

See, Imagine, Plan: Discovering and Hallucinating Tasks from a Single Image

Chenyang Ma^(✉), Kai Lu, Ta-Ying Cheng, Niki Trigoni, and Andrew Markham

Department of Computer Science, University of Oxford, Oxford, UK
chenyang.ma@cs.ox.ac.uk

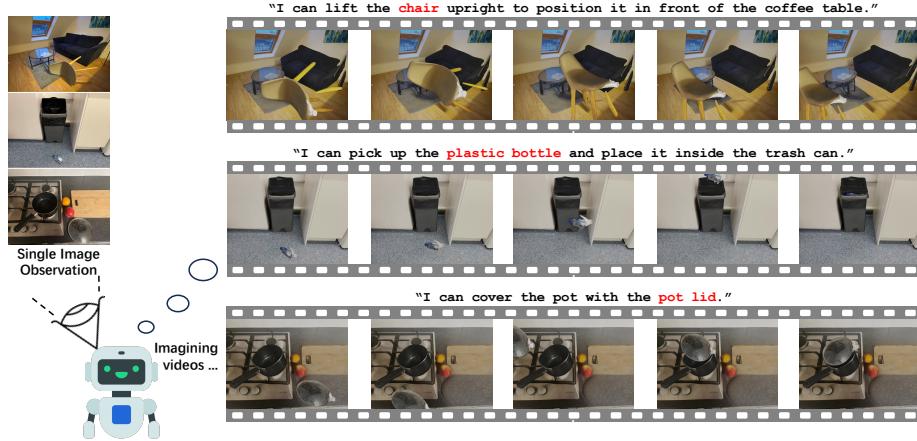


Fig. 1: We present a model for *zero-shot task hallucination*. With a single RGB image of any scene comprising unknown environments and objects, we identify potential tasks (task discovery) and imagine their execution (manipulation) in a vivid narrative, realized as a video. We also provide the trajectories of the executed tasks.

Abstract. Humans can not only recognize and understand the world in its current state but also envision future scenarios that extend beyond immediate perception. To resemble this profound human capacity, we introduce zero-shot task hallucination—given a single RGB image of any scene comprising unknown environments and objects, our model can identify potential tasks and imagine their execution in a vivid narrative, realized as a video. We develop a modular pipeline that progressively enhances scene decomposition, comprehension, and reconstruction, incorporating VLM for dynamic interaction and 3D motion planning for object trajectories. Our model can discover diverse tasks, with the generated task videos demonstrating realistic and compelling visual outcomes that are understandable by both machines and humans. All code, data, and additional videos are available at our project page: https://dannymcy.github.io/zeroshot_task_hallucination/

“A rock pile ceases to be a rock pile the moment a single man contemplates it, bearing within him the image of a cathedral.”
—Antoine de Saint-Exupéry

1 Introduction

The theory of mental time travel [59] highlights the human ability to recall past knowledge and anticipate future scenarios through visual perceptions. For example, the sight of a fallen chair immediately prompts in our mind an imagined rehearsal to straighten it back up. Equipping intelligent agents with this level of imaginative foresight and the ability to visualize such imagination could open up exciting possibilities, such as serving interactive tutors, offering step-by-step visual guidance on tasks, or teaching robots to autonomously discover and undertake tasks in new environments.

In this work, we aim to resemble this profound human capability to see, imagine, and plan, by utilizing the recent advancements of large foundation models [7, 47]. We introduce the problem of ***zero-shot task hallucination***. Given a single RGB image of any scene comprising unknown environments and objects, our objective is to identify potential tasks (task discovery) and imagine their execution (manipulation) in a vivid narrative, realized as a video, as shown in Fig 1. The proposed tasks within the 3D scenes should be diverse yet reasonable. Moreover, the generated task videos should be geometric-aware and photo-realistic. The object motion should be feasible, fine-grained, and consistent to ensure the task execution is interpretable by both humans and machines.

The challenge of zero-shot task hallucination is threefold. First, the model must encompass strong capabilities of scene understanding. This includes not only the ability to recognize arbitrary objects from diverse scene settings, but also the understanding of interrelations between objects to propose reasonable tasks. Second, the model must have a 3D understanding of the scene so that the planned task execution follows the given spatial constraints. Finally, this hallucinated task execution should be presented by a human-interpretable fashion (*e.g.*, a video). The closest existing approaches to addressing these challenges lie in video generation and 3D-aware image editing fields. Yet, no current solution seamlessly combines high-quality task execution visualization with geometric awareness. State-of-the-art (SOTA) video generation models [32, 52] are confined to 2D pixel space. They show satisfactory performance only in short sequences and struggle with the precision needed for detailed, prolonged manipulation tasks. Leading 3D-aware image editing models [18, 48] fail to maintain smooth transitions and consistency across interpolated frames. Moreover, both approaches lack the ability to understand 3D scenes and the capability to process detailed, geometry-aware prompts (*e.g.*, covering the pot with the pot lid), which are needed for manipulation tasks.

Our solution addresses these challenges with a modular pipeline that progressively enhances scene decomposition, comprehension, and reconstruction, incorporating Vision-Language Model (VLM) for dynamic interaction and 3D motion planning for object trajectories, producing geometric-aware task videos. To understand the image scene, we use VLM to identify interactive objects and propose context-dependent tasks in a role-play manner, complemented by language-guided segmentation and repainting models to obtain occlusion-free object masks. Elevating 2D understanding to 3D, we use depth estimation and

single-view 3D reconstruction models to generate a semi-reconstructed 3D scene, with the full 3D representation of foreground objects and the background as a plane. With the reconstructed 3D scene, we introduce a novel axes-constrained 3D planning approach that enables VLM to plan the motion of objects for given tasks by specifying waypoints.

Through the combination of traditional path planning algorithms, our model generates complete, feasible, and natural trajectories from merely a single image observation. With the entire framework being fully modularized, each component can be easily replaced with the latest improvements within its specific domain.

In summary, our main contributions are threefold:

- We study zero-shot task hallucination—the capability for models to discover and propose possible tasks and plans of execution given a single image.
- We devise a plug-and-play framework that leverages large pretrained VLM and 3D reconstruction model, combining with traditional path planning algorithms to provide geometric-aware trajectories for diverse tasks.
- We show our model can convert these task plans into human-interpretable formats such as videos for various potential applications, supported by extensive experiments.

2 Related Work

3D-Aware Image Editing The introduction of diffusion models [24, 56] has created a plethora of high-performing aesthetic text-to-image generative models [3, 24, 50, 53, 54]. Various editing methods soon followed to bring more controls over the images generated. [6, 23, 49] enables edits to real-world images with additional words, while [19] allows fine-grained controls over the pixel space.

Several works also explored precise edits over the 3D space (*e.g.*, object rotation, translation, illumination, non-rigid shape changes) [12, 44, 48, 57]. Zero-1-to-3 [40], ViewNETI [8], and Continuous 3D Words [12] enable rotations of an object in the image, but are generally single-object centric. 3DIT uses large-scale 3D datasets [18] to simulate and learn scene arrangements, while Diffusion Handles [48] elevates the image editing space to 3D via depth estimation.

Inspired by these works, and in light of the complicated tasks that could be discovered in our pipeline, and the recent advancements of many generalizable single-image object reconstruction techniques [2, 38, 39], we propose to explicitly reconstruct part of the scene to perform edits and rerender back to 2D.

Task Discovery Previous research on robotic agent targeted predefined tasks [27, 42] or environments [20, 61]. Very recently, a few works [1, 66] proposed the concept of task discovery/proposal. RoboGen [66] presented the idea of generative simulation, in which they leveraged foundation models to generate robotic learning data by configuring new scenes and proposing robotic tasks in physics engines. AutoRT [1] observed and proposed tasks with a large fleet of robots. Nevertheless, both of these works made the underlying assumption that the scenes either created or observed are suitable for operation by certain robot types.

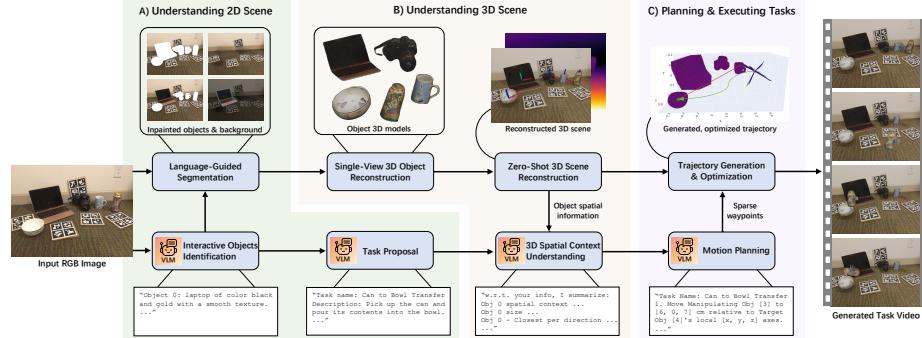


Fig. 2: The proposed framework. Our plug-and-play framework is fully modularized and zero-shot. Each module can be easily replaced with the latest improvements. Exact prompts for VLM are presented in Appendix. E.

Our work diverges from these constraints and is applicable to all real scenes. Moreover, our method requires only a single RGB image as observation/input.

Spatial Reasoning and Planning with VLMs With the internet-scale training data and the birth of LLMs [7, 47], VLMs are encompassed with reasoning and scene understanding capabilities [11, 31, 37]. Several works build from the foundation and establish datasets to help with spatial reasoning/understanding [28, 36, 44]. As a concurrent work, Spatial VLM [10] enhanced standard VLMs with the ability to reason and answer questions requiring spatial understanding. In this work, we take one step further—asking VLMs to not only reason about the spatial information, but also plan the trajectory for tasks/actions in 3D space.

3 Method

Given a single RGB image $I \in \mathbb{R}^{H \times W \times 3}$ of a scene with K unknown objects, our goal is to discover Q tasks suitable for execution by a robotic/human hand, and generate Q task videos, each visually depicting the execution of these tasks.

Our fully modularized framework progressively enhances scene decomposition, comprehension, and reconstruction, incorporating VLM for dynamic interaction and 3D motion planning for object trajectories, producing geometric-aware task videos (Fig. 2). It begins by understanding 2D image scene (Sec. 3.1). VLM identifies interactive objects and proposes context-dependent tasks through role-play, complemented by a two-step segmentation to acquire occlusion-free object masks with language-guided segmentation and repainting models. We then reconstruct the 3D scene in a semi-reconstructed fashion leveraging single-view 3D reconstruction and depth estimation models (Sec. 3.2). Our method for camera pose and object poses and sizes estimation does not require camera intrinsic matrix and uses only normalized depth, thus can be generalized to any input image. With the reconstructed 3D scene, we conduct in-context learning to let VLM understand the 3D spatial context. Subsequently, we introduce a novel axes-constrained 3D planning approach that enables VLM to plan the object

motion based on the proposed tasks by specifying waypoints (Sec. 3.3). To generate complete, feasible, and natural trajectories, we use traditional robotics path planning and optimization methods. Finally, we execute the tasks within our reconstructed 3D scene following the planned paths, and render as videos.

3.1 Understanding Objects and Context of 2D Image Scene

3.1.1 Interactive Objects Identification

We start with querying VLM for intractable objects identification given I . We explicitly ask VLM to identify and name useful objects in term of affordances. Note that we exclude background objects (*e.g.*, window, ceiling, stair), as they tend to be fixed and typically offer limited interaction opportunities. We also prompt VLM to provide concise descriptions, capturing the distinct color and texture of each identified object. This step is vital for differentiating between items of similar categories and appearances.

As a concrete example, given the left image of Fig. 2, VLM outputs:

```
"object 0: laptop of color black and gold with a smooth texture.  
object 1: camera of color black with a textured grip. ..."
```

3.1.2 Occlusion-Free Object 2D Masks and Background Inpainting

Occlusion-Free Object 2D Masks The concise descriptions of identified objects are used as input text prompts for a language-guided segmentation model, enabling the acquisition of K segmentation masks $\{M_k^{occ}\}_{k=1}^K$, with each mask corresponding to a unique object.

However, an object $i \in [1, K]$ may be occluded by other object(s), leading to an incomplete mask M_i^{occ} . To resolve this, we create an inpainting mask, M_i^{inp} , for each object, in which all objects except the one itself are removed and replaced with white pixels. The inpainted masks are again fed to the language-guided segmentation model along with input text prompts such that occlusion-free object masks, $\{M_k^{of}\}_{k=1}^K$, are obtained. This two-step segmentation process for object i can be formulated as:

$$M_i^{occ} = \text{seg}(I, \tau_i), \quad M_i^{of} = \text{seg}(\text{inpaint}(M_i^{inp}), \tau_i), \quad (1)$$

where seg denotes the language-guided segmentation and τ_i denotes the description of object i . In practice, to cleanly remove objects without residual fragments for inpainting, we apply dilation to and expand the white areas.

Background Inpainting We also remove and replace all objects with dilated white pixels, and inpaint to acquire the inpainted background I_{bg} .

3.1.3 Task Proposal

We query VLM to propose meaningful, diverse tasks, each with a one-sentence task description. Instead of directly querying VLM for task proposal, we employ a hybrid approach that integrates role-play and object-based initialization. In



Fig. 3: Our method of **partial 3D scene reconstruction**. (a) full representation of foreground objects and background as a plane, (b) the reconstructed scene from the active camera view, (c) the original input image.

the role-play scenario, we prompt VLM to envision itself as a robotic/human hand working in the scene to perform household tasks. For the object-based initialization, we guide VLM to sequentially focus on each identified object within the scene. When the scene contains more than one identified objects, VLM is instructed to suggest two tasks emphasizing interactions between the manipulating object and any of the detected objects, and an additional task focused solely on the manipulating object. If only one object is detected, VLM is directed to propose a task involving just that object. This strategy guarantees a broad spectrum of task suggestions, ensuring comprehensive object engagement.

To further tailor the task proposals, we impose specific constraints, directing VLM to consider the practical affordances of objects while encouraging creative assumptions (*e.g.*, a bowl’s capacity to hold water) and potential interactions (*e.g.*, transferring water from a cup into a bowl). Additionally, we delineate clear boundaries by excluding tasks that entail the assembly or disassembly of objects, functionality tests, or the involvement of imaginary objects, thereby focusing on feasible and meaningful tasks.

As a concrete example, given the image on the left of Fig. 2, with the manipulating object to be the red can, VLM will propose the following tasks:

```
"Task name: Can to Bowl Transfer
Description: Pick up the can and pour its contents into the bowl.
Task name: Can Relocation
Description: Pick up the can and place it inside the bowl.
Task name: Can Rotation
Description: Rotate the can 90 degrees on its vertical axis. ..."
```

3.2 Reconstructing and Understanding 3D Image Scene

Our method for partial reconstruction of the 3D scene focuses on full representation of foreground objects while simplifying the inpainted background as a plane. The reconstructed scene is captured by a pinhole camera in the 3D environment, as shown in Fig. 3(a). Given that our framework is designed to accommodate input images captured by users using arbitrary cameras, we make the assumption that the camera intrinsic matrix is unknown. Our partially reconstructed scene demonstrates high visual alignment with the input image, see Fig. 3(b) and (c).

3.2.1 Single-View 3D Object Reconstruction and Depth Estimation

The first step of reconstructing the 3D image scene involves reconstructing 3D object models and estimating depth. We feed occlusion-free object masks, $\{M_k^{of}\}_{k=1}^K$, to an off-the-shelf single-view 3D reconstruction model to acquire object 3D models, $\{O_k\}_{k=1}^K$. We also feed the original map I to an off-the-shelf depth estimation model to acquire depth image, denoted as I_{dep} .

3.2.2 Camera 6D Pose Estimation and Object Scale Initialization

Camera 6D Pose Estimation Single-view 3D reconstruction model reconstructs mesh O_i of object i at the 3D origin, in the coordinate frame set by the input mask M_i^{of} , and captures O_i by a pinhole camera. This camera, with 6D pose $P_c = [R_c \mid t_c^{origin}]$, captures O_i 's canonical pose within the image. Thus, we can restore all objects' canonical poses across all images by identifying P_c .

We use One-2-3-45++ [38], which provides the camera pose. For models without this information, we develop an efficient method to determine the pose by comparing object masks with rendered 3D model templates, inspired from matching-based 6D pose estimation works [9, 34, 41, 46]. Details in Appendix. B.

For pinhole camera to focus on the background plane center within the reconstructed 3D scene, we backproject the 2D center of the inpainted background I_{bg} using a camera intrinsic matrix C_{Mat} , and adjust the translation vector t_c^{origin} in the camera pose P_c . C_{Mat} is arbitrary, with the only condition of principal point coordinates, c_x and c_y , set to image I center. Focal lengths f_x and f_y are set to $\epsilon \in \mathbb{R} \setminus \{0\}$ because backprojecting the image center results in $u = c_x$ and $v = c_y$. Thus, our camera intrinsic matrix and adjusted, final camera pose are:

$$C_{Mat} = \begin{bmatrix} \epsilon & 0 & W/2 \\ 0 & \epsilon & H/2 \\ 0 & 0 & 1 \end{bmatrix}, \quad P_c = [R_c \mid t_c], \quad t_c = t_c^{origin} + \text{backproject}(I_{dep}, \text{center}(I), C_{Mat}). \quad (2)$$

Setting Background Plane The 6D pose of the background plane O_{bg} (visually identical to I_{bg}), denoted as $P_{bg} = [R_{bg} \mid t_{bg}]$, matches the camera's rotation R_c as its orientation aligns with the camera. The position t_{bg} is shifted along the camera's directional vector ($v_c = -R_c \times [0, 0, 1]$), ensuring fitting precisely within the camera's view based on the camera's focal length and sensor width.

Object Scale Initialization 3D reconstruction models typically standardize objects within a normalized sphere [13, 38, 39, 45, 71]. To set the initial scale for O_i , we multiply the contour length of its occlusion-free 2D mask M_i^{of} by a predefined scale factor δ_{init} , tailored to model's normalized space dimensions:

$$S_i^{init} = \text{arcLength}(\text{findContour}(M_i^{of})) \times \delta_{init}. \quad (3)$$

However, this method overlooks objects' depth, inaccurately assuming uniform distance from the camera (*e.g.*, moving an apple from close to the camera to a distant corner reduces its contour length). To correct this, we refine the object scales after determining their 6D poses, detailed in the subsequent Sec. 3.2.3.

3.2.3 Object 6D Pose Estimation and Object Scale Calibration

Object 6D Pose Estimation Our goal is to position object 3D models into the reconstructed 3D scene without visual discrepancies and ensure accurate depth. Because a significant portion of depth estimation model [4, 5, 51, 69] is trained on depth datasets with depth data determined by sensors [21, 55] and stereo matching [15, 63, 64, 67, 70], we assume that the predicted normalized depth is the perpendicular distance to the camera plane, instead of a straight line from the object to the camera lens [30]. We start by determining the 2D centers of each occlusion-free object mask, $\{M_k^{of}\}_{k=1}^K$, using center of mass. Then, we calculate the corresponding object 3D centers on the background plane O_{bg} , denoted as $\{t_k^{bg}\}_{k=1}^K$. This process involves initially mapping the 2D centers onto O_{bg} upon its importation into the scene and then transforming these points by multiplying the background plane's pose P_{bg} . Similarly, denote the camera plane as O_c with pose $P_c = [R_c \mid t_c]$ and each object 3D center on O_c as $\{t_k^c\}_{k=1}^K$. For an object $i \in [1, K]$, we perform raycasting from t_i^c on the camera plane to t_i^{bg} on the background plane, and its distance d_i to move from t_i^c towards t_i^{bg} is written as:

$$d_i = \left| k_{init} \times I_{dep}(\text{center}(M_i^{of})) \right|. \quad (4)$$

Here, k_{init} represents the depth scaling factor that adjusts the normalized depth values to the appropriate scale. Because the pinhole camera is used to capture the 3D scene, we then raycast from the camera lens t_c to t_i^{bg} with distance calculated based on d_i such that the perpendicular distance moved from the camera plane towards t_i^{bg} remains unchanged. The 3D coordinate of object i is:

$$t_i = t_c + \frac{d_i}{|t_i^{bg} - t_i^c|} \times (t_i^{bg} - t_c). \quad (5)$$

The rotation matrix R_i of the 6D pose of object i , $P_i = [R_i \mid t_i]$, is set to identity matrix because the reconstructed 3D model O_i is aligned in its canonical pose.

Object Scale Calibration After integrating all 3D object models $\{O_k\}_{k=1}^K$ into the 3D scene, each with a pose $\{P_k = [R_k \mid t_k]\}_{k=1}^K$ and an initial scale $\{S_k^{init}\}_{k=1}^K$, we refine their scales to accurately reflect depth variations. Through the lens of the pinhole camera with pose P_c , we render binary masks for each object and evaluate the length of their contour lines relative to their occlusion-free masks. The adjusted, final scale of object i can be expressed as:

$$S_i^{adj} = S_i^{init} \times \frac{\text{arcLength}(\text{findContour}(M_i^{of}))}{\text{arcLength}(\text{findContour}(M_i^{rend}))}, \quad (6)$$

where M_i^{rend} is the rendered mask of object i .

Finding Object Principle Axes We determine the principal axes (x-axis, y-axis, and z-axis) of each object, which is essential for our novel 3D motion planning method, explained in Sec. 3.3.1. Using the minimal oriented bounding

box (OBB), we encapsulate each object in the smallest possible box, aligning the object’s principal axes with the box’s dimensions. OBB employs principal component analysis on the object’s point cloud to identify directions of maximum, second greatest, and least variance, which naturally correspond to the object’s orientation along its length, width, and height. This approach is straightforward and effective, especially since our input images primarily consist of rigid objects.

3.2.4 Image and Spatial Context Understanding

With accurate camera pose, background plane pose, and object poses and scales, we can reconstruct the 3D scene, denoted as V_0 , from the input image I . Before using VLM for motion planning within the 3D scene based on proposed tasks, we conduct in-context learning to familiarize VLM with the reconstructed 3D environment. We feed V_0 with visualized object axes (see Fig. 2B). Visualizing 3D spatial information is pivotal in improving VLMs’ understanding of 3D spatial contexts derived from 2D images, validated by 3DAxesPrompts [35]. To augment VLM’s understanding, we also feed a paragraph describing the objects’ poses, sizes, and principal axes in physical units, alongside their spatial relationships. This information is derived entirely from our reconstructed 3D scene. As a concrete example for the left image on Fig. 2:

```
"Obj 0 spatial context:
- 3D center: [-21.0, 101.0, 4.0] cm
- Local x-axis (towards 'right'): [0.8637, 0.293, -0.41]
- Local y-axis (towards 'back_left'): [-0.5943, 0.7879, 0.1614]
- Local z-axis (towards 'front'): [-0.1606, -0.7822, 0.6019]
Obj 0 size: 29.55 cm x 26.91 cm x 13.19 cm (WxDxH)
Obj 0 - Closest per direction: right: Obj 1 (29.16 cm); down: Obj 4 (15.72 cm) ..."
```

3.3 Planning and Task Execution within Reconstructed 3D Scene

3.3.1 Axes-Constrained Motion Planning through Waypoints

We introduce a novel method to guide VLM to conduct motion planning within a 3D scene based on a proposed task by planning motion waypoints along the manipulating object’s principal axes. More specifically, we define four types of manipulations that VLM can use:

Rotation type 1: Axial rotation. The object rotates around its principle axes.

Rotation type 2: Rotation relative to the target object.

- pitch: Tilt similar to pouring water, around a horizontal axis formed by the cross product of the connecting directional vector and the target’s vertical axis.
- yaw: Horizontal rotation, like a camera panning, around a vertical axis formed by the cross product of the connecting directional vector and the pitch axis.
- roll: Rotation like drilling a surface, around the connecting directional vector.

Translation type 1: Defines the goal relative to the target object’s principle axes, with translation values for its $[x, y, z]$ axes in centimeters. $[0, 0, 0]$ cm indicates the goal is the center of the target object.

Translation type 2: Sets the goal relative to a directional vector between two reference objects, specifying how far (in cm) object 1 should move towards or away from object 2 along this vector.

Since VLM inherently lacks the capability to provide 3D coordinates and low-level actions directly [17, 65], our method offers a practical workaround by translating natural language instructions into precise motion waypoints. This approach significantly enhances VLM’s utility in spatial reasoning and manipulation tasks without requiring direct 3D coordinate generation capabilities. Also, the four types of manipulations we defined are both simple and comprehensive, covering a broad spectrum of manipulation tasks.

As a concrete example, given the image on the left of Fig. 2, with "Task name: Can to Bowl Transfer", VLM will plan as follows:

```
"Task Name: Can to Bowl Transfer
Manipulating obj idx: 3
Interacting obj idx: 4
1. Move Manipulating Obj [3] to [6, 0, 7] cm relative to Target Obj [4]'s local [x, y, z] axes.
2. rotate_wref: Rotate Manipulating Obj [3] relative to Target Obj [4] around [pitch] axis by [75]
degrees."
```

3.3.2 Trajectory Generation and Optimization

Trajectory Generation With the waypoints planned by VLM, we generate the manipulating object’s trajectory using path planning algorithm, specifically rapidly-exploring random tree star (RRT*) [29]. To generate accurate collision-free path, we perform K-means clustering on the point clouds of object 3D model with a high number of clusters, segmenting the object mesh into discrete voxels and treating each voxel as an obstacle. Then, to accurately consider the manipulating object’s dimensions, we grow the size of each voxel by its dimensions.

Handling VLM Planning Discrepancies The waypoints generated by VLM are typically accurate and practical. Nonetheless, there are instances where the waypoints suggested by VLM lead to collisions as determined by the RRT* planner. This discrepancy is less about VLM’s misunderstanding of the objects’ sizes and their spatial relationship and more about the precision level of the waypoints, which may not match the exacting standards of the RRT* planner’s outcomes. To resolve this, we implement Gaussian sampling around the initially planned waypoints whenever a collision is detected. The sampling strategy is guided by a predefined set of geometric rules. Please see our Appendix. B for details.

Trajectory Optimization Finally, to ensure our trajectory is natural and smooth, we linearly interpolate rotation and interpolate translation using cubic spline. The final trajectory for proposed task $q \in [1, Q]$ is denoted as $V^q = \{V_r^q\}_{r=0}^R$. Each pose in the final trajectory V^q is rendered as a frame in the rendering engine, where $V_0^q = V_0$, the rendering of the reconstructed 3D scene.



Fig. 4: Dataset statistics. Our dataset presents 51 scenes—13 from NOCS and 38 captured from varied perspectives—featuring a wide range of object categories, quantities, and a diverse set of tasks, task videos, and planned trajectories.

4 Experiments

Implementations We use ChatGPT-4 [47] as our VLM. The language-guided segmentation model is Language Segment-Anything [43] and the repainting model is LaMa [60]. We use One-2-3-45++ [38] for single-view 3D reconstruction and Depth Anything [69] for depth estimation. For partial 3D scene reconstruction, we use Blender [16] as the 3D software. The pinhole camera focal length and sensor width are set to 80 mm and 36 mm, with $\delta_{init} = 2.5e - 4$, $k_{init} = 1$. All inference is run on 1 NVIDIA A10 GPU with 24GB RAM.

Dataset As we are the first to propose zero-shot task hallucination, there lacks a pre-existing dataset for evaluation. Therefore, we craft a diverse evaluation dataset by combining self-captured photos and scenes from the NOCS dataset [62]. We capture 38 photos using an iPhone 12 Pro Max. From NOCS, we use its real-world part of both the training and test sets, which encompasses 13 distinct scenes. Since each scene is a video sequence, we randomly extract one frame as our data. We standardize all image dimensions by resizing all to 640×480 .

Dataset Statistics Our dataset covers diverse scenes (*e.g.*, office, kitchen, bathroom), and features a rich diversity of object categories (116) and quantities (185), with each image containing 1 – 7 objects and 1 – 3 tasks proposed for each object (278 tasks/task videos/planned trajectories in total). The dataset’s diversity is further enhanced by the variety of perspectives from which the images are captured or selected (*e.g.*, frontal, top-down, side views). This deliberate choice of diverse angles, both in our own image capturing process and through the random extraction of frames from NOCS, aims to simulate a realistic and challenging array of scenes for evaluation. See Fig. 4 for statistics and visuals.

Table 1: Comparison of task diversity. We sample 106 proposed tasks for fair comparison with RoboGen and previous RL benchmarks.

	Ours	RoboGen [66]	Behavior-100 [58]	RLbench [26]	MetaWorld [72]	Maniskill2 [22]
Number of Tasks	106	106	100	106	50	20
Self-BLEU ↓	0.269	0.284	0.299	0.317	0.322	0.674
Embedding Similarity ↓	0.154	0.165	0.210	0.200	0.263	0.194

Table 2: User study. We ask 25 users to rate (scale 1 – 5) and rank (scale 1 – 3) for 5 rotation and 5 translation tasks, and rate (scale 1 – 5) for 10 complex context-dependent tasks. All results take average. Note that our user study size is similar to those representative works such as ControlNet [73] and Prompt-to-Prompt [23].

	Rotation		Translation		Context-Dependent
	Rating ↑	Ranking ↓	Rating ↑	Ranking ↓	Rating ↑
3DIT [18]	1.93	2.34	1.90	2.27	-
Runway [52]	1.55	2.62	1.43	2.68	-
Ours	4.58	1.04	4.43	1.05	4.29

4.1 Task Diversity

We evaluate the diversity of the proposed tasks in terms of semantic meaning using Self-BLEU and the embedding similarity [74] following RoboGen [66], where lower scores mean higher diversity. We also compare with previous reinforcement learning (RL) benchmarks. From Table. 1, ours method generates most diverse tasks as our pipeline is open to all scenes with no constraint.

4.2 Qualitative Demonstration

We present qualitative examples spanning various scenes and tasks, illustrating our pipeline’s capability to produce diverse, visually appealing, and accurate hallucinated task videos for a wide range of input images, as shown in Fig. 5. Additional examples are available at our [project page](#).

We also compare with our baselines, SOTA 3D-aware image editing and video generation models, 3DIT [18] and Runway Gen-2 [52], as shown in Fig. 6. We limit comparisons to rotation and translation tasks to avoid bias against 3DIT and Runway in more nuanced context-dependent tasks (*e.g.*, slicing an apple), where their performances fall short. For 3DIT, we generate and connect frames with our planned trajectories. For Runway, we prompt it with task descriptions.

4.3 Human Preference Evaluation: User Study

We rely on human preference evaluation as one of our quantitative metrics. We let users rate and rank the rotation and translation task videos generated by 3DIT [18] and Runway Gen-2 [52] against our own in terms of both video quality and task description alignment. For these complex context-dependent tasks, we instead ask users to judge the videos’ authenticity and execution relative to

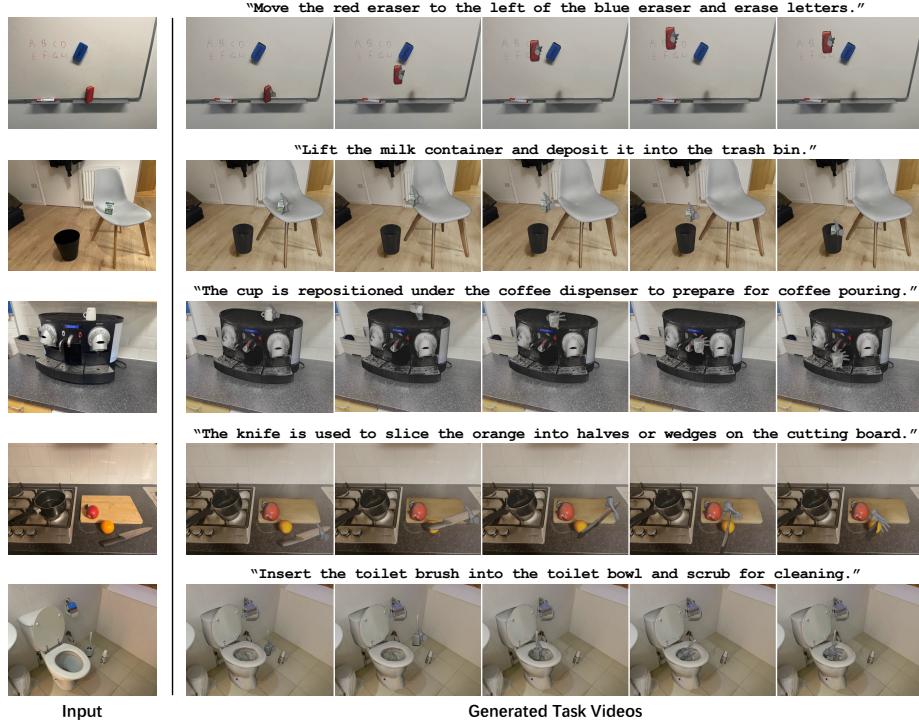


Fig. 5: Qualitative results. With diverse input scenes and proposed tasks, our model produces high-quality task execution visualizations with geometric awareness that aligns with the task descriptions. *Zoom in* for better view.

human action, and to encapsulate their perception of the action in our videos with a single sentence. The usage of sentence description will be explained in the subsequent Sec. 4.4. Table. 2 reveals the details and results.

4.4 Understanding Hallucination

Machine Understanding

We assess the interpretability of our generated task videos from a machine’s perspective using SOTA video understanding model, Video-LLaVA-7B [33]. We use two approaches: binary classification and descriptive generation. For classification, we feed the model with the task descriptions generated by VLM and ask question (is the video doing...?). In generation, we prompt Video-LLaVA-7B to articulate its interpretation of the tasks depicted in our videos. To quantify the correspondence between the model’s perception and the tasks, we use OpenCLIP cosine

However, we find that even SOTA video understanding model shows limited performance. To assess false positive rate in classification, we deliberately mis-

Table 3: Results for machine understanding (classification) on all 278 task videos.

	Machine
Raw Acc \uparrow	0.974
Fal-Pos Rate \downarrow	0.363
True Acc \uparrow	0.611



Fig. 6: Comparison with 3DIT [18] and Runway Gen-2 [52] on rotation and translation tasks. 3DIT fails to maintain consistency and precision across interpolated frames. Runway produces unpredictable artifacts. Our task videos are geometric-aware, photo-realistic with fine-grained and consistent object motion. *Zoom in for better view.*

Table 5: Task proposal ablation study. We use all 278 proposed tasks.

	Ours	Ours w/o obj-init	Ours w/o role-play	Ours w/o both
Involved Objects Number \uparrow	185	144	185	146
Self-BLEU \downarrow	0.273	0.280	0.276	0.282
Embedding Similarity \downarrow	0.143	0.156	0.141	0.159

align the sequence of generated task videos with their corresponding task descriptions, expecting a theoretical accuracy of 0%. Contrary to expectations, Video-LLaVA-7B reports a false positive rate of 36.3%. To adjust for this anomaly, we subtract this rate from the model’s raw accuracy for correctly aligned video-task pairs. This method, while unconventional, provides a more fair and reasonable evaluation of machine video understanding, underscoring the current challenges faced by video understanding models in accurately interpreting complex video content. Results in Table. 3.

Human Understanding We aim to understand human perception of our generated task videos by asking participants to provide a one-sentence description. We then evaluate the alignment between these descriptions and the ground truth task descriptions proposed by VLM, using OpenCLIP [14]. Table. 4 reveals a high degree of alignment. Intriguingly, it appears humans understand our hallucinated videos more accurately than machines do. We hypothesize this discrepancy stems from the limitations of current video understanding models, whereas humans draw on their prior experiences for a deeper comprehension.

4.5 Task Proposal Ablation Study

We aim to understand if our approach of task proposal, employing role-play and object-based initialization, yields the most diverse and feasible tasks by modifying the prompt to exclude these techniques. In addition to using Self-BLEU and

Table 4: Results for machine understanding (generation) on all 278 task videos and human understanding, where 25 users write descriptions for 5 videos.

	Machine	Human
OpenCLIP \uparrow	0.636	0.823

the embedding similarity, we also count the number of objects involved across all 278 proposed tasks.

The results in Table. 5 indicate that our method yields the most diverse tasks. Evaluating on all 278 proposed tasks shows a slight rise in Self-BLEU and a decrease in embedding similarity compared to the 106 tasks sampled (Table. 1). This difference arises because more task descriptions lead to repetitive structures and wording, yet the semantics grow more varied. Excluding object-based initialization reduces diversity by causing the VLM to overlook certain objects. Semantic meaning of tasks cannot reflect feasibility. Instead, we show instances where VLM generates impractical tasks without role-play in Appendix. C. We find that the impact of removing role-play is more complex, affecting task diversity less (Table. 5) but reducing task feasibility.

5 Discussion

We proposed a framework for *zero-shot task hallucination*. From the observation of a single image, our framework is able to discover diverse tasks and hallucinate task executions as geometric-aware and photo-realistic videos with fine-grained and consistent object motion. We see our work as a step towards equipping machines with the profound human capacity, in which humans can recall past knowledge and anticipate future scenarios through visual perception. Our generated task videos and planned trajectories could spark exciting applications, such as letting robots automatically explore and perform tasks in new environments.

Limitations Not all generated task videos achieve the highest quality. We identified key factors that contribute to the failure cases. Please see Appendix. D.

References

1. Ahn, M., Dwibedi, D., Finn, C., Arenas, M.G., Gopalakrishnan, K., Hausman, K., Ichter, B., Irpan, A., Joshi, N.J., Julian, R., Kirmani, S., Leal, I., Lee, T.E., Levine, S., Lu, Y., Leal, I., Maddineni, S., Rao, K., Sadigh, D., Sanketi, P., Sermanet, P., Vuong, Q., Welker, S., Xia, F., Xiao, T., Xu, P., Xu, S., Xu, Z.: Autort: Embodied foundation models for large scale orchestration of robotic agents. arXiv [abs/2401.12963](#) (2024) 3
2. AI, S.: Stable zero123: Quality 3d object generation from single images. <https://stability.ai/news/stable-zero123-3d-generation> (2023) 3
3. Betker, J., Goh, G., Jing, L., Brooks, T., Wang, J., Li, L., Ouyang, L., Zhuang, J., Lee, J., Guo, Y., Manassra, W., Dhariwal, P., Chu, C., Jiao, Y., Ramesh, A.: Improving image generation with better captions. <https://cdn.openai.com/papers/dall-e-3.pdf> (2023) 3
4. Bhat, S.F., Birk, R., Wofk, D., Wonka, P., Müller, M.: Zoedepth: Zero-shot transfer by combining relative and metric depth. arXiv [abs/2302.12288](#) (2023) 8
5. Birk, R., Wofk, D., Müller, M.: Midas v3.1 - A model zoo for robust monocular relative depth estimation. arXiv [abs/2307.14460](#) (2023) 8
6. Brooks, T., Holynski, A., Efros, A.A.: Instructpix2pix: Learning to follow image editing instructions. In: Conference on Computer Vision and Pattern Recognition. pp. 18392–18402. IEEE (2023) 3
7. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: Advances in Neural Information Processing Systems (2020) 2, 4
8. Burgess, J., Wang, K., Yeung, S.: Viewpoint textual inversion: Unleashing novel view synthesis with pretrained 2d diffusion models. arXiv [abs/2309.07986](#) (2023) 3
9. Cai, D., Heikkilä, J., Rahtu, E.: OVE6D: object viewpoint encoding for depth-based 6d object pose estimation. In: Conference on Computer Vision and Pattern Recognition. pp. 6793–6803. IEEE (2022) 7, 21
10. Chen, B., Xu, Z., Kirmani, S., Ichter, B., Driess, D., Florence, P., Sadigh, D., Guibas, L., Xia, F.: Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. arXiv [abs/2401.12168](#) (2024) 4
11. Chen, T., Saxena, S., Li, L., Fleet, D.J., Hinton, G.E.: Pix2seq: A language modeling framework for object detection. In: International Conference on Learning Representations (2022) 4
12. Cheng, T.Y., Gadelha, M., Groueix, T., Fisher, M., Mech, R., Markham, A., Trigoni, N.: Learning continuous 3d words for text-to-image generation. arXiv [abs/2402.08654](#) (2024) 3
13. Cheng, T.Y., Gadelha, M., Pirk, S., Groueix, T., Mech, R., Markham, A., Trigoni, N.: 3dminer: Discovering shapes from large-scale unannotated image datasets. In: International Conference on Computer Vision. pp. 9297–9307. IEEE (2023) 7
14. Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., Schuhmann, C., Schmidt, L., Jitsev, J.: Reproducible scaling laws for contrastive language-image learning. In: Conference on Computer Vision and Pattern Recognition. pp. 2818–2829. IEEE (2023) 13, 14

15. Cho, J., Min, D., Kim, Y., Sohn, K.: DIML/CVL RGB-D dataset: 2m RGB-D images of natural indoor and outdoor scenes. arXiv **abs/2110.11590** (2021) 8
16. Community, B.O.: Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam (2018), <http://www.blender.org> 11
17. Dalal, M., Chiruvolu, T., Chaplot, D.S., Salakhutdinov, R.: Plan-seq-learn: Language model guided RL for solving long horizon robotics tasks. In: CoRL Workshop on Learning Effective Abstractions for Planning (2023) 10
18. Deitke, M., Schwenk, D., Salvador, J., Weihl, L., Michel, O., VanderBilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., Farhadi, A.: Objaverse: A universe of annotated 3d objects. In: Conference on Computer Vision and Pattern Recognition. pp. 13142–13153. IEEE (2023) 2, 3, 12, 14
19. Epstein, D., Jabri, A., Poole, B., Efros, A.A., Holynski, A.: Diffusion self-guidance for controllable image generation. In: Advances in Neural Information Processing Systems (2023) 3
20. Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D., Zhu, Y., Anandkumar, A.: Minedojo: Building open-ended embodied agents with internet-scale knowledge. In: Advances in Neural Information Processing Systems (2022) 3
21. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research* pp. 1231–1237 (2013) 8
22. Gu, J., Xiang, F., Li, X., Ling, Z., Liu, X., Mu, T., Tang, Y., Tao, S., Wei, X., Yao, Y., Yuan, X., Xie, P., Huang, Z., Chen, R., Su, H.: Maniskill2: A unified benchmark for generalizable manipulation skills. In: Conference on Learning Representations (2023) 12
23. Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y., Cohen-Or, D.: Prompt-to-prompt image editing with cross-attention control. In: International Conference on Learning Representations (2023) 3, 12
24. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems (2020) 3
25. Hu, M.: Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory* pp. 179–187 (1962) 21
26. James, S., Ma, Z., Arrojo, D.R., Davison, A.J.: Rlbench: The robot learning benchmark & learning environment. *Robotics and Automation Letters* pp. 3019–3026 (2020) 12
27. Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., Fan, L.: Vima: General robot manipulation with multimodal prompts. In: International Conference on Machine Learning (2023) 3
28. Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C.L., Girshick, R.B.: CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In: Conference on Computer Vision and Pattern Recognition. pp. 1988–1997. IEEE (2017) 4
29. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research* pp. 846–894 (2011) 10
30. Khoshelham, K., Elberink, S.O.: Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors* pp. 1437–1454 (2012) 8
31. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollár, P., Girshick, R.: Segment anything. arXiv **abs/2304.02643** (2023) 4
32. Labs, P.: Pika. <https://pikalabs.org/about/> (2023) 2

33. Lin, B., Ye, Y., Zhu, B., Cui, J., Ning, M., Jin, P., Yuan, L.: Video-llava: Learning united visual representation by alignment before projection. arXiv **abs/2311.10122** (2023) **13**
34. Lin, J., Liu, L., Lu, D., Jia, K.: SAM-6D: segment anything model meets zero-shot 6d object pose estimation. arXiv **abs/2311.15707** (2023) **7, 21**
35. Liu, D., Dong, X., Zhang, R., Luo, X., Gao, P., Huang, X., Gong, Y., Wang, Z.: 3daxesprompts: Unleashing the 3d spatial task capabilities of GPT-4V. arXiv **abs/2312.09738** (2023) **9**
36. Liu, F., Emerson, G.E.T., Collier, N.: Visual spatial reasoning. Transactions of the Association for Computational Linguistics (2023) **4**
37. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. In: Advances in Neural Information Processing Systems (2023) **4**
38. Liu, M., Shi, R., Chen, L., Zhang, Z., Xu, C., Wei, X., Chen, H., Zeng, C., Gu, J., Su, H.: One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. arXiv **abs/2311.07885** (2023) **3, 7, 11, 23**
39. Liu, M., Xu, C., Jin, H., Chen, L., T, M.V., Xu, Z., Su, H.: One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. arXiv **abs/2306.16928** (2023) **3, 7**
40. Liu, R., Wu, R., Hoorick, B.V., Tokmakov, P., Zakharov, S., Vondrick, C.: Zero-1-to-3: Zero-shot one image to 3d object. In: International Conference on Computer Vision. pp. 9264–9275. IEEE (2023) **3**
41. Liu, Y., Wen, Y., Peng, S., Lin, C., Long, X., Komura, T., Wang, W.: Gen6d: Generalizable model-free 6-dof object pose estimation from RGB images. In: European Conference on Computer Vision. pp. 298–315 (2022) **7, 21**
42. Ma, Y.J., Liang, W., Wang, G., Huang, D., Bastani, O., Jayaraman, D., Zhu, Y., Fan, L., Anandkumar, A.: Eureka: Human-level reward design via coding large language models. arXiv **abs/2310.12931** (2023) **3**
43. Medeiros, L.: Language segment-anything. <https://github.com/luca-medeiros/lang-segment-anything> (2023) **11, 23**
44. Michel, O., Bhattacharjee, A., VanderBilt, E., Krishna, R., Kembhavi, A., Gupta, T.: OBJECT 3DIT: Language-guided 3d-aware image editing. In: Advances in Neural Information Processing Systems (2023) **3, 4**
45. Monnier, T., Fisher, M., Efros, A.A., Aubry, M.: Share with thy neighbors: Single-view reconstruction by cross-instance consistency. In: European Conference on Computer Vision. pp. 285–303 (2022) **7**
46. Nguyen, V.N., Groueix, T., Poniatkin, G., Lepetit, V., Hodan, T.: CNOS: A strong baseline for cad-based novel object segmentation. In: International Conference on Computer Vision Workshops. pp. 2126–2132. IEEE (2023) **7, 21**
47. OpenAI: GPT-4 technical report. arXiv **abs/2303.08774** (2023) **2, 4, 11, 22**
48. Pandey, K., Guerrero, P., Gadelha, M., Hold-Geoffroy, Y., Singh, K., Mitra, N.J.: Diffusion handles: Enabling 3d edits for diffusion models by lifting activations to 3d. arXiv **abs/2312.02190** (2023) **2, 3**
49. Parmar, G., Singh, K.K., Zhang, R., Li, Y., Lu, J., Zhu, J.: Zero-shot image-to-image translation. In: SIGGRAPH. pp. 11:1–11:11. ACM (2023) **3**
50. Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., Rombach, R.: SDXL: improving latent diffusion models for high-resolution image synthesis. arXiv **abs/2307.01952** (2023) **3**
51. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V.: Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1623–1637 (2022) **8**

52. Research, R.: More control, fidelity and expressibility. <https://research.runwayml.com/more-control-fidelity-and-expressibility> (2023) 2, 12, 14
53. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Conference on Computer Vision and Pattern Recognition. pp. 10674–10685. IEEE (2022) 3
54. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E.L., Ghasemipour, S.K.S., Lopes, R.G., Ayan, B.K., Salimans, T., Ho, J., Fleet, D.J., Norouzi, M.: Photorealistic text-to-image diffusion models with deep language understanding. In: Advances in Neural Information Processing Systems (2022) 3
55. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from RGBD images. In: European Conference on Computer Vision. pp. 746–760 (2012) 8
56. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: International Conference on Learning Representations (2021) 3
57. Song, Y., Zhang, Z., Lin, Z.L., Cohen, S., Price, B., Zhang, J., Kim, S.Y., Aliaga, D.G.: Objectstitch: Object compositing with diffusion model. In: Conference on Computer Vision and Pattern Recognition. pp. 18310–18319. IEEE (2023) 3
58. Srivastava, S., Li, C., Lingelbach, M., Martín-Martín, R., Xia, F., Vainio, K.E., Lian, Z., Gokmen, C., Buch, S., Liu, C.K., Savarese, S., Gweon, H., Wu, J., Fei-Fei, L.: BEHAVIOR: benchmark for everyday household activities in virtual, interactive, and ecological environments. In: Conference on Robot Learning. pp. 477–490 (2021) 12
59. Suddendorf, T., Addis, D.R., Corballis, M.C.: Mental time travel and the shaping of the human mind. Philosophical Transactions of the Royal Society B pp. 1317–1324 (2009) 2
60. Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Silvestrov, A., Kong, N., Goka, H., Park, K., Lempitsky, V.: Resolution-robust large mask inpainting with fourier convolutions. In: Winter Conference on Applications of Computer Vision. pp. 3172–3182. IEEE (2022) 11, 23
61. Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., Anandkumar, A.: Voyager: An open-ended embodied agent with large language models. arXiv **abs/2305.16291** (2023) 3
62. Wang, H., Sridhar, S., Huang, J., Valentin, J., Song, S., Guibas, L.J.: Normalized object coordinate space for category-level 6d object pose and size estimation. In: Conference on Computer Vision and Pattern Recognition. pp. 2642–2651. IEEE (2019) 11
63. Wang, Q., Zheng, S., Yan, Q., Deng, F., Zhao, K., Chu, X.: IRS: A large naturalistic indoor robotics stereo dataset to train deep models for disparity and surface normal estimation. In: International Conference on Multimedia and Expo. pp. 1–6. IEEE (2021) 8
64. Wang, W., Zhu, D., Wang, X., Hu, Y., Qiu, Y., Wang, C., Hu, Y., Kapoor, A., Scherer, S.A.: Tartanair: A dataset to push the limits of visual SLAM. In: International Conference on Intelligent Robots and Systems. pp. 4909–4916. IEEE (2020) 8
65. Wang, Y., Zhang, B., Chen, J., Sreenath, K.: Prompt a robot to walk with large language models. arXiv **abs/2309.09969** (2023) 10
66. Wang, Y., Xian, Z., Chen, F., Wang, T., Wang, Y., Erickson, Z., Held, D., Gan, C.: Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. arXiv **abs/2311.01455** (2023) 3, 12

67. Xian, K., Zhang, J., Wang, O., Mai, L., Lin, Z., Cao, Z.: Structure-guided ranking loss for single image depth prediction. In: Conference on Computer Vision and Pattern Recognition. pp. 608–617. IEEE (2020) [8](#)
68. Xu, Z., Qi, B., Agrawal, S., Song, S.: Adagrasp: Learning an adaptive gripper-aware grasping policy. In: International Conference on Robotics and Automation. pp. 4620–4626. IEEE (2021) [22](#)
69. Yang, L., Kang, B., Huang, Z., Xu, X., Feng, J., Zhao, H.: Depth anything: Unleashing the power of large-scale unlabeled data. arXiv **abs/2401.10891** (2024) [8, 11](#)
70. Yao, Y., Luo, Z., Li, S., Zhang, J., Ren, Y., Zhou, L., Fang, T., Quan, L.: Blended-mvs: A large-scale dataset for generalized multi-view stereo networks. In: Conference on Computer Vision and Pattern Recognition. pp. 1787–1796. IEEE (2020) [8](#)
71. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images. In: Conference on Computer Vision and Pattern Recognition. pp. 4578–4587. IEEE (2021) [7](#)
72. Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., Levine, S.: Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In: Conference on Robot Learning. pp. 1094–1100 (2019) [12](#)
73. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: International Conference on Computer Vision. IEEE (2023) [12](#)
74. Zhu, Y., Lu, S., Zheng, L., Guo, J., Zhang, W., Wang, J., Yu, Y.: Texxygen: A benchmarking platform for text generation models. In: SIGIR. pp. 1097–1100. ACM (2018) [12](#)

See, Imagine, Plan: Discovering and Hallucinating Tasks from a Single Image

Appendix

A Project Page

All code, data, and additional videos are available at our project page: https://dannymcy.github.io/zeroshot_task_hallucination/.

B Additional Details of Method

B.1 Matching-Based Camera 6D Pose Estimation

For single-view 3D reconstruction models which do not provide the 6D pose, $P_c = [R_c \mid t_c^{origin}]$, of the pinhole camera, we develop an efficient method to determine the pose by comparing object masks with rendered 3D model templates, inspired from matching-based 6D pose estimation works [9, 34, 41, 46].

We generate a set of object templates, denoted as $\{T_j^i\}_{j=1}^J$, each rendered from the object’s 3D model O_i . These templates are created by positioning the camera at various locations on an icosphere surrounding the object in $SE(3)$ space, which simulates a spherical coverage around the object to capture its geometry from all angles uniformly. For each template T_j^i , we compute a matching score against the occlusion-free object mask M_i^{of} .

We propose a simple yet effective score matching method. We draw a bounding rectangle around the segmented object inside M_i^{of} and across all $\{T_j^i\}_{j=1}^J$, and crop the bounding rectangle. We then calculate the shape similarity between the contour line of cropped M_i^{of} and that of each cropped T_j^i using Hu moments [25]. Additionally, we crop and resize the bounding rectangle to the same dimension, and evaluate the similarity based on the pixel area of the cropped and resized masks. Our score matching method can be formulated as:

$$\begin{aligned} m_i^{A,h} &= \text{Hu}(\text{findContour}(\text{crop}(M_i^{of}))), & pa_i^A &= \text{sum}(\text{resize}(\text{crop}(M_i^{of}))) \\ m_{i,j}^{B,h} &= \text{Hu}(\text{findContour}(\text{crop}(T_j^i))), & pa_{i,j}^B &= \text{sum}(\text{resize}(\text{crop}(T_j^i))) \\ \mathcal{L}(M_i^{of}, T_j^i) &= \alpha \left| 1 - \frac{\min(pa_i^A, pa_{i,j}^B)}{\max(pa_i^A, pa_{i,j}^B)} \right| + \beta \sum_{h=1}^7 \left| \frac{1}{\text{sgn}(m_i^{A,h}) \cdot \log(m_i^{A,h})} - \frac{1}{\text{sgn}(m_{i,j}^{B,h}) \cdot \log(m_{i,j}^{B,h})} \right| \end{aligned} \quad (1)$$

This dual approach allows for a comprehensive comparison that incorporates both the geometric configuration and the scale of the object representations. The best-matched template can be found by $\arg \min_{j=1}^J \mathcal{L}(M_i^{of}, T_j^i)$, and we set its corresponding camera pose to be the 6D pose of our pinhole camera, denoted as $P_c = [R_c \mid t_c^{origin}]$.

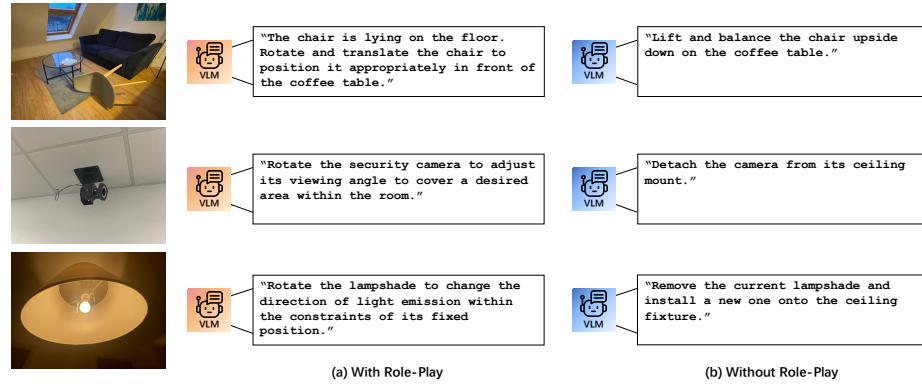


Fig. 1: Feasibility comparison of the proposed tasks with and without role-play.

The camera now captures the object’s canonical view, centered at the 3D origin. Subsequently, we adjust the camera’s translation component to focus on the center of the inpainted background I_{bg} , resulting in $P_c = [R_c \mid t_c]$, as detailed in our main paper.

B.2 Sampling Rules for VLM Planning Discrepancies

We define a set of geometric rules for Gaussian sampling around the initially planned waypoints whenever a collision is detected. In our 3D coordinate system, positive x-axis $[1, 0, 0]$ points right, positive y-axis $[0, 1, 0]$ is away from viewer, positive z-axis $[0, 0, 1]$ is up. For translation type 1, we denote the goal pose relative to the target object’s principle axes as $[dx, dy, dz]$. For translation type 2, we denote the distance that object 1 moves towards object 2 as dD . The set of geometric rules are as follows:

```

if type 1 &  $[dx, dy, dz] = [0, 0, 0]$ : sample along  $[x, y, z]$  axes
elif type 1 &  $dx = 0 \& dy = 0 \& dz \neq 0$ : sample along  $[z]$  axis,  $z_{\text{sampled}} \cdot dz > 1$ 
elif type 1 &  $dz = 0$ : sample along  $[x, y]$  axes
elif type 1 &  $dz \neq 0$ : sample along  $[x, y, z]$  axes,  $z_{\text{sampled}} \cdot dz > 1$ 
if type 2: sample along the connecting directional vector,  $D_{\text{sampled}} \cdot dD > 1$ 
  
```

B.3 Hand Insertion

Each pose in the final trajectory V^q is rendered as a frame in the rendering engine. To enhance human comprehension and perception, we integrate a 3D model of “a magic hand” [68] in proximity to the object being manipulated.

C Task Proposal Comparison of Feasibility

We show instances where ChatGPT-4 [47] generates impractical tasks without role-play in Fig. 1. These unfeasible tasks usually cannot be performed by a

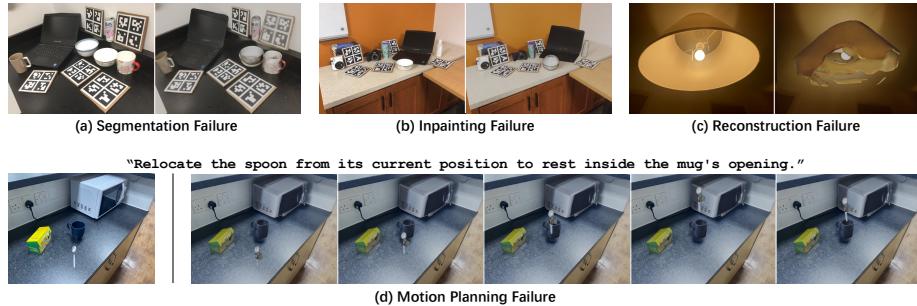


Fig. 2: Key factors causing failure cases. For each factor, the leftmost image is the input.

robotic/human hand due to physical limitations or require additional tools beyond the capabilities of the hand alone.

D Failure Cases

Not all generated task videos achieve the highest quality. We identify several key factors that contribute to the failure cases. Fig. 2 shows visualization.

Segmentation Failure When an image contains objects with very similar appearances, ChatGPT-4 can distinguish them, providing unique short descriptions. For instance, in the input image on the left of Fig. 2(a), ChatGPT-4 differentiates two white bowls as "Object 2: bowl of color white with blue pattern, "Object 3: bowl of color white with black pattern". Nonetheless, the language-guided segmentation model, Language Segment-Anything [43], may fail to capture the nuance, incorrectly merging object masks. This issue is depicted in Fig. 2(a)'s right reconstructed scene, where Language Segment-Anything recognizes and merges the two distinct bowls as one.

Inpainting Failure In rare cases, the repainting model LaMa [60] introduces unintended elements into and distorts object masks, illustrated by the deformed white bowl in Fig. 2(b)'s right reconstructed scene. Such distortions result in inaccurately reconstructed 3D models.

3D Model Reconstruction Failure The model One-2-3-45++ [38], despite its strength, occasionally reconstructs less-than-ideal 3D object models, as illustrated by the misshapen lamp shade in Fig. 2(c)'s right reconstructed scene.

Motion Planning Failure ChatGPT-4 occasionally produces suboptimal planned motions, primarily due to incorrect use of object principal axes and inaccurate translation measurements. For instance, as shown in Fig. 2(d), one would expect the spoon to be inserted into the mug. However, ChatGPT-4's planning positions the spoon too high, making it appear to float above the mug.



Fig. 3: Example input scene. (a) the original input image, (b) the reconstructed scene with visualized axes.

E Prompt Details

We show exact prompts and example responses from ChatGPT-4 for the white plate with coffee grounds shown in the input images of Fig. 3.

Interactive Objects Identification Prompt.

Input:

RGB image (640, 480) = (width, height) with multiple objects.

Your task is to identify and name useful objects (in term of affordances) with precise color and texture. Exclude background objects. Do not use vague phrase like multi-colored.

Please write in the following format. Do not output anything else:

Object idx (actual integer, start from 0): x of color y

An example response from ChatGPT-4:

```
"Object 0: sink of color silver
Object 1: faucet of color silver
Object 2: dish soap of color yellow
Object 3: hand soap of color purple
Object 4: sponge of color blue
Object 5: white plate with coffee grounds of color brown"
```

Task Proposal Prompt.

Input:

1. RGB image (640, 480) = (width, height) with multiple objects.
2. Detected objects with index.

You are a single robot hand working in this image scene to perform simple household tasks. Tasks must be discovered from the image. Consider objects' affordances and feel free to make assumptions

(e.g., a bowl can contain water) and interactions with other objects (e.g., pouring water from a cup into a bowl).

Task types:

1. Interaction between the manipulating object and one of the detected objects (involve translation, or translation + rotation).
2. Rotate manipulating object (involve rotation).

Strictly follow constraints:

1. Exclude tasks involving assembly or disassembly of objects.
2. Exclude tasks involving cleaning or functionality testing.
3. Exclude tasks involving imaginary objects.
4. Manipulating object moves; interacting object static.
5. Assume all objects are rigid, without joints or moveable parts (i.e., cannot deform, disassemble, transform). This applies even to objects that are typically articulated (e.g., laptop).

Propose 3 tasks (2 interaction, 1 rotation) for manipulating Object 5. Write in the following format.

Do not output anything else:

Task Name: xxx

Manipulating obj idx: 5

Interacting obj idx: obj_idx (actual integer, or manipulating obj idx)

Description: basic descriptions.

An example response from ChatGPT-4:

"Task Name: Plate Orientation

Manipulating obj idx: 5

Interacting obj idx: 5

Description: Rotate the plate 180 degrees horizontally to change its orientation while remaining in the same location on the sink.

Task Name: Plate Relocation

Manipulating obj idx: 5

Interacting obj idx: 0

Description: Move the plate from its current location to the right side of the sink to clear space for other kitchen activities.

Task Name: Washing Plate

Manipulating obj idx: 5

Interacting obj idx: 1

Description: Position the plate beneath the faucet and simulate the process of rinsing it."

Image and Spatial Context Understanding Prompt.

Inputs:

1. RGB image (640, 480) = (width, height) with multiple objects and their visualized local axes (x red, y green, z blue).

2. Detected objects with index.
3. For each detected object, its 3D center, local xyz-axes, size, and spatial relationship relative to other objects.

The 3D coordinate system of the image is in centimeters and follows Blender. Positive x-axis [1, 0, 0] right, positive y-axis [0, 1, 0] away from viewer, positive z-axis [0, 0, 1] up. Positive rotation is counter-clockwise around all axes.

Your task is to learn the spatial context. Also, consider how a robot or human hand might navigate or interact with these objects, focusing on proximity, accessibility, and collision avoidance.
Do not output.

An example input of object spatial information to ChatGPT-4:

```
"Obj 0 spatial context:  
3D center: [-21.0, 90.0, -14.0] cm  
Local x-axis (towards 'right'): [0.9932, -0.0882, -0.0766]  
Local y-axis (towards 'back'): [0.1104, 0.9234, 0.3675]  
Local z-axis (towards 'up'): [0.0384, -0.3735, 0.9268]  
Obj 1 spatial context:  
3D center: [-35.0, 85.0, 8.0] cm  
Local x-axis (towards 'front_right'): [0.8565, -0.5126, 0.0612]  
Local y-axis (towards 'back_right'): [0.5151, 0.8563, -0.0377]  
Local z-axis (towards 'up'): [-0.0331, 0.0638, 0.9974]  
Obj 2 spatial context:  
3D center: [31.0, 80.0, 13.0] cm  
Local x-axis (towards 'right'): [0.9971, -0.0062, 0.0756]  
Local y-axis (towards 'back'): [0.008, 0.9997, -0.0226]  
Local z-axis (towards 'up'): [-0.0754, 0.0231, 0.9969]  
Obj 3 spatial context:  
3D center: [33.0, 71.0, 8.0] cm  
Local x-axis (towards 'right'): [0.8162, -0.0206, 0.5775]  
Local y-axis (towards 'back'): [-0.0169, 0.9981, 0.0595]  
Local z-axis (towards 'up'): [-0.5776, -0.0583, 0.8142]  
Obj 4 spatial context:  
3D center: [22.0, 97.0, 3.0] cm  
Local x-axis (towards 'right'): [0.999, 0.0383, -0.0231]  
Local y-axis (towards 'back'): [-0.0399, 0.9966, -0.0721]  
Local z-axis (towards 'up'): [0.0203, 0.073, 0.9971]  
Obj 5 spatial context:  
3D center: [-12.0, 78.0, -13.0] cm  
Local x-axis (towards 'right'): [0.9998, -0.0216, -0.001]  
Local y-axis (towards 'back'): [0.0216, 0.9998, 0.0052]  
Local z-axis (towards 'up'): [0.0009, -0.0052, 1.0]  
  
Obj 0 size: 47.63 cm x 39.15 cm x 22.25 cm (WxDxH)
```

```

Obj 1 size: 19.11 cm x 13.82 cm x 17.83 cm (WxDxH)
Obj 2 size: 9.69 cm x 9.19 cm x 15.63 cm (WxDxH)
Obj 3 size: 7.74 cm x 6.91 cm x 13.97 cm (WxDxH)
Obj 4 size: 9.57 cm x 2.17 cm x 5.35 cm (WxDxH)
Obj 5 size: 12.17 cm x 11.90 cm x 7.03 cm (WxDxH)

Obj 0 - Closest per direction: up: Obj 1 (26.75 cm); right: Obj 4 (46.28 cm); front_right: Obj
5 (14.55 cm)
Obj 1 - Closest per direction: down: Obj 0 (26.75 cm); right: Obj 5 (31.93 cm)
Obj 2 - Closest per direction: left: Obj 5 (50.92 cm); front: Obj 3 (11.06 cm); back_left: Obj
4 (21.93 cm)
Obj 3 - Closest per direction: left: Obj 5 (49.76 cm); back: Obj 2 (11.06 cm)
Obj 4 - Closest per direction: left: Obj 0 (46.28 cm); front_right: Obj 2 (21.93 cm); front: Obj
3 (28.77 cm); front_left: Obj 5 (41.97 cm)
Obj 5 - Closest per direction: back_left: Obj 0 (14.55 cm); left: Obj 1 (31.93 cm); right: Obj
3 (49.76 cm); back_right: Obj 4 (41.97 cm)"

```

Motion Planning Prompt.

Inputs:

1. RGB image (640, 480) = (width, height) with multiple objects and their visualized local axes (x red, y green, z blue).
2. Detected objects with index.
3. Simple household tasks and descriptions to be performed by a single robot hand.

Your goal is to plan fine-grained motions for the manipulating object to complete the tasks using four manipulations, explained as follows:

Rotation:

`rotate_self`: Axial rotation. The object rotates around its local [x/y/z] axis by [degrees].
`rotate_wref`: Rotation relative to the target object:
- `pitch`: Tilt similar to pouring water, around a horizontal axis formed by the cross product of the connecting directional vector and the target's z-axis.
- `yaw`: Horizontal rotation, like a camera panning, around a vertical axis formed by the cross product of the connecting directional vector and the pitch axis.
- `roll`: Rotation like a drill entering a surface, around the connecting directional vector.

The degrees can be specified in two ways:

- Exact [degrees]. Positive values rotate the manipulating object towards the target object.
- `Fixed_towards/fixed_back`. '`fixed_towards`' orients the object towards the target, mimicking actions like pouring (pitch), facing (yaw), or drilling into (yaw+roll) the target. '`fixed_back`' reverses this alignment.

Translation:

`translate_tar_obj`: Defines the goal relative to the target object's local axes, with translation values for its [local_x, local_y, local_z] axes in centimeters. [0, 0, 0] cm indicates the goal is the center of the target object.

`translate_direc_axis`: Sets the goal relative to a directional vector between two reference objects, specifying how far (in cm) object 1 should move towards or away from object 2 along this vector (positive closer, negative away). Object indices must differ, and if one reference object is the manipulating object, its current location is used.

Strictly follow caveats:

1. Apply `rotate_wref` thoughtfully and sequentially around different axes as needed.
2. Use the provided spatial information and image effectively for understanding and planning within the 3D scene.
3. Combine common physical understanding with the scene's spatial details (like relative positions and sizes of objects) for strategic planning.
4. Remember that objects' local axes' positive directions might require using negative values in rotation and translation for authentic motion planning.

Plan as below. Fill in `obj_idx` based on the tasks.

```
rotate_self: Rotate Manipulating Object [obj_idx] around its local axis [x/y/z] by [degrees].
rotate_wref: Rotate Manipulating Object [obj_idx] relative to Target Object [target_obj_idx] around [pitch/yaw/roll] axis by [degrees/fixed_towards/fixed_back].
translate_tar_obj: Move Manipulating Object [obj_idx] to [a, b, c] cm relative to Target Object [target_obj_idx]'s local [x, y, z] axes.
translate_direc_axis: Move Manipulating Object [obj_idx] [a] cm along the directional vector from Reference Object [ref_obj_1_idx] to Reference Object [ref_obj_2_idx].
```

Here are some full examples. Please write in the following format. Do not output anything else:

Task Category: Bear rotation

Description: Rotate the toy bear 90 degrees on its vertical axis.

Motion Planning:

Manipulating obj idx: `bear_idx` (actual integer)

Interacting obj idx: `bear_idx` (actual integer)

1. `rotate_self`: Rotate Manipulating Object [`bear_idx`] around its local axis [`z`] by [90] degrees.

Task Name: Cup content transfer

Description: Pick up the mug and pour its contents into the bowl.

Motion Planning:

Manipulating obj idx: `cup_idx` (actual integer)

Interacting obj idx: `bowl_idx` (actual integer)

1. `translate_tar_obj`: Move Manipulating Object [`cup_idx`] to [5, -7, 5] cm relative to Target Object [`bowl_idx`]’s local [x, y, z] axes.
2. `rotate_wref`: Rotate Manipulating Object [obj_idx] relative to Target Object [bowl_obj_idx] around [pitch] axis by [fixed_towards].

Task Name: Screwdriver penetration

Description: Use a screwdriver to penetrate an avocado.

Motion Planning:

Manipulating obj idx: `screw_idx` (actual integer)

```

Interacting obj idx: avocado_idx (actual integer)
1. translate_tar_obj: Move Manipulating Object [screw_idx] to [-5, -5, 0] cm relative to Target Object [avocado_idx]'s local [x, y, z] axes.
2. rotate_wref: Rotate Manipulating Object [screw_idx] relative to Target Object [avocado_idx] around [yaw] axis by [fixed_towards].
3. rotate_wref: Rotate Manipulating Object [screw_idx] relative to Target Object [avocado_idx] around [roll] axis by [360] degrees.

```

An example response from ChatGPT-4:

```

"Task Name: Plate Orientation
Description: Rotate the plate 180 degrees horizontally to change its orientation while remaining in the same location on the countertop.
Motion Planning:
Manipulating obj idx: 5
Interacting obj idx: 5
1. rotate_self: Rotate Manipulating Object 5 around its local axis [z] by [180] degrees.

Task Name: Plate Relocation
Description: Move the plate from its current location to the right side of the sink to clear space for other kitchen activities.
Motion Planning:
Manipulating obj idx: 5
Interacting obj idx: 0
1. translate_tar_obj: Move Manipulating Object 5 to [26.5, 0, 0] cm relative to Target Object 0's local [x, y, z] axes.

Task Name: Washing Plate
Description: Position the plate beneath the faucet and simulate the process of rinsing it.
Motion Planning:
Manipulating obj idx: 5
Interacting obj idx: 1
1. translate_tar_obj: Move Manipulating Object 5 to [6.5, 0, -5] cm relative to Target Object 1's local [x, y, z] axes.
2. rotate_self: Rotate Manipulating Object 5 around its local axis [x] by [-15] degrees.
3. rotate_self: Rotate Manipulating Object 5 around its local axis [y] by [-15] degrees.
4. rotate_self: Rotate Manipulating Object 5 around its local axis [y] by [10] degrees.
5. rotate_self: Rotate Manipulating Object 5 around its local axis [x] by [10] degrees."

```