outputs two things: (a) the *object relations* essential to the task and (b) the *trajectory dimensions* that need to be tracked. Object relations are defined as target-context object pairs $\xi_\tau = \{(\xi_t^1, \xi_c^1), \cdots (\xi_t^K, \xi_c^K)\}$, where $\xi_t, \xi_c \in \xi_o$. For example, when "pouring the coffee" a target could be the coffee mug and the context could be the coffee pot. The trajectory dimensions refer to the specific elements of these motions which should be tracked (e.g., tracking the position specifically in the x and y axes).

The resulting LLM outputs are then applied to each video v that we have sampled for the task. We compute $\bar{\xi}_\tau$, the *median position and mean orientation* for the object relations along dimensions identified by the LLM. We also consider the *variance* among these sampled videos. Specifically, we compute time-dependent *weights* w which decay exponentially with cross-demonstration variance and emphasize consistent features across all the sampled videos. This overall process of using an LLM to identify relevant object relations and then calculating their mean and variance across multiple synthetic videos constitutes the feature extractor ϕ.

B. Constrained Optimization in a Digital Twin

We optimize Equation (4) in a digital twin. Within our experiments we use Isaac Sim [22], and reconstruct objects from the initial observation o using SAM 3D [23]. Optimization is performed over the robot's end-effector trajectory *splines* with control parameters $\xi_a \in \mathbb{R}^{6P}$. Here P is the number of control points (selected by the designer), and each point is parameterized as a six-dimensional pose. We optimize these parameters following the cross-entropy method (CEM) [24], evaluating each candidate through a simulated rollout with the feature-based cost function:

$$\xi_a^* \in \arg\min_{\xi_a} \sum_t w_t \cdot \left\| \phi(\xi_o^t) - \bar{\xi}_\tau^t \right\| \\ \text{subject to: } \|F(\xi_a)\| \leq \bar{F}, \quad q_{\min} \leq q(\xi_a) \leq q_{\max} \tag{19}$$

Vector F is the force applied by the simulated robot, and $\bar{F}$ is the maximum force threshold when interacting with task-irrelevant objects. The constraints prevent unintended collisions that exceed $\bar{F}$, as well as joint values q that exceed the robot's kinematic limits. Comparing our approximation in Equation (19) to the idealized optimization in Equation (4), we notice one key distinction. Equation (19) does not directly include the semantic consistency constraint: this is because we pre-process the synthetic videos used to compute the reference motions $\bar{\xi}_\tau$, and ensure these videos perform the intended task. Based on our analysis from Section IV-B, if these videos seed the optimization within a semantic basin of attraction their task consistency will be preserved.

VI. Experiments

We conduct real-world experiments to evaluate the effectiveness of our unified framework. Within these experiments

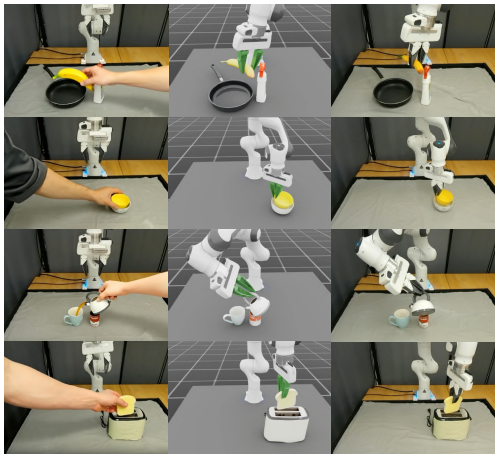

Fig. 3. Examples of our real-world tasks. (Left) Generated videos. (Middle) Digital twin rollouts. (Right) Real-world execution. The rows show four tasks: placing a banana in a pan, stacking bowls, pouring from a kettle, and toasting bread. See our supplemental video for more examples.

the robot is expected to perform the 12 tasks from Table I on its first attempt, without any real-world demonstrations or additional interactions. To find the correct trajectory the robot uses synthetically generated videos of the human performing the task. Examples of these videos are shown in Figure 3. We chose these tasks because they require compliance with semantic instructions and multiple constraints on position, orientation, and collision avoidance.

Baselines. We compare **CASTER** with two baselines representing common strategies for transferring generated motion, as exemplified by two prior works from Section IV-C:

1) **Retarget***:* Similar to RIGVid [3], this baseline directly retargets the object trajectory from the synthetic video to the robot's end effector using a rigid-body transformation.

2) **Mimic***:* Similar to PhysWorld [4], this baseline optimizes the robot's trajectory in a digital twin so that the resulting object motions precisely mimic the synthetic videos while maintaining physical feasibility.

Setup and Video Generation. Our real-world setup consists of a Franka Emika robot arm and a stationary Orbbec Femto Mega camera. The digital twin is constructed on the fly from an initial RGB-D image of the scene using SAM 3D [23] and Isaac Sim [22]. Synthetic videos are generated using HunyuanVideo-1.5 480P-I2V Step-Distilled [25]. We generated 20 videos per task using random seeds 0 to 19; all methods were evaluated on the same set of videos. Each video is conditioned on the initial scene image and the task-specific text prompts. The videos contained 121 frames at 24 fps and 480p resolution. Prompt rewriting and super-resolution were disabled.

Metrics and Trajectory Sampling. Task success was defined as achieving the goal specified in the prompt while avoiding collisions with irrelevant objects. The task failed if the robot tried to violate its joint limits.

After generating the 20 synthetic videos as described above, we next used Qwen3.5 [26] as a validator to score the semantic consistency of each video (i.e., did the synthetic video appear to complete the task). Videos with a score of ≥ 0.75 were retained. All the retained videos were used for extracting task features. We then randomly sampled a set of 7 valid videos for real-world evaluation. Each of the 7 sampled videos initializes a separate CEM run. **Mimic** tracks the corresponding video's object trajectory, whereas **CASTER** uses the shared feature reference $\bar{\xi}_\tau$ and weights w_t from all retained videos. For both methods, we discard candidates from each run's final CEM population that violate collision or joint-limit constraints, then randomly select 3 trajectories from the lowest-cost quartile of feasible candidates. Hence, each task comprises seven evaluation trials for **Retarget** and 21 for **Mimic** and **CASTER**. Trials that cannot be executed because of insufficient valid videos or eligible feasible trajectories are counted as failures. For **Mimic** and **CASTER**, we report the mean and standard error across the success rates of 7 optimization runs.

TABLE I

ZERO-SHOT SUCCESS (%) WHEN PERFORMING REAL-WORLD MANIPULATION TASKS. MIMIC AND **CASTER** REPORT MEAN ± STANDARD ERROR ACROSS SEVEN VIDEO INITIALIZED CEM RUNS PER TASK; RETARGET REPORTS SUCCESS RATE ONLY.

Task	**Retarget**	**Mimic**	**CASTER**
1. Move banana in pan	0.0	66.7 ± 14.5	100.0 ± 0.0
2. Pour cup	42.9	52.4 ± 16.0	95.2 ± 4.8
3. Place block in basket	28.6	76.2 ± 14.0	100.0 ± 0.0
4. Pour kettle	57.1	66.7 ± 12.6	61.9 ± 8.7
5. Plate orange	57.1	61.9 ± 15.3	100.0 ± 0.0
6. Pile up blocks	0.0	9.5 ± 6.1	57.1 ± 20.2
7. Stack bowl	14.3	28.6 ± 15.3	100.0 ± 0.0
8. Cover bucket	42.9	28.6 ± 11.3	33.3 ± 7.3
9. Close lid	14.3	4.8 ± 4.8	57.1 ± 14.0
10. Pick w/o collision	85.7	90.5 ± 6.1	95.2 ± 4.8
11. Toast bread	28.6	14.3 ± 9.9	90.5 ± 6.1
12. Draw on whiteboard	14.3	9.5 ± 9.5	28.6 ± 18.4
Total	**32.1**	$\mathbf{42.5 \pm 4.5}$	$\mathbf{76.6 \pm 3.9}$

Task Success Results. Our results are summarized in Table I. Across the 12 tasks, **CASTER** achieved a higher average zero-shot success rate than the baselines. When comparing **CASTER** and **Retarget**, we see the advantage of using a digital twin to enforce the physical feasibility constraint. Since **Retarget** directly maps the video motion to the robot's trajectory, any infeasible behaviors in the video will carry over to the real world. The resulting **Retarget** motion frequently collides with surrounding objects. When comparing **CASTER** and **Mimic**, we observe the effect of our feature based, multi-video objective function. In **Mimic** the robot is trying to match each waypoint and frequently fails to find a solution that satisfies the physical constraints. In contrast, our features only encourage the robot to match waypoints that are consistent across multiple synthetic videos, granting the optimizer greater flexibility.

A major source of failure was the discrepancy between simulated and real-world physics, as illustrated by the "pour

TABLE II

EVALUATING **CASTER** WITH DIFFERENT VIDEO GENERATION MODELS. VALID REPORTS THE NUMBER OF TASK-ALIGNED VIDEOS OUT OF 80 SAMPLED VIDEOS ACROSS FOUR TASKS. SUCCESS REPORTS MEAN TASK SUCCESS (%) ACROSS SEVEN SAMPLED VIDEOS PER TASK.

Model	Valid	Success
HunyuanVideo-1.5 [25]	55/80	79.8 ± 6.5
Wan 2.2 I2V-A14B [27]	71/80	76.2 ± 6.1
LTX-2.3 22B Distilled [28]	52/80	47.6 ± 8.6
CogVideoX1.5 5B-I2V [29]	1/80	0.0 ± 0.0

kettle" and "cover bucket" tasks. In the former, squeezing the kettle handle causes the kettle to rotate within the gripper, a behavior not captured in simulation. In the latter, the bucket's handle has rotational joints that are not represented by our rigid body mesh reconstruction. These digital twin discrepancies contribute to the limited performance of both our method and the baselines on these tasks.

Video Generator Comparison. Thus far we have compared our unified approach to state-of-the-art baselines. But it is possible that these results depend on the specific video generation model used to seed the search. Accordingly, we here compare four open-source image-to-video models, and see how they impact our method's performance. The selected models include [25], [27], [28], and [29]. All models receive the same text prompts and initial scene images. We use model-specific inference settings and standardize videos to 121 frames at 24 fps before downstream processing.

Our averaged results across tasks 1, 4, 6, and 7 are shown in Table II. We report the number of synthesized videos that pass the VLM check out of 80 videos generated across four tasks, 20 videos per task. Overall, we find that validity count alone does not explain downstream success. Wan 2.2 [27] generated more valid videos than HunyuanVideo-1.5 [25], but achieved similar success, while LTX-2.3 [28] produced a similar validity count yet substantially lower success. We hypothesize that additional valid videos may depict redundant motion strategies that provide limited new guidance. Section IV-B also suggests that performance may instead depend on task-relevant feature error relative to the optimizer's semantic basin, whose extent depends on task difficulty and robot constraints.

VII. CONCLUSION

In this paper we leverage video generation models at inference time for zero-shot manipulation. By unifying recent works, we introduce a mathematical formalism for transferring the motions in generated videos to real robot trajectories. We pose the setting as a constrained optimization problem. Using a digital twin, the robot attempts to mimic task-critical aspects of the object trajectories while enforcing semantic and physical constraints. This unified formalism enables (a) establishing a conditional relationship between video feature error and successful initialization, (b) casting state-of-the-art approaches as instances of our optimization framework, and (c) removing simplifying assumptions from these approaches to more closely mirror the formalism. Across 12 real-world tasks, **CASTER** achieves 76.6% average zero-shot task success, exceeding the best-performing baseline by 34.1%. Future work could train closed-loop policies on **CASTER** generated trajectories.

REFERENCES

[1] A. Khazatsky *et al.*, "DROID: A large-scale in-the-wild robot manipulation dataset," arXiv:2403.12945v2, 2025.

[2] A. Iyer *et al.*, "OPEN TEACH: A versatile teleoperation system for robotic manipulation," arXiv:2403.07870, 2024.

[3] S. Patel *et al.*, "Robotic manipulation by imitating generated videos without physical demonstrations," arXiv:2507.00990, 2025.

[4] J. Mao *et al.*, "Robot learning from a physical world model," arXiv:2511.07416, 2025.

[5] Y. Du *et al.*, "Learning universal policies via text-guided video generation," arXiv:2302.00111, 2023.

[6] J. Liang *et al.*, "Video generators are robot policies," arXiv:2508.00795, 2025.

[7] X. Lang *et al.*, "VAG: Dual-stream video-action generation for embodied data synthesis," arXiv:2604.09330, 2026.

[8] J. Jang *et al.*, "DreamGen: Unlocking generalization in robot learning through video world models," arXiv:2505.12705v2, 2025.

[9] Y. Guo *et al.*, "Ctrl-World: A controllable generative world model for robot manipulation," arXiv:2510.10125v3, 2026.

[10] B. Qiu *et al.*, "GE-Sim 2.0: A roadmap towards comprehensive closed-loop video world simulators for robotic manipulation," arXiv:2605.27491, 2026.

[11] Z. Bi *et al.*, "GenHOI: Contact-aware humanoid-object interaction by imitating generated videos without task-specific training," arXiv:2606.12995, 2026.

[12] H. Bharadhwaj *et al.*, "Gen2Act: Human video generation in novel scenarios enables generalizable robot manipulation," arXiv:2409.16283, 2024.

[13] B. Paliwal *et al.*, "Do as I do: Dexterous manipulation data from everyday human videos," arXiv:2606.19333, 2026.

[14] J. Liang *et al.*, "Dreamitate: Real-world visuomotor policy learning via video generation," arXiv:2406.16862, 2024.

[15] H. Li *et al.*, "NovaFlow: Zero-shot manipulation via actionable flow from generated videos," arXiv:2510.08568, 2025.

[16] J. Fu *et al.*, "NovaPlan: Zero-shot long-horizon manipulation via closed-loop video language planning," arXiv:2602.20119, 2026.

[17] S. He *et al.*, "RoboReact: Agentic skill distillation from generated egocentric videos for generalizable whole-body manipulation," arXiv:2608.03387, 2026.

[18] J. Ni *et al.*, "From generated human videos to physically plausible robot trajectories," arXiv:2512.05094, 2025.

[19] K. Dharmarajan *et al.*, "Dream2Flow: Bridging video generation and open-world manipulation with 3D object flow," arXiv:2512.24766, 2025.

[20] H. Lin *et al.*, "Depth Anything 3: Recovering the visual space from any views," arXiv:2511.10647, 2025.

[21] J. Nam *et al.*, "TrackCraft3R: Repurposing video diffusion transformers for dense 3D tracking," arXiv:2605.12587, 2026.

[22] NVIDIA, "Isaac Sim," 2026, version 6.0.1. [Online]. Available: https://github.com/isaac-sim/IsaacSim

[23] X. Chen *et al.*, "SAM 3D: 3Dfy anything in images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2026, pp. 7220–7232.

[24] R. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Methodol. Comput. Appl. Probab.*, pp. 127–190, 1999.

[25] Tencent Hunyuan Foundation Model Team, "HunyuanVideo 1.5 technical report," arXiv:2511.18870, 2025.

[26] Qwen Team, "Qwen3.5: Towards native multimodal agents," February 2026. [Online]. Available: https://qwen.ai/blog?id=qwen3.5

[27] Team Wan *et al.*, "Wan: Open and advanced large-scale video generative models," arXiv:2503.20314, 2025.

[28] Y. HaCohen *et al.*, "LTX-2: Efficient joint audio-visual foundation model," arXiv:2601.03233, 2026.

[29] Z. Yang *et al.*, "CogVideoX: Text-to-video diffusion models with an expert transformer," arXiv:2408.06072, 2024.