

IMPACT[🔥]: Intelligent Motion Planning with Acceptable Contact Trajectories via Vision-Language Models

Yiyang Ling*, Karan Owalekar*, Oluwatobiloba Adesanya, Erdem Biyik, Daniel Seita

Abstract—Motion planning involves determining a sequence of robot configurations to reach a desired pose, subject to movement and safety constraints. Traditional motion planning finds collision-free paths, but this is overly restrictive in clutter, where it may not be possible for a robot to accomplish a task without contact. In addition, contacts range from relatively benign (e.g., brushing a soft pillow) to more dangerous (e.g., toppling a glass vase), making it difficult to characterize which may be acceptable. In this paper, we propose IMPACT, a novel motion planning framework that uses Vision-Language Models (VLMs) to infer environment semantics, identifying which parts of the environment can best tolerate contact based on object properties and locations. Our approach generates an anisotropic cost map that encodes directional push safety. We pair this map with a contact-aware A* planner to find stable contact-rich paths. We perform experiments using 20 simulation and 10 real-world scenes and assess using task success rate, object displacements, and feedback from human evaluators. Our results over 3200 simulation and 200 real-world trials suggest that IMPACT enables efficient contact-rich motion planning in cluttered settings while outperforming alternative methods and ablations. Our project website is available at <https://impact-planning.github.io/>.

I. INTRODUCTION

Classical motion planning for robot manipulation [23], [36] frames the problem as finding a path for the robot’s end-effector to reach a target while avoiding collisions with obstacles. This formulation is generally desirable, but can be highly restrictive, especially in densely cluttered environments [7]. In such cases, some incidental contact may be necessary to achieve a task, or to accomplish it more efficiently than by strictly avoiding all collisions.

Consider the example in Fig. 1, which shows a robot manipulator making contact with a toy bear to efficiently reach the target spice jar. Due to the clutter, a collision-free path either does not exist or would require a longer parabolic motion to go above the obstacles (which, again, may not be feasible in cluttered cabinets and boxes). We thus desire robots that can achieve a task while moving through a cluttered environment, making appropriate contact as needed. In this work, the term “contacts” does not refer to when a robot uses its grippers to touch a target object (e.g., for grasping). Instead, we consider other types of contact: when any part of the robot touches any *non-target* (or “distractor”) object in the environment. We study this in motion planning for tasks that involve reaching to a target in dense clutter.

*Equal contribution.

All authors are with Thomas Lord Department of Computer Science, Viterbi School of Engineering, University of Southern California, USA.

Correspondence: {lingyiya, kowaleka, seita}@usc.edu.

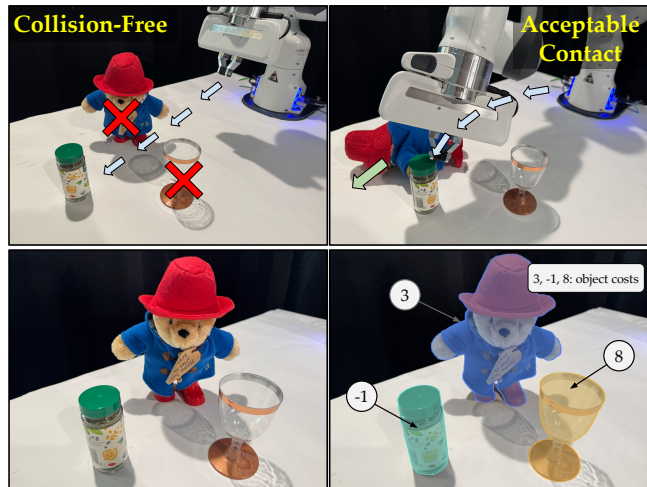


Fig. 1: An example of a reaching task and object costs. The first row shows the difference between collision-free paths and paths with acceptable contact. Left: Collision-free paths prevent a “straight” path to the spice jar because of the toy bear and wine glass obstacles (each marked with a red “X”). Right: With semantically acceptable contact, the robot can successfully reach the spice jar by pushing the toy bear and avoiding the fragile wine glass. The second row shows the cost of each object generated by GPT-4o. Left: the original scene. Right: GPT-4o assigns different costs to objects, with the target assigned -1 (toy bear: 3, spice jar: -1 , wine glass: 8).

To address this challenge, we propose Intelligent Motion Planning with Acceptable Contact Trajectories (IMPACT), a novel framework that leverages modern Vision-Language Models (VLMs) such as GPT-4o [27] to infer initial semantic costs for objects in the scene. However, knowing that an object is generally safe to touch is different from knowing *how* and *where* to push it without causing a cascade of collisions. Our method repeatedly samples push outcomes to estimate the probability of causing a collision with obstacles or the target. This risk is condensed into safety scores that are used to construct an anisotropic cost map, where interaction costs vary with direction since pushing an object from one side versus another can lead to different outcomes. We integrate this directional cost map with a contact-aware A* planner [10] that computes paths to reach targets. We perform experiments on contact-rich, densely cluttered reaching tasks in simulation and real-world settings. We evaluate IMPACT using multiple quantitative metrics, including human preference rankings from a user study. Results indicate that users prefer IMPACT over alternatives, suggesting that it is a promising approach for facilitating semantically-acceptable contact-rich manipulation.

Our contributions are as follows:

- IMPACT, a framework that formalizes “acceptable contact” by transforming initial VLM-inferred semantic costs into a dense, anisotropic cost map to represent the directional safety of physical interactions.
- A contact-aware A* planner capable of interpreting this anisotropic map to execute paths with intelligent and minimal-impact contact.
- Extensive simulation and real-world experiments, including a human subjects study, that validate our approach using both objective and subjective metrics to evaluate the success of contact-rich trajectories.

II. RELATED WORK

A. Robot Motion Planning

Classical motion planning algorithms can be broadly characterized [36] as optimization-based such as TrajOpt [32], or sampling-based such as PRM [15], RRT [18], and RRT* [14]. While there are numerous variants of these methods, a common underlying theme is the constraint of avoiding any collisions. In densely cluttered scenarios where contact is inevitable, these motion planners might not find a solution, even though one might exist if the robot engages in light contact. Prior work like Navigation Among Movable Obstacles [35] and Minimum Constraint Removal [11], [37] reason about relocating or removing obstacles to obtain feasible paths. Our method seamlessly integrates with prior motion planning algorithms for flexible contact-rich manipulation, without requiring explicit object relocation or environment modification.

Some motion planning works modify cost functions so that certain obstacles, such as leaves in foliage, are permeable [16], [26] and thus allow for some contact. These methods have been applied to domain-specific foliage settings and not tested in common cluttered scenarios with rigid objects. In closely-related work, Xie et al. [41] propose to use semantic language commands to enable semantically-acceptable contact. Unlike [41], we do not require explicit language instructions about which contacts are acceptable or not, since we utilize VLMs to automate this process.

B. Contact-Rich Robot Manipulation

Dealing with contacts is challenging in manipulation [36]. This may refer to frequent contact between an object (that the robot grips/holds) and an environment, such as tight placement tasks like connector and peg insertion as explored in robotic reinforcement learning works [19], [20], [31]. Another method for contact-rich problems is extrinsic dexterity [4], [42], which takes advantage of contacts between an object and rigid parts of the environment (such as walls) to reorient the object to improve subsequent manipulation. Other works consider contact-rich interactions in different applications such as assistive robots that incorporate human feedback and touch [13]. In contrast, we consider the relatively less-explored “contact-rich” setting, where “contact” refers to robot parts touching the environment’s obstacles.

C. Vision-Language Models (VLMs) for Robotics

VLMs such as GPT-4o [27] and Gemini [9] are increasingly applied in robotics for high-level planning [1], [8], [21], [33], reward or task design [24], [34], [38], [39], and low-level affordance reasoning [17], [22]. In contrast to these works, we use VLMs specifically to infer object contact tolerances to guide contact-rich motion planning.

In closely-related work, VoxPoser [12] exploits VLMs for open-world reasoning and visual grounding to compose a 3D value map to guide robotic interactions. As in [12], we use VLMs to assign semantic costs to objects. However, unlike VoxPoser, which requires the user to explicitly specify which objects to avoid with language, e.g., “*get the item, but watch out for that vase!*” we do not require explicit language commands. Instead, we leverage the improving spatial and semantic knowledge of recent VLMs [5] to determine object contact tolerances, and show manipulation in more densely cluttered environments. Other recent work relies on VLMs for semantically-safe manipulation [2] but assumes that any collisions are undesirable.

III. PROBLEM STATEMENT AND ASSUMPTIONS

We assume a single robot arm with a standard gripper operates in a densely cluttered environment that contains n objects, denoted as $\mathcal{O} = \{o_1, o_2, \dots, o_n\}$. The robot must reach a given target object $o_{\text{targ}} \in \mathcal{O}$ while minimizing unwanted contact with other objects (i.e., obstacles) in $\mathcal{O} \setminus o_{\text{targ}}$. In environments with significant clutter, o_{targ} may be behind or close to multiple objects and thus reaching it may be infeasible with a collision-free trajectory. To reduce occlusions, at least two cameras provide respective RGBD images at the start of the task. Given these image observations, the objective is to compute a trajectory τ for the robot, defined as a sequence of end-effector poses, such that its end-effector ultimately touches o_{targ} . Thus, among the dynamically feasible trajectories, our objective is to select one that reaches the target while engaging in “semantically-acceptable” contact with obstacles when needed.

IV. METHOD: IMPACT

Our framework consists of two main steps (see Fig. 2). First, it uses a VLM to obtain object costs in a cluttered scene (Sec. IV-A), and then uses those costs for contact-rich motion planning (Sec. IV-B).

A. Obtaining Object Costs using a VLM

A key technical challenge is defining the notion of an “acceptable” contact. This depends heavily on semantics, or the general-purpose commonsense knowledge that humans have about the behavior of diverse objects. Different objects with varying materials, geometries, sizes or purposes have different tolerances to potential contact. Furthermore, tolerance to contact should also depend on an object’s proximity to nearby objects. Therefore, we leverage the commonsense knowledge in VLMs to estimate the tolerance rate for contact of each object. We encode this information by assigning each object to an integer in $\{0, 1, \dots, 10\}$ as the cost, where a

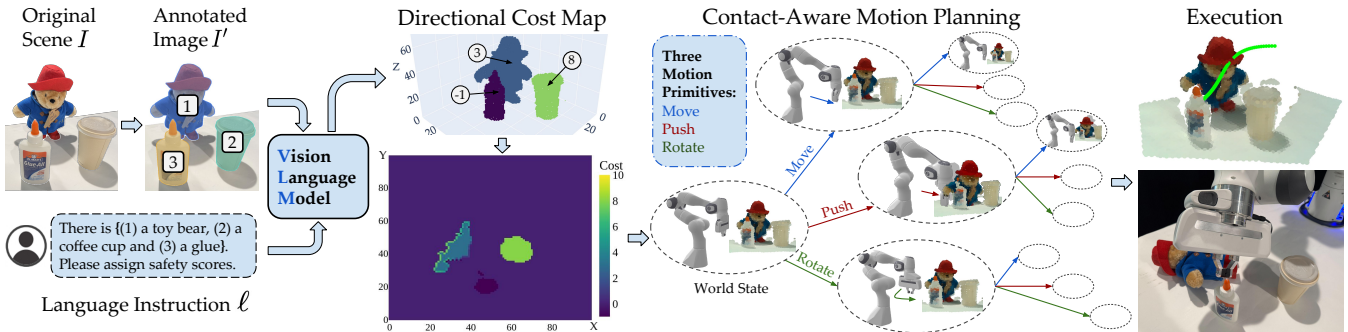


Fig. 2: Overview of IMPACT. There is a toy bear, a coffee cup and a glue bottle (target) on the table. The VLM receives an annotated image I' and a language template prompt ℓ with object information from SAM2 [29], and outputs *costs* for the three objects. We use a cost of -1 for the target object. We construct a 3D voxel grid V using these costs and then flatten it to produce an anisotropic, contact-aware cost map M' . The contact-aware A* planner searches over three motion primitives in this map: *Move*, *Push* and *Rotate* to generate a trajectory. The planner’s state space includes the robot’s end-effector pose and the displaced positions of low-cost objects. These guide the robot to avoid the coffee cup but make contact with the toy bear at the appropriate direction to reach the glue bottle.

higher cost indicates lower tolerance to contact with any part of the robot arm as it executes a trajectory. For example, the cost of a fragile object (e.g., wine glass) should be significantly higher compared to the cost of an object that can better absorb contact (e.g., foam rubber).

We use GPT-4o (hereafter, GPT) as the VLM to generate the cost of all objects in the scene due to its strong spatial reasoning capabilities [5], but our approach is compatible with other VLMs. The VLM is used in a zero-shot inference setting without any fine-tuning. The input to the VLM includes both an annotated image and a text prompt. We mount an RGBD camera to capture a scene image I , and use SAM2 [29] to segment and annotate the objects. The segmented and annotated image I' is ultimately provided as part of the input to GPT. We also design a text prompt template ℓ which includes the list of objects in the scene (but is otherwise task-agnostic), and some general principles related to the concept of contact tolerance. Objects are labeled with the same numbers as in the input image, marked by SAM2. The full text prompt can be found on our website. The output of GPT is a dictionary with the cost of all queried objects. Fig. 1 (second row) shows an example of object costs generated from GPT. It assigns a high cost of 8 to the wine glass and a lower cost of 3 to the toy bear, which indicates different tolerances to contact across objects.

B. Directional Contact-Aware Motion Planning

We use motion planning algorithms to synthesize robot trajectories. We construct a 3D voxel grid V where each voxel $V[x, y, z]$ denotes the cost at position (x, y, z) . During planning, we assign all voxels corresponding to the target object a cost of -1 to encourage the planner to find a path to it. To evaluate directional contact information, we project V into a 2D grid M using a top-down view, where $M[x, y] = \max_z V[x, y, z]$. From this 2D perspective, all obstacles are divided into low-cost (5 or lower) and high-cost (6 or higher) sets based on their VLM-assigned costs. To simplify planning, we define three motion primitives in 2D space: (i) *Move*, which translates the robot to one of eight neighboring positions; (ii) *Rotate*, which discretizes

the end-effector orientation changes within $\pm 45^\circ$; and (iii) *Push*, which moves the end-effector within a bounded radius while making contact with an object. We assume the first two change the end-effector’s pose without affecting the environment. In contrast, the *Push* primitive denotes end-effector translations that involve some robot-object contact.

1) *Anisotropic Cost Map Generation:* The map M encodes object costs independent of motion direction. In clutter, however, contact safety depends strongly on the direction where the robot approaches an object. To address this, we construct a new **anisotropic cost map** M' that augments each point with directional safety information. Defining M' over both 2D positions and 2D directions would be high-dimensional and computationally expensive, so we keep M' as a 2D function. For each point (x, y) on the boundary of a low-cost object, we consider the *reverse surface normal* vector, which points “inwards” towards the object. This gives the most natural and direct push for the object, but to consider different possible pushes, we sample m push outcomes with small variations in distance and angle, weighted by a 2D Gaussian centered on that normal (so outcomes closer to that vector receive higher likelihood). Each hypothesized outcome is evaluated on M by checking whether the displaced object would overlap with other objects, and classified into four categories: safe, low-cost contact, high-cost contact or contact with the target. Push outcomes in the same category are assigned an identical safety score $u \in (0, 1]$, with larger values indicating safer outcomes. The aggregated safety score at (x, y) is:

$$f_s(x, y) = \frac{\sum_{i=1}^m l_i u_i}{\sum_{i=1}^m l_i}, \quad (1)$$

where u_i encodes the safety level, and l_i is the likelihood of the i -th push outcome. This score is then used to update the cost map at (x, y) , creating the final anisotropic cost map M' where each point has an associated directional push score:

$$M'[x, y] = \alpha M[x, y] + (1 - \alpha)[10 - 10f_s(x, y)]. \quad (2)$$

Here, $\alpha \in (0, 1)$ is a weight parameter to balance the object cost and the directional safety score $f_s \in [0, 1]$, and

the constant 10 ensures the updated cost in M' remains consistent with the original object cost range in M .

2) *Contact-Aware A* Planning*: To generate a plan, we search for a sequence of primitives that guides the end-effector to the target while accounting for directional costs in M' . We formulate this as a graph search problem and use A* [10], which is well-suited for this setting and provides deterministic control compared to sampling-based algorithms. The planner’s state is defined as $S = (p, r, \mathcal{D})$, where (p, r) is the robot end-effector position and orientation in 2D space and \mathcal{D} is a list that tracks the cumulative 2D displacement of low-cost objects. At each step, the planner expands nodes using the three primitives. For Push, we estimate the object displacement using $d = D \cdot (\cos(\varphi))^\gamma$, where D is the maximum displacement when pushing in the optimal direction, φ is the angular deviation between the push direction and the object’s reverse surface normal at the contact point, and the decay parameter γ is set to 1.3. IMPACT extends standard A* search by incorporating world state changes caused by pushes. Each valid Push creates a new branch in the search tree where an object has been permanently displaced to a new position. The resulting state, $S' = (p', r', \mathcal{D}')$, contains the updated robot pose and an updated world configuration.

A* search evaluates each state s using the cost function $f(s) = g(s) + h(s)$, where $g(s)$ is the accumulated path cost from the start to state s , and $h(s)$ is a heuristic estimate of the remaining cost to the goal. We use the standard Euclidean distance to the target object as the heuristic $h(s)$. The accumulated path cost $g(s')$ is updated at each step with action $a : s = (p, r, \mathcal{D}) \rightarrow s' = (p', r', \mathcal{D}')$ as:

$$g(s') = g(s) + C(a) + P(s'), \quad (3)$$

where $C(a)$ is the action cost and $P(s')$ is a gripper placement penalty. We define action costs based on the primitive. Move assigns a cost proportional to the Euclidean distance between two end-effector positions. Rotate applies a fixed cost for orientation changes. Push generates its cost from the value $M'[x, y]$ at the contact point (x, y) . The penalty $P(s')$ consists of three terms: (i) the average cost of the region the gripper overlaps, (ii) a collision penalty if the gripper collides with an object, and (iii) a proximity penalty proportional to the distance to high-cost objects. For additional details of push analysis and the planner, please refer to the appendix in the arXiv version. By combining these costs, the planner can find a trajectory that minimizes risk while permitting acceptable contact when necessary; see Fig. 3 for more information about the A* planner. The path is then converted back to 3D space for evaluation and execution.

V. SIMULATION EXPERIMENTS

A. Experiment Setup in Simulation

We build and test our pipeline using PyBullet simulation [6]. In simulation, we create 20 scenes designed to test contact-rich manipulation. These use a hybrid object dataset, which includes tall and bulky items (e.g., sugar boxes and water pitchers) from the YCB [3] dataset and

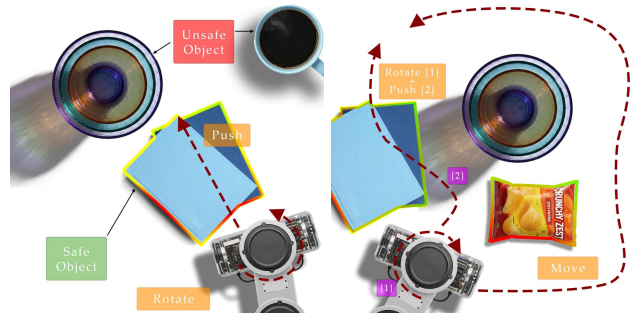


Fig. 3: A* planner decision-making in two key scenarios, where the books and pack of chips are low-cost objects, while the mug and the stack of bowls are high-cost objects. The colored border around low cost objects visualizes anisotropic costs (red is unsafe, green is safe). On the left, the planner avoids a direct Push towards the stack of bowls (cost ∞) and instead chooses a low-cost Rotate to navigate between objects (cost 7.0). On the right, by planning several steps ahead, it finds an efficient path by rotating and pushing stack of books (total cost 23.5). It avoids a simpler but high-cost detour that only considers Move (cost 50.0).

fragile objects (e.g., wine glasses and stacked bowls) using 3D models generated from TRELIS [40]. TRELIS allows constructing delicate object meshes from single-view images and introduces fragility constraints not present in YCB objects. All objects are placed on a tabletop.

We use three cameras to capture the scene for cost map generation: one in front of the scene and two on the sides. The camera in front also captures the scene image I used as input to the VLM. To prevent interactions with the tabletop, we explicitly assign its cost to be 10. To quantify the benefit of using VLM-generated costs, we compare with two control groups: (i) with no push action allowed to obtain a collision-free path, which we test in our “Collision-Free Planning” baseline (see Sec. V-C), and (ii) with the cost of all objects set to 0 to see how the planner will plan the path if all collisions are allowed (see Sec. V-F).

B. Motion Planning Methods

As presented in Sec. IV-B, IMPACT performs directional contact-aware path planning on the flattened cost map M' . In simulation, we also test the 3D cost map V with RRT [18] and RRT* [14] as the motion planners, and thus call the respective methods VLM Cost+RRT and VLM Cost+RRT*. Each trajectory consists of a sequence of robot end-effector positions. The total cost of a trajectory τ is the sum of costs of obstacles the robot arm collides with, $Cost(\tau) = \sum_{i=1}^{|\tau|} V[\tau_i]$. Here, $|\tau|$ denotes the length of the trajectory τ , and τ_i is the position of the i -th waypoint. In addition, we test another planning method that directly applies A* search without any directional push safety analysis. We refer to it as VLM Cost+A* w/o direction. It only relies on the isotropic costs in the flattened cost map M .

C. Baseline Methods

We evaluate IMPACT against the following baselines.

1) *Collision-Free Planning*: This avoids all collisions, and serves as a baseline to demonstrate that in cluttered

Category	Path Planning Algorithm	Reach Target \uparrow	Path Cost \downarrow	Contact Duration (s) \downarrow	Unsafe Object Displacement (cm) \downarrow	Success Rate \uparrow
Collision Free	RRT	66.75%	-	1.12	6.47	28.75%
	RRT*	61.00%	-	1.25	7.78	26.50%
	A* w/o direction	20.00%	-	8.33	6.51	15.00%
	A*	23.33%	-	5.14	1.93	15.75%
VLM Cost	RRT	62.91%	9.04	1.22	7.84	57.25%
	RRT*	59.50%	9.11	1.26	7.80	50.75%
	A* w/o direction	58.25%	6.71	5.93	3.82	57.50%
	A* (IMPACT)	78.00%	5.93	4.18	2.51	73.75%
-	LAPP	50.00%	-	9.37	9.22	25.00%

TABLE I: Comparison of path planning algorithms and results in PyBullet simulation. We report five quantitative metrics (see Sec. V-D). “Success Rate” only counts the trajectories that reach the target and satisfy thresholds on “Path Cost,” “Contact Duration,” and “Unsafe Object (human-designated) Displacement.” The arrow \uparrow indicates larger values of the metric correspond to better performance, and \downarrow represents the opposite. Collision-free baselines and LAPP [41] do not use VLMs to generate object costs, so they do not have values for “Path Cost” in this table.

environments, a collision-free path may not exist. We test this with A*, A* w/o direction, RRT, and RRT*.

2) *Language-Conditioned Path Planning (LAPP)*: We use LAPP [41] as a strong baseline, because (like IMPACT) it also allows robots to make collisions with specific objects in the environment. LAPP trains a language-conditioned collision function that predicts whether a robot will collide with objects *other* than the one specified in a language instruction (which the robot is permitted to collide with). The collision function has three inputs: (i) CLIP [28] image embeddings of the scene, (ii) CLIP text embeddings of the language instruction (e.g., “can collide with toys”), and (iii) joint configurations of the robot arm. We build directly upon the official open-source LAPP code.

D. Evaluation Metrics

To evaluate the quality of trajectories in our densely cluttered environments, we compute the following metrics in 3D space. The robot’s trajectory terminates when the robot’s end-effector reaches the target, or if a time limit is reached.

- `reach_target`: whether the robot reaches the target object at the end of the trajectory:

$$\text{reach_target} = \|\mathbf{p}'_e - \mathbf{p}'_{o_{\text{targ}}}\| < 0.01\text{m}.$$

- `path_cost`: the sum of VLM-assigned costs of all collided obstacles during execution:

$$\text{path_cost} = \sum_{o \in \mathcal{O} \setminus o_{\text{targ}}} c_o.$$

- `contact_duration`: the sum of duration for which the robot is in contact with any obstacle:

$$\text{contact_duration} = \sum_{o \in \mathcal{O} \setminus o_{\text{targ}}} t_o.$$

- `displacement`: the displacement of each object:

$$\text{displacement}_o = \|\mathbf{p}'_o - \mathbf{p}_o\|, o \in \mathcal{O}.$$

Here, e is the robot end effector and \mathbf{p}'_e is its final 3D position. For each object $o \in \mathcal{O}$, \mathbf{p}_o and \mathbf{p}'_o denote its initial and final 3D position, respectively. The cost of object o is represented by c_o while t_o is the contact duration between the robot and o . After calculating these metrics, we define a trajectory as a success if the following are all

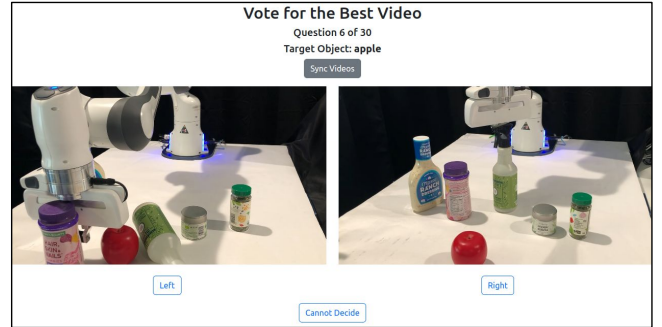


Fig. 4: Our user study evaluation website interface. For each question, the human evaluates two videos of robot trajectories without knowing the underlying robotics method that caused each robot motion. For each video pair, they select which video is more preferable to them. To aid comparisons, we enable the users to sync the videos. We also allow the option of “Cannot Decide.”

true: (i) the robot reaches the target with `path_cost` < 10 , (ii) `contact_duration` $< 100\text{s}$ and (iii) for all unsafe obstacles, we have `displacement` $< 10\text{cm}$. To decide on which objects are unsafe in a given scene, a skilled human annotator pre-selects the 1-2 highest-cost objects in a scene, and those are set for all methods evaluated.

1) *User Study Evaluation*: While the prior quantitative metrics evaluate trajectory cost, they may not fully capture whether a trajectory is “acceptable” to humans. For example, if all collisions with objects are counted and penalized equally, a trajectory where the robot solidly collides with only one object can have a lower cost compared to one where it gently contacts multiple objects. Thus, motion planning algorithms optimize to select the former trajectory, even though the latter trajectory may be more acceptable to humans due to the gentle contacts. Furthermore, tolerance for contact may vary from person to person.

To better assess IMPACT’s ability to generate semantically acceptable behavior, we conduct a user study approved by the IRB office of our institution. We develop a website to collect human feedback, where participants evaluate robot trajectories by watching videos (see Fig. 4). Each question presents two videos of robot trajectories in the same scene with the same target object. One video is generated by an algorithm that utilizes VLM-assigned costs, and the other by the corresponding Collision-Free baseline with the same

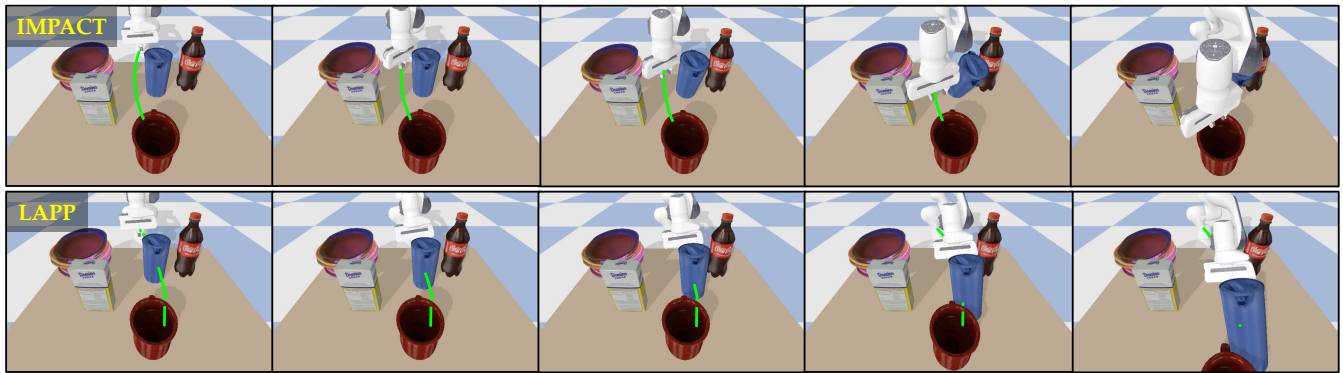


Fig. 5: Examples of trajectories planned by IMPACT (top row) and LAPP (bottom row) in PyBullet simulation [6]. The obstacles are: a coke bottle, a pitcher, a sugar box and a pile of bowls. The target object is the mug behind the obstacles. The planned paths are shown in an overlaid green curve in each image. We also provide LAPP with a language instruction “Can collide with the pitcher and the sugar box.” See Sec. V-E for more details.

planner or by LAPP. Users are not informed which algorithm produced each trajectory. Following a similar approach as Mirjalili et al. [25], participants are asked to select the trajectory they find most acceptable.

E. Simulation Results

Table I reports our simulation results. IMPACT achieves the highest success rate with lower path cost, shorter contact duration, and smaller displacement of unsafe objects (pre-selected by human). Paths planned by Collision-Free baselines can also lead to unexpected collisions. These baselines prioritize avoiding all obstacles, but the robot may lack continuous feasible joint configurations to execute the path and becomes too close to obstacles, increasing the risk of collisions. IMPACT also outperforms the VLM Cost+A* w/o direction, which we attribute to the contact-aware information embedded in the anisotropic cost map. The LAPP baseline has a lower `reach_target` and success rate, mainly because it sometimes fails to find paths or push obstacles along the path.

Fig. 5 shows trajectories planned by different methods in the same scene. IMPACT finds a trajectory that makes contact with the pitcher. LAPP also guides the robot to make contact with the pitcher while avoiding the pile of bowls and the coke bottle, but the pitcher is pushed along the path, preventing the robot from reaching the target. As shown in Table I, this common behavior leads to a higher average unsafe object displacement.

Fig. 6 shows the human evaluation results collected from 25 participants in total. We collect 100 video pairs in simulation and uniformly sample 20 questions for each participant. Each question contrasts a trajectory generated by one method against some baseline. The results show that the method using VLM-assigned costs is the most preferred method across all scenes, suggesting that cost maps produce motion plans that better align with human preferences by leveraging commonsense knowledge in the VLM.

F. Ablation Study

To investigate the benefit of VLM-generated object costs, we conduct an ablation study by setting all object costs to

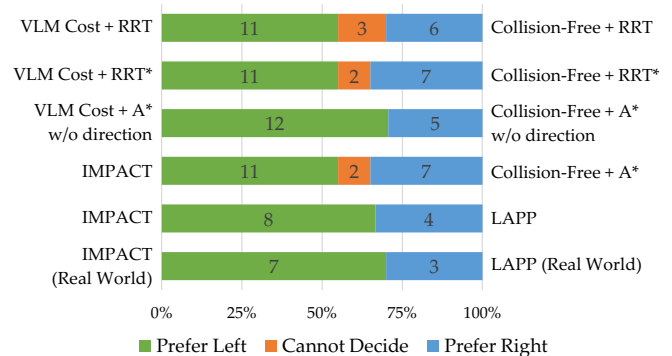


Fig. 6: Human evaluation results. The first five rows report results from simulation, and the last row show the real-world experiments. Each bar represents the results of comparing two methods. It is divided into three segments: the left method, the right method, and “Cannot decide.” Each segment counts the number of questions where people prefer the trajectory generated by the corresponding method. For example, in the top row (VLM Cost+RRT vs. Collision-Free+RRT), more participants prefer the trajectory planned by VLM Cost+RRT in 11 cases out of 20, Collision-Free+RRT in six cases, and have no preference in three cases.

0, so all collisions are allowed during path planning. Results in Table II demonstrate that VLM-assigned costs improve the trajectory success rate (see Sec. V-D) of reaching tasks. For all three motion planning algorithms, the VLM takes properties of obstacles into account and assigns object costs based on different scenes. This encourages planners to avoid obstacles and generate trajectories with reduced contact.

VI. PHYSICAL EXPERIMENTS

A. Experiment Setup

We evaluate IMPACT on a real robotic system to validate our findings from simulation. This does not involve sim-to-real transfer, as we provide real-world images directly to the VLM for IMPACT. The hardware setup consists of a Franka Panda arm with a standard parallel-jaw gripper, and a flat tabletop surface. A human operator arranges multiple objects in close proximity on top of the surface. For reasonably fair comparisons among methods, the human tries to place objects in consistent locations for each scene.

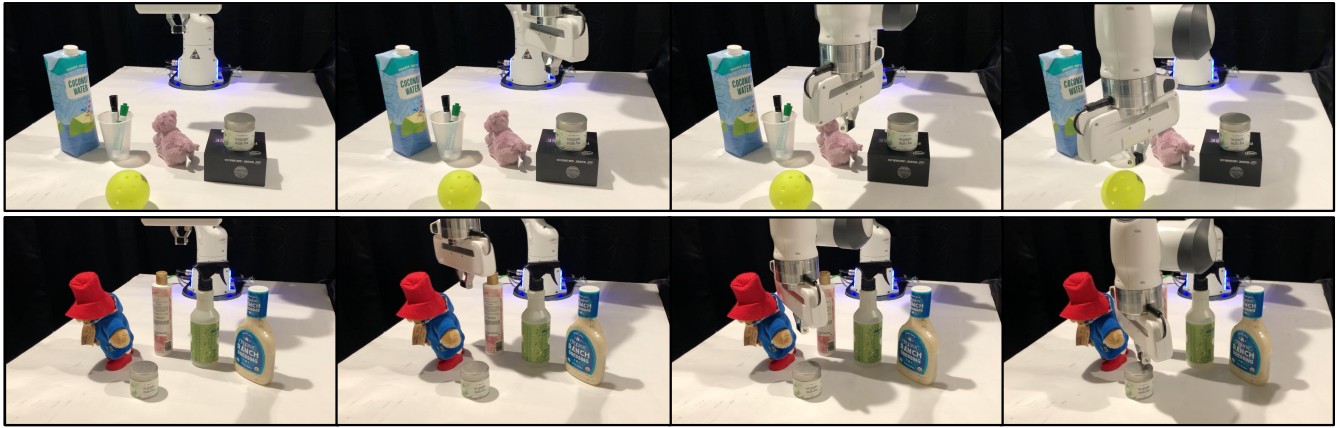


Fig. 7: Examples of successful trajectories from IMPACT in the real world. Top row: the robot reaches the target pickleball while moving closer to the soft toy hippo. Bottom row, the robot rotates its gripper to avoid contact with tall bottles and reaches the target matcha can.

Path Planning Algorithm	Cost	Success Rate
RRT	VLM Cost	57.25%
	Same Cost for All	44.50%
RRT*	VLM Cost	50.75%
	Same Cost for All	42.50%
A*	VLM Cost (IMPACT)	73.75%
	Same Cost for All	65.00%

TABLE II: Results of our ablation study on object costs. “Same cost for all” refers to assigning all object costs to be 0, instead of querying the VLM to generate costs. See Sec. V-F for details.

Method	Success Rate
IMPACT	61%
LAPP	49% (Seen Obj.) 40% (Unseen Obj.)

TABLE III: Results of our method versus LAPP in the real world. Seen objects refer to objects seen during fine-tuning of LAPP while unseen objects are novel objects to LAPP. Our method is zero-shot, so all the objects can be considered unseen to it.

We mount two Intel RealSense D435 cameras on both sides of the scene to capture RGBD images and generate voxel grids. In the real world, we need to get each object’s position to assign the associated object cost to those voxels. Unlike in simulation, we use Grounded SAM 2 [29], [30] to generate the segmented point cloud of each object as we build the cost map. Each image and the object list are provided as input to Grounded SAM 2 to predict the segmentation masks of the objects. We then get a multi-view segmented point cloud and convert it to the cost map. The motion planning part is the same as in simulation.

For physical experiments, we test IMPACT and LAPP. To adapt LAPP for real-world experiments, we fine-tune it on a small dataset containing 25 real-world RGB images, manually annotated with collision scores. The images are captured by an Intel RealSense D435 camera that faces both the robot and the scene. The annotations explicitly label which objects are safe to collide with (for example, “can collide with plastic bottle”). Objects not mentioned in the language prompts are treated as unsafe by default, aligning with LAPP’s methodology of explicit collision rules. Human operators pair each scene with joint configurations and task-



Fig. 8: Examples of real-world failures generated by IMPACT. In the left image, the robot gets stuck on the toy bear while reaching the target glue bottle. In the right image, the robot does not rotate the gripper in time and collides with the ranch bottle.

aligned prompts, reflecting the original work’s use of free-form language constraints [41]. This retains LAPP’s pre-trained reasoning while grounding predictions in real-world spatial relationships and safety priorities. Of the 10 test scenes, 8 use objects seen during fine-tuning, while 2 scene contains novel objects to evaluate generalization.

B. Real World Results

Table III shows real world results of different methods. IMPACT outperforms LAPP on average across all scenes. Furthermore, the performance gap is larger in scenes with unseen objects (61% versus 40% success). The results suggest the strong generalization ability of our method since it does not need sim-to-real transfer. However, LAPP requires fine-tuning on unseen real world objects to achieve comparable results. We also conduct a user study on real-world experiments. The same 25 participants evaluate 10 pairs of real-world trajectories. As shown in the last row of Fig. 6, IMPACT is preferred in more scenarios. Fig. 7 shows two examples of successful trajectories planned by IMPACT.

Failure Cases. While IMPACT generates semantically-acceptable trajectories in most trials, failures can happen during execution. Fig. 8 shows two examples of failed trajectories. The main failure cases are: (i) part of the robot gets stuck on an obstacle, (ii) the gripper collides with an obstacle before or during rotation, and (iii) the VLM predicts object costs that are misaligned with human preferences.

VII. LIMITATIONS

While promising, IMPACT has several limitations that point to interesting directions for future work. First, after

selecting the most semantically-acceptable trajectory, the robot follows it open-loop. This means it cannot react to unexpected disturbances in real-time. Second, IMPACT relies on having relatively complete RGBD observations, and may be less effective under partial observability with severe occlusions. Developing a closed-loop procedure that can actively perceive the environment may address these limitations.

VIII. CONCLUSION

In this work, we introduce IMPACT, a framework for motion planning in clutter that leverages the broad knowledge in Vision-Language Models (VLMs) to assess object contact tolerance. IMPACT represents this information in an anisotropic cost map for motion planning. Our results in simulation and in real-world experiments demonstrate that robots can efficiently reach targets while making semantically-acceptable contact when needed. We hope that this work inspires future work towards flexible and contact-rich robot manipulation in densely cluttered environments.

ACKNOWLEDGMENTS

We thank our colleagues Ebonee Davis, Sicheng He, Ayano Hiranaka, Minjune Hwang, Yunshuang Li, and Qian (Peter) Wang for helpful technical advice and discussions.

REFERENCES

- [1] M. Ahn et al., “Do as i can, not as i say: Grounding language in robotic affordances,” in *Conference on Robot Learning (CoRL)*, 2022.
- [2] L. Brunke et al., “Semantically Safe Robot Manipulation: From Semantic Scene Understanding to Motion Safeguards,” *arXiv preprint arXiv:2410.15185*, 2024.
- [3] B. Calli et al., “Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols,” *IEEE Robotics and Automation Magazine*, 2015.
- [4] N. Chavan-Dafle et al., “Extrinsic Dexterity: In-Hand Manipulation with External Forces,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.
- [5] W. Chow et al., “PhysBench: Benchmarking and Enhancing Vision-Language Models for Physical World Understanding,” in *International Conference on Learning Representations (ICLR)*, 2025.
- [6] E. Coumans and Y. Bai, *PyBullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning*, <http://pybullet.org>, 2016–2020.
- [7] M. R. Dogar and S. S. Srinivasa, “A Framework for Push-grasping in Clutter,” in *Robotics: Science and Systems (RSS)*, 2011.
- [8] D. Driess et al., “PaLM-E: An Embodied Multimodal Language Model,” in *International Conference on Machine Learning (ICML)*, 2023.
- [9] G. T. Google, “Gemini: A Family of Highly Capable Multimodal Models,” *arXiv preprint arXiv:2312.11805*, 2023.
- [10] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
- [11] K. K. Hauser, “Minimum constraint displacement motion planning,” in *Robotics: science and systems*, Berlin, Germany, vol. 6, 2013, p. 2.
- [12] W. Huang et al., “Voxposer: Composable 3d value maps for robotic manipulation with language models,” in *Conference on Robot Learning (CoRL)*, 2023.
- [13] R. Jenamani et al., “Robot-assisted Inside-mouth Bite Transfer using Robust Mouth Perception and Physical Interaction-Aware Control,” in *International Conference on Human-Robot Interaction (HRI)*, 2024.
- [14] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *International Journal of Robotics Research (IJRR)*, 2011.
- [15] L. Kavraki and J.-C. Latombe, “Randomized preprocessing of configuration for fast path planning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 1994.
- [16] M. D. Killpack, A. Kapusta, and C. C. Kemp, “Model predictive control for fast reaching in clutter,” in *Autonomous Robots*, 2016.
- [17] Y. Kuang et al., “RAM: Retrieval-Based Affordance Transfer for Generalizable Zero-Shot Robotic Manipulation,” in *Conference on Robot Learning (CoRL)*, 2024.
- [18] S. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” *Research Report 9811*, 1998.
- [19] M. A. Lee et al., “Making Sense of Vision and Touch: Self-Supervised Learning of Multimodal Representations for Contact-Rich Tasks,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.
- [20] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end Training of Deep Visuomotor Policies,” in *Journal of Machine Learning Research (JMLR)*, 2016.
- [21] J. Liang et al., “Code as policies: Language model programs for embodied control,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.
- [22] F. Liu, K. Fang, P. Abbeel, and S. Levine, “Moka: Open-vocabulary robotic manipulation through mark-based visual prompting,” in *Robotics: Science and Systems (RSS)*, 2024.
- [23] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.
- [24] Y. J. Ma et al., “Eureka: Human-Level Reward Design via Coding Large Language Models,” in *International Conference on Learning Representations (ICLR)*, 2024.
- [25] R. Mirjalili et al., “Lan-grasp: Using large language models for semantic object grasping,” in *International Symposium on Robotics Research (ISRR)*, 2024.
- [26] H. Nemlekar et al., “Robotic lime picking by considering leaves as permeable obstacles,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.
- [27] OpenAI et al., “Gpt-4o system card,” *arXiv preprint arXiv:2410.21276*, 2024.
- [28] A. Radford et al., “Learning transferable visual models from natural language supervision,” in *International Conference on Machine Learning (ICML)*, 2021.
- [29] N. Ravi et al., “Sam 2: Segment anything in images and videos,” *arXiv preprint arXiv:2408.00714*, 2024.
- [30] T. Ren et al., *Grounded sam: Assembling open-world models for diverse visual tasks*, 2024. [arXiv: 2401.14159 \[cs.CV\]](https://arxiv.org/abs/2401.14159).
- [31] G. Schoettler et al., “Deep Reinforcement Learning for Industrial Insertion Tasks with Visual Inputs and Natural Rewards,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.
- [32] J. Schulman et al., “Finding locally optimal, collision-free trajectories with sequential convex optimization,” in *Robotics: Science and Systems (RSS)*, 2013.
- [33] I. Singh et al., “Progprompt: Generating situated robot task plans using large language models,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.
- [34] S. Sontakke et al., “Roboclip: One demonstration is enough to learn robot policies,” in *Neural Information Processing Systems (NeurIPS)*, 2023.
- [35] M. Stilman and J. J. Kuffner, “Navigation among movable obstacles: Real-time reasoning in complex environments,” *International Journal of Humanoid Robotics*, vol. 2, no. 04, pp. 479–503, 2005.
- [36] R. Tedrake, *Robotic Manipulation, Perception, Planning, and Control*. 2025. [Online]. Available: <http://manipulation.mit.edu>.
- [37] A. Thomas, G. Ferro, F. Mastrogiovanni, and M. Robba, “Computational tradeoff in minimum obstacle displacement planning for robot navigation,” *arXiv preprint arXiv:2302.07114*, 2023.
- [38] L. Wang et al., “GenSim: Generating Robotic Simulation Tasks via Large Language Models,” in *International Conference on Learning Representations (ICLR)*, 2024.
- [39] Y. Wang et al., “RL-VLM-F: Reinforcement Learning from Vision Language Foundation Model Feedback,” in *International Conference on Machine Learning (ICML)*, 2024.
- [40] J. Xiang et al., “Structured 3d latents for scalable and versatile 3d generation,” *arXiv preprint arXiv:2412.01506*, 2024.
- [41] A. Xie, Y. Lee, P. Abbeel, and S. James, “Language-conditioned path planning,” in *Conference on Robot Learning (CoRL)*, 2023.
- [42] W. Zhou and D. Held, “Learning to Grasp the Ungraspable with Emergent Extrinsic Dexterity,” in *Conference on Robot Learning (CoRL)*, 2022.