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

Yiyang Ling*, Karan Owalekar*, Oluwatobiloba Adesanya, Erdem Bıyık, Daniel Seita
University of Southern California
*Equal contribution
Correspondence: {lingyiya,kowaleka,seita}@usc.edu

***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). Due to this diversity, it is difficult to characterize which contacts may be acceptable or unacceptable. 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 uses the VLM's outputs to produce a dense 3D "cost map" that encodes contact tolerances and seamlessly integrates with standard motion planners. 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 3620 simulation and 200 real-world trials suggest that IMPACT enables efficient contact-rich motion planning in cluttered settings while outperforming alternative methods and ablations. Supplementary material is available at https://impact-planning.github.io/.**

I. Introduction

Classical motion planning for robot manipulation [30], [42] 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. 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 motivating example in Fig. 1, which shows a robot manipulator making contact with a toy bear to efficiently reach the target salt shaker. Due to the clutter, a collision-free path to the salt shaker 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.

To address this challenge, we propose **I**ntelligent **M**otion **P**lanning with **A**cceptable **C**ontact **T**rajectories (IMPACT), a novel framework that leverages modern Vision-Language Models (VLMs) such as GPT-4o [34] to infer object contact tolerances. After analyzing object properties, IMPACT generates a 3D cost map to represent the tolerance rate of different regions in the scene. This cost map integrates seamlessly with standard off-the-shelf motion planning algorithms to enable trajectories which engage in efficient and semantically-acceptable contact. We pair IMPACT with the RRT* motion planning algorithm [17] as our primary method, and we refer to this combination as IMPACT+RRT*. We perform experiments in simulation and real-world settings, testing a variety of contact-rich, densely cluttered reaching tasks. We evaluate IMPACT+RRT* using multiple quantitative metrics, including human preference rankings from a user study. Our results indicate that users prefer IMPACT+RRT* 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" and generates a 3D cost map to densely represent object tolerance information.
- A motion planning pipeline that integrates contact cost maps with standard motion planning to find semantically-acceptable contact-rich trajectories.
- Simulation and real-world experiments of IMPACT+RRT* for reaching-based tasks in dense clutter, and evaluation of contact-rich trajectories using quantitative task-based and human feedback metrics.

II. Related Work

A. Robot Motion Planning

Classical motion planning algorithms can be broadly characterized as optimization-based or sampling-based. Optimization-based methods, such as TrajOpt [39], frame the problem as minimizing a cost function subject to kinematic, dynamics, and obstacle avoidance constraints. Sampling-based methods, such as PRM [18], RRT [24], and RRT* [17], incrementally build a graph or tree of feasible paths. While there are numerous variants of these methods, a common underlying theme is the constraint of avoiding any collisions. This is generally desirable, but means motion planning under this standard formulation has limitations. In densely cluttered scenarios where contact is inevitable, these motion planners might not find a solution, even though one might exist if

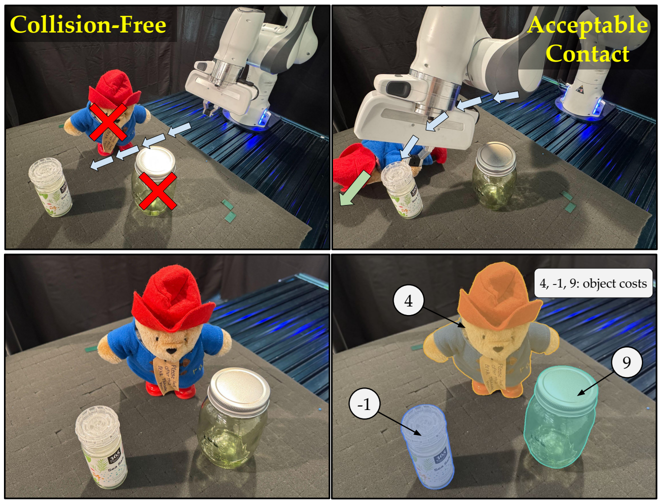


Fig. 1: An example of a reaching task and object costs. The first row shows the difference between collision-free paths and paths with acceptable contact. Left: Collision-free paths prevent a "straight" path to the salt shaker because of the toy bear and the glass bottle obstacles (each marked with a red "X"). Right: With semantically acceptable contact, the robot can successfully reach the salt shaker by pushing the toy bear and avoiding the fragile glass bottle. The second row shows cost of each object generated by GPT-4o. Left: original scene. Right: GPT-4o assigns different costs to objects, with the target assigned -1 (toy bear: 4, salt shaker: -1, glass bottle: 9).

the robot engages in light contact. Our method seamlessly integrates with prior motion planning algorithms for flexible contact-rich manipulation.

Some motion planning works modify cost functions so that certain obstacles, such as leaves, are permeable [20], [33] and thus allow for some contact. However, these methods have been in domain-specific foliage settings and not tested in common cluttered scenarios with rigid objects. In closely-related work, [47] proposes to use semantic language commands to enable semantically-acceptable contact. Unlike [47], we do not require explicit language instructions about which contacts are acceptable, since we utilize VLMs to automate this process.

B. Contact-Rich Robot Manipulation

Dealing with contacts is challenging in manipulation [42]. 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 [25], [26], [38]. Another method for contact-rich problems is extrinsic dexterity [4], [50], 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 [15]. 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 [34] and Gemini [10] are trained on broad Internet-scale data and have remarkable semantic and spatial knowledge. Thus, the research community has explored numerous applications of VLMs in robotics [8], [11], [19]. One way to use VLMs in robotics is to generate high-level task plans, in the form of natural language [1], [7], [12] or executable code [27], [40]. Another way to use VLMs is for reward [31], [41], [49] or task design [13], [43], [44]. VLMs can also aid low-level affordance reasoning [22], [28] which, in an extension of this direction, can involve generating full trajectories [23]. In contrast to these works, we use VLMs for a complementary objective: to infer object contact tolerances to guide contact-rich motion planning.

In closely-related work, VoxPoser [14] exploits VLMs for open-world reasoning and visual grounding to compose a 3D value map to guide robotic interactions. As in [14], we use VLMs to construct a 3D value map (equivalently, a cost map). 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 reaching it may be infeasible with a collision-free trajectory. To reduce occlusions, at least two cameras provide respective RGBD images at the start. Given these image observations, the objective is to compute a trajectory τ for the robot, defined as a sequence of gripper poses, such that its gripper 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 GPT

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

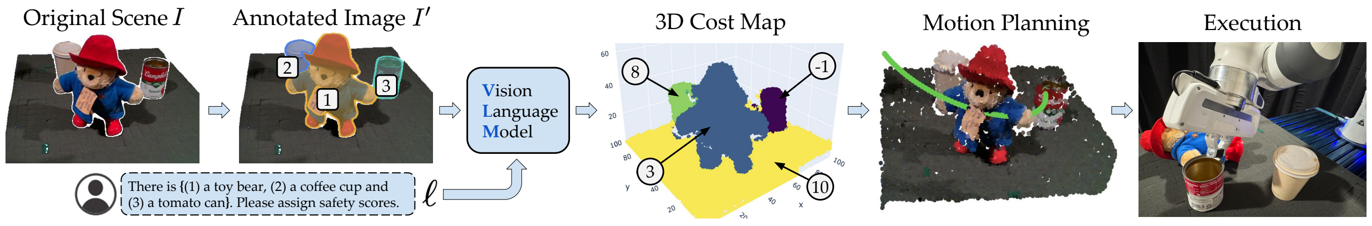


Fig. 2: Overview of IMPACT. There is a toy bear, a coffee cup and a tomato can on the table. The objective is to reach the tomato can. We use SAM2 [36] to segment the image and label the objects using "1," "2," and "3" to assist GPT's visual reasoning. GPT also receives a language template prompt ℓ with object information from SAM2. GPT produces *costs* for the three objects, which are projected into a voxel grid C indicating the cost for the robot end-effector to enter each voxel. The costs are high for the coffee cup (GPT-assigned cost: 8) and the tabletop (a fixed cost of 10). We use a cost of -1 for the target object. Finally, an off-the-shelf motion planner (RRT*) uses this to guide the robot, which avoids the coffee cup but makes contact with the toy bear to successfully reach the tomato can.

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 $\{0, 1, \ldots, 10\}$ as the cost, where a 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 input to GPT includes both an annotated image and a text prompt. To get the image input, we mount an RGBD camera between the scene and the robot to capture a front-view image I in simulation, while in the real world setup, we mount a camera at the right front of the scene to get I. Cluttered scenes may contain too many objects for GPT to accurately recognize all of them. Consequently, we use SAM2 [36] to segment the objects. Then, we use Set-of-Mark prompting [48] to annotate the image with numerical indices over 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. We include Chain-of-Thought prompting [45] in ℓ to improve GPT's reasoning. 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 generated object costs, suggesting that GPT can accurately reason about object properties. For example, it understands that a glass bottle is fragile, and thus assigns it a high cost of 9. In contrast, other objects have a lower cost (i.e., better contact tolerance) such as the toy bear with a cost of 4.

B. *Motion Planning with Contacts*

We use motion planning algorithms to synthesize robot trajectories in the cluttered environment. Inspired by the 3D value map in VoxPoser [14], we propose to construct a 3D cost map C to guide the motion planner. The cost map is represented as a voxel grid of dimension $(L \times W \times H)$, where each voxel $C[x, y, z]$ denotes the cost at position (x, y, z). We initialize C to be all zero. Then, for each object in the scene, we identify the occupied voxels in 3D space and assign them the corresponding object cost from GPT (see Sec. IV-A). During planning, we further assign the cost of the target to -1 instead of using the cost generated by GPT. This encourages the planning algorithm to find a path towards the target. We visualize cost maps in Fig. 2 and Fig. 3.

Once we construct the 3D cost map, we use a standard motion planning algorithm to compute a trajectory that minimizes cost. 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:

$$Cost(\tau) = \sum_{i=1}^{|\tau|} C[\tau_i]. \tag{1}$$

In Equation 1, $|\tau|$ denotes the length of the trajectory τ and τ_i is the position of the i-th waypoint. By combining the cost map and the motion planning algorithm, we can compute a trajectory that minimizes contact with high-cost obstacles while permitting acceptable contact when necessary. Notably, we only use Equation 1 during motion planning. For evaluation, we also compute the cost based on the obstacles the robot arm makes contact with, but we do not count the same obstacle multiple times (see Sec. V-D).

As our main method, we use IMPACT with the RRT* [17] motion planner, and denote this as IMPACT+RRT*.

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

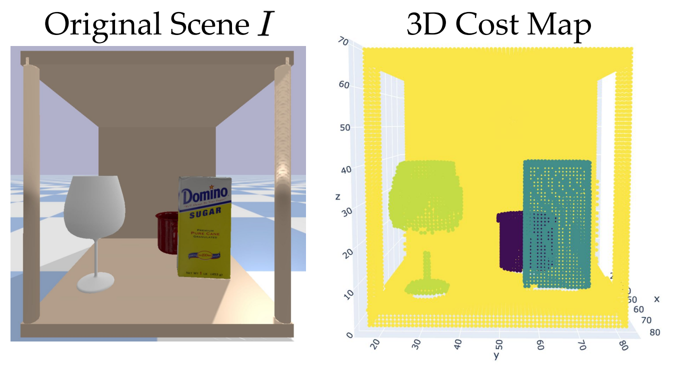


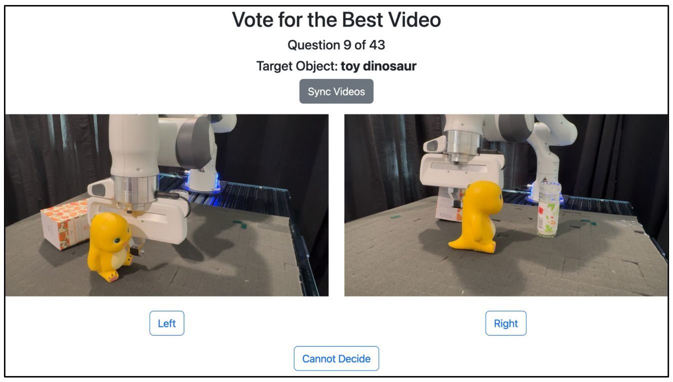
Fig. 3: Our simulation setup in PyBullet simulation [6]. In this example scene, there is a box which contains a wine glass, a sugar box, and a mug. The objective is to reach the mug. The yellow and green regions (e.g., for the wine glass and the larger box) indicate higher costs. For visual clarity, we show the original scene I, before it is annotated for GPT. See Sec. V for more details.

pitchers) from the YCB [3] data and fragile objects (e.g., wine glasses and stacked bowls) using 3D models generated from TRELLIS [46]. TRELLIS allows constructing delicate object meshes from single-view images and introduces fragility constraints not present in YCB objects. All objects are on a shelf so that the robot cannot reach the target object from above. See Fig. 3 for an example scene.

We use three cameras which capture the scene for cost map generation: one in front of the shelf and two on the sides. The camera in front of the shelf also captures the scene image I used as input to GPT. To prevent interactions with the shelf or tabletop, we explicitly assign their costs to be 10. We set the maximum allowed path cost to 10 to balance efficiency in path planning and contact avoidance. To quantify the benefit of using GPT-generated costs, we compare with two control groups: (i) with the maximum path cost set to 0 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 reviewed in Sec. IV-B, our method is IMPACT+RRT*. In simulation, we also test IMPACT with RRT [24] and MPC [9] as the motion planners, and thus call the respective methods IMPACT+RRT and IMPACT+MPC. We slightly modify these planning algorithms to improve their effectiveness for manipulation in dense clutter. For example, to encourage the robot to avoid high-cost obstacles, we define a threshold for object cost. During planning, we check the distance to each obstacle exceeding this cost threshold to make sure the nodes close to them are not considered as waypoints of the trajectory. We apply this modification to both RRT and RRT*. For MPC, the cost of a state is defined as the sum of (i) the distance to the target object, (ii) the cost of the current end effector position decided by the cost map and (iii) a penalty for collisions with high-cost obstacles.


Vote for the Best Video

Question 9 of 43

Target Object: **toy dinosaur**

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 aid comparisons, we enable the users to sync the videos. We also allow the option of "Cannot Decide."

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 environments, a collision-free path may not exist. We test this with MPC, RRT, and RRT*.

2) Language-Conditioned Path Planning (LAPP): We use LAPP [47] 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 [35] 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. 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 any point during the trajectory:

$$\texttt{reach\_target} = \|\mathbf{p}'_e - \mathbf{p}'_{o_{\text{targ}}}\| < 0.01.$$

- `path_cost`: the sum of the cost of all collided obstacles during execution:

$$\texttt{path\_cost} = \sum\nolimits_{o \in \mathcal{O} \setminus o_{\text{targ}}} c_o.$$

Category	**Path Planning Algorithm**	**Reach Target ↑**	**Path Cost ↓**	**Contact Duration (s) ↓**	**High Cost Object Displacement (cm) ↓**	**Success Rate ↑**
Collision Free	MPC	76.93%	-	12.5	8.05	20.75%
	RRT	78.92%	-	9.39	6.36	34.75%
	RRT*	83.12%	-	7.57	6.32	37.75%
IMPACT	MPC	81.45%	11.59	12.0	7.97	41.00%
	RRT	**84.67%**	**8.52**	7.64	6.11	58.75%
	RRT* (Ours)	82.50%	8.85	**7.50**	**6.08**	**63.25%**
-	LAPP	41.18%	-	9.53	10.58	35.00%

TABLE I: Comparison of path planning algorithms and results in PyBullet simulation. We report 5 quantitative metrics (see Sec. V-D). "Reach Target" reports when the robot's end effector reaches the target after executing the trajectory, and where collisions are allowed. "Success Rate" only counts the trajectories that strictly contain no collisions to (human-designated) high-cost obstacles. The arrow ↑ indicates larger values of the metric correspond to better performance, and ↓ represents the opposite. Collision-Free baselines and LAPP do not use GPT to generate object costs, so they do not have values for "Path Cost" in this table.

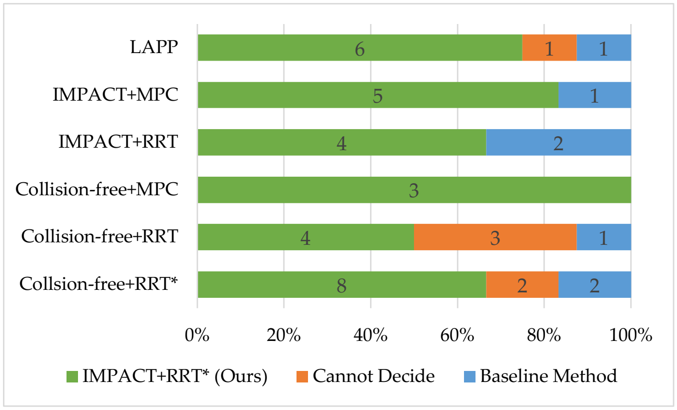


Fig. 5: Human evaluation results of simulation experiments. Each bar represents the results of comparing IMPACT+RRT* (i.e., our method) versus one baseline. It is divided into three segments: IMPACT+RRT*, the baseline method, and "Cannot decide." Each segment counts the number of questions where people prefer the trajectory generated by the corresponding method. For example, the top row reports eight questions that compare the trajectory from LAPP and IMPACT+RRT*. Among six of these questions, more participants prefer the trajectory planned by IMPACT+RRT*. In one question, they prefer the trajectory generated by LAPP. In another question, most people do not have a strong preference.

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

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

- `displacement`: displacement of each object:

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

Here, e is the robot end effector. For each object $o \in \mathcal{O}$, $\mathbf{p}_o$ and $\mathbf{p}'_o$ denote its initial and final position, respectively; the final position considers early termination. 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 true: (i) the robot reaches the target with `path_cost` < 10, (ii) `contact_duration` < 100 and (iii) for all high-cost obstacles, we have `displacement` < 0.1. To decide on which objects are "high-cost" 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 has 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. We develop a website to collect human feedback, where participants evaluate robot trajectories by watching videos. Each question presents two videos of robot trajectories in the same scene with the same target object. One video is generated from our algorithm, and the other by a baseline algorithm. Users are not informed which algorithm produced each trajectory. Following a similar approach as Mirjalili et al. [32], participants are asked to select the trajectory they find most acceptable. See Fig. 4 for a visualization of the website interface. Our study has been approved by the Institutional Review Board (IRB) at the University of Southern California. None of the human evaluators is an author of this paper.

E. Simulation Results

Table I reports our simulation results. IMPACT+RRT* achieves the highest success rate with lower path cost, shorter contact duration, and smaller displacement of high-cost objects. IMPACT significantly improves the success rate of different motion planning algorithms compared to collision-free planning, while `reach_target` is similar. This indicates that IMPACT can guide motion planning algorithms

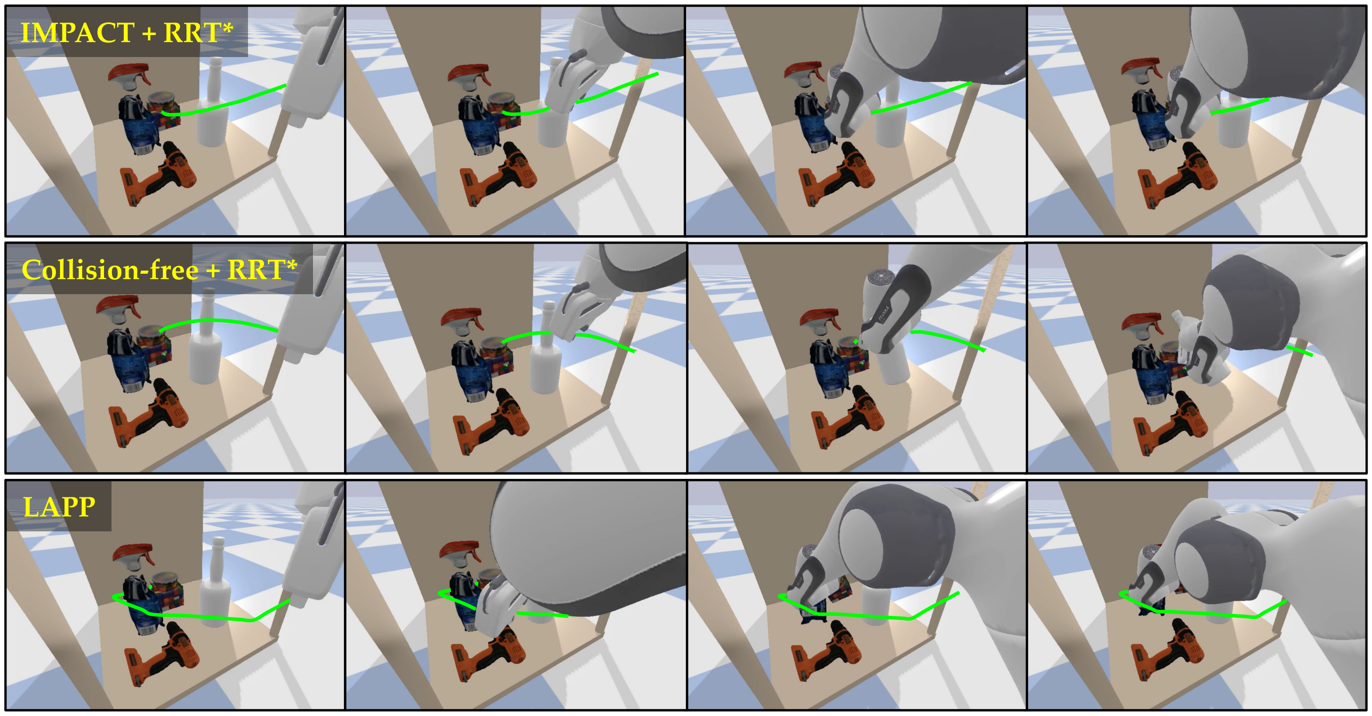


Fig. 6: Examples of trajectories planned using different motion planning algorithms and cost configurations in PyBullet simulation [6]; our method is IMPACT+RRT* (top row). The scene contains three obstacles: a spray bottle, a power drill and a wine glass. The target object is the jar behind the obstacles. The paths planned by different methods are shown in an overlaid green curve in each image. We also provide LAPP with a language instruction "Can collide with the spray bottle and the power drill." See Sec. V-E for more details.

Path Planning Algorithm	Cost	Success Rate
RRT*	IMPACT (Ours)	**63.25%**
	Same Cost for All	42.75%
RRT	IMPACT (Ours)	**58.75%**
	Same Cost for All	45.50%
MPC	IMPACT (Ours)	**41.00%**
	Same Cost for All	39.50%

TABLE II: Results of our ablation study on object costs. "Same cost for all" refers to assigning all object costs the same value 0 instead of querying GPT to generate costs. See Sec. V-F for details.

to plan trajectories with more acceptable contacts. Furthermore, RRT* achieves the best performance compared to other planning methods. The other baseline, LAPP, has a lower `reach_target` rate and success rate, mainly because it fails to find a path in some scenes.

Fig. 6 shows trajectories planned by different methods in the same scene. Our method IMPACT+RRT* finds a trajectory that makes contact with the spray bottle. However, the path planned by Collision-free+RRT* intends to avoid all the obstacles, but the robot collides with the fragile wine glass during execution. This happens because the Collision-Free baseline prioritizes moving through the gap between the obstacles and brings the robot too close to the wine glass, increasing the risk of collision. LAPP also guides the robot to make contact with the spray bottle while avoiding the wine glass, but the spray bottle 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 high-cost object `displacement`.

Fig. 5 shows the human evaluation results of simulation experiments. We collect feedback from 27 participants, each of them answering 43 questions. Each question compares a video pair that contrasts our method with some baseline. The results show that IMPACT+RRT* is the most preferred method across all scenes, suggesting that our method produces motion plans that better align with human preferences by leveraging commonsense knowledge in GPT.

F. Ablation Study

To investigate the benefit of IMPACT, we conduct an ablation study by setting all object costs to 0, so all collisions are allowed during path planning. Results in Table II demonstrate that IMPACT improves the trajectory success rate (see Sec. V-D) of reaching tasks. For all three motion planning algorithms, IMPACT takes properties of obstacles into account and assigns object costs based on different scenes. This encourages motion planning algorithms to avoid obstacles and generate trajectories with reduced contact.

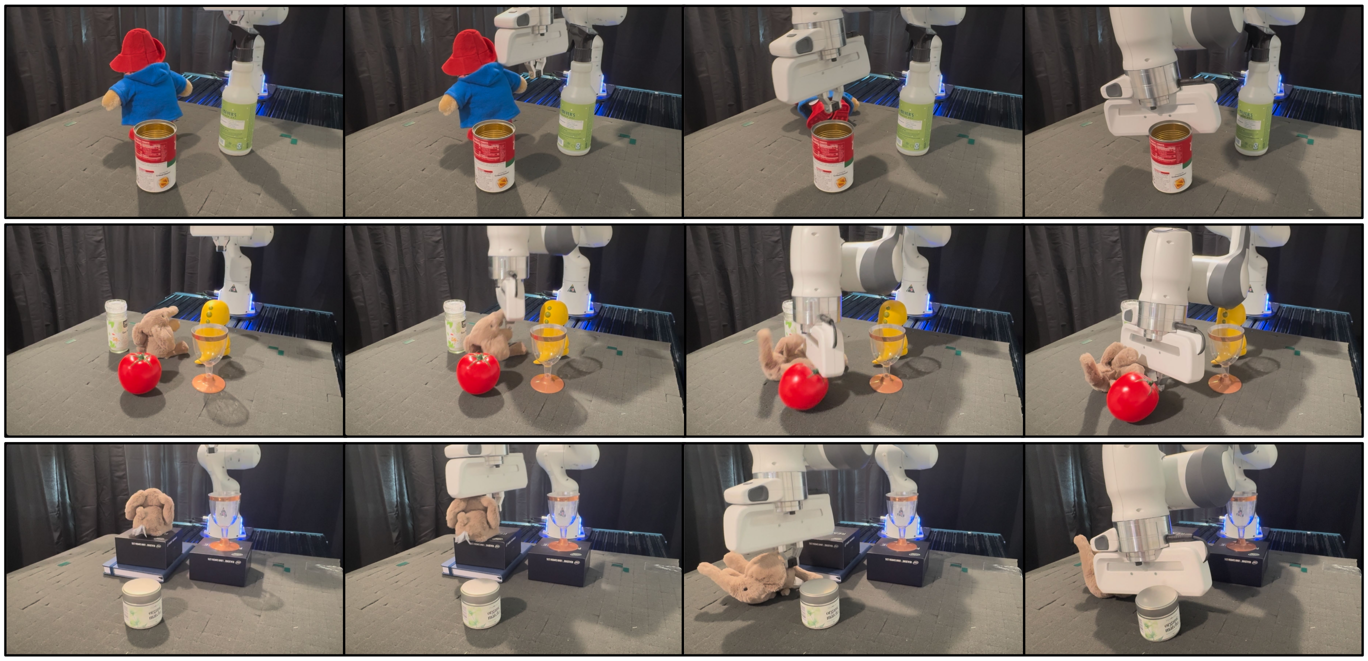


Fig. 7: Examples of successful trajectories with acceptable contact in the real world. The targets in each scene (from top to bottom) are the tomato can, the tomato, and the matcha can, respectively.

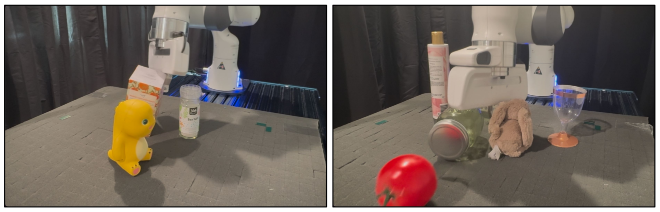


Fig. 8: Examples of real-world failures. In the left image, the robot gets stuck on the box while reaching the target toy dinosaur. In the right image, the robot collides with the glass bottle. The glass bottle hits the target tomato as it falls and the tomato rolls away.

Method	**Success Rate**	
IMPACT+RRT* (Ours)	**63%**	
LAPP	54% (Seen Obj.)	50% (Unseen Obj.)

TABLE III: Results of our method versus LAPP in the real world. Seen objects refers 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.

VI. Physical Experiments

A. Experiment Setup

We evaluate IMPACT on a real robotic system to validate our findings from simulation. This does not involve sim2real transfer, as we provide real-world images directly to GPT for IMPACT. The hardware setup consists of a Franka Panda arm with a standard parallel-jaw gripper, and a flat tabletop surface of size 68 cm×56 cm. 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.

We mount two Intel RealSense L515 cameras on both sides of the scene to capture RGBD images and generate voxel grids. In the real world, we need to get the position of each object to assign the associated object cost to those voxels. Unlike in simulation, we use Grounded SAM 2 [16], [21], [29], [36], [37] to generate the segmented point cloud of each object as we build the 3D cost map. Each image, along with the object list, is provided as input to Grounded SAM 2, to predict the segmentation mask of the objects. We then generate a multi-view segmented point cloud and convert it to the 3D cost map. The motion planning part is the same as in simulation.

For physical experiments, we test IMPACT+RRT* and LAPP. To adapt LAPP for real-world experiments, we fine-tune it on a small dataset containing 20 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-aligned prompts, reflecting the original work’s use of free-form language constraints [47]. This retains LAPP’s pre-trained reasoning while grounding predictions in real-world spatial relationships and safety priorities. Of the 10 test scenes, 9 use objects seen during fine-tuning, while 1 scene contains novel objects to evaluate generalization.

B. Real World Results

Table III shows real world results of different methods. IMPACT+RRT* outperforms LAPP in all scenes. Furthermore, the performance gap is larger in the scene with unseen objects (70% versus 50% success). The results suggest the strong generalization ability of our method since it does not need sim2real transfer. However, LAPP requires fine-tuning on unseen real world objects to achieve comparable results.

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 as follows: (i) part of the robot gets stuck on an obstacle, (ii) the target is displaced when the robot pushes an obstacle, and (iii) GPT 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 to provide to the VLMs. Thus, it 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 cluttered environments that leverages the broad knowledge in vision-language models (VLMs) to assess object contact tolerance. IMPACT represents this information in a 3D 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, A. Brohan, N. Brown, *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, Y. Zhang, R. Römer, *et al.*, "Semantically Safe Robot Manipulation: From Semantic Scene Understanding to Motion Safeguards," *arXiv preprint arXiv:2410.15185*, 2024.

[3] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols," *IEEE Robotics and Automation Magazine*, 2015.

[4] N. Chavan-Dafle, A. Rodriguez, R. Paolini, *et al.*, "Extrinsic Dexterity: In-Hand Manipulation with External Forces," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[5] W. Chow, J. Mao, B. Li, D. Seita, V. Guizilini, and Y. Wang, "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] D. Driess, F. Xia, M. S. M. Sajjadi, *et al.*, "PaLM-E: An Embodied Multimodal Language Model," in *International Conference on Machine Learning (ICML)*, 2023.

[8] R. Firoozi, J. Tucker, S. Tian, *et al.*, "Foundation Models in Robotics: Applications, Challenges, and the Future," *arXiv preprint arXiv:2312.07843*, 2023.

[9] C. E. Garcia, D. M. Prett, and M. Morari, "Model Predictive Control: Theory and Practice - A Survey.," *Autom.*, 1989.

[10] G. T. Google, "Gemini: A Family of Highly Capable Multimodal Models," *arXiv preprint arXiv:2312.11805*, 2023.

[11] Y. Hu, Q. Xie, V. Jain, *et al.*, "Toward General-Purpose Robots via Foundation Models: A Survey and Meta-Analysis," *arXiv preprint arXiv:2312.08782*, 2023.

[12] Y. Hu, F. Lin, T. Zhang, L. Yi, and Y. Gao, "Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning," in *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*.

[13] P. Hua, M. Liu, A. Macaluso, *et al.*, "Gensim2: Scaling robot data generation with multi-modal and reasoning llms," in *Conference on Robot Learning (CoRL)*, 2024.

[14] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," in *Conference on Robot Learning (CoRL)*, 2023.

[15] R. Jenamani, D. Stabile, Z. Liu, A. Anwar, K. Dimitropoulou, and T. Bhattacharjee, "Robot-assisted Inside-mouth Bite Transfer using Robust Mouth Perception and Physical Interaction-Aware Control," in *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2024.

[16] Q. Jiang, F. Li, Z. Zeng, T. Ren, S. Liu, and L. Zhang, *T-rex2: Towards generic object detection via text-visual prompt synergy*, 2024. arXiv: 2403.14610 `[cs.CV]`.

[17] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research (IJRR)*, 2011.

[18] L. Kavraki and J.-C. Latombe, "Randomized preprocessing of configuration for fast path planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1994.

[19] K. Kawaharazuka, T. Matsushima, A. Gambardella, J. Guo, C. Paxton, and A. Zeng, "Real-World Robot Applications of Foundation Models: A Review," *arXiv preprint arXiv:2402.05741*, 2024.

[20] M. D. Killpack, A. Kapusta, and C. C. Kemp, "Model predictive control for fast reaching in clutter," in *Autonomous Robots (AURO)*, 2016.

[21] A. Kirillov, E. Mintun, N. Ravi, *et al.*, "Segment anything," *arXiv:2304.02643*, 2023.

[22] Y. Kuang, J. Ye, H. Geng, *et al.*, "RAM: Retrieval-Based Affordance Transfer for Generalizable Zero-Shot Robotic Manipulation," in *Conference on Robot Learning (CoRL)*, 2024.

[23] T. Kwon, N. D. Palo, and E. Johns, "Language Models as Zero-Shot Trajectory Generators," in *IEEE Robotics and Automation Letters (RA-L)*, 2024.

[24] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[25] M. A. Lee, Y. Zhu, K. Srinivasan, *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.

[26] 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.

[27] J. Liang, W. Huang, F. Xia, *et al.*, "Code as policies: Language model programs for embodied control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[28] 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.

[29] S. Liu, Z. Zeng, T. Ren, *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[30] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

[31] Y. J. Ma, W. Liang, G. Wang, *et al.*, "Eureka: Human-level reward design via coding large language models,"

in *International Conference on Learning Representations (ICLR)*, 2024.

[32] R. Mirjalili, M. Krawez, S. Silenzi, Y. Blei, and W. Burgard, "Lan-grasp: Using large language models for semantic object grasping," in *International Symposium on Robotics Research (ISRR)*, 2024.

[33] H. Nemlekar, Z. Liu, S. Kothawade, S. Niyaz, B. Raghavan, and S. Nikolaidis, "Robotic lime picking by considering leaves as permeable obstacles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[34] OpenAI, A. Hurst, A. Lerer, *et al.*, "Gpt-4o system card," *arXiv preprint arXiv:2410.21276*, 2024.

[35] A. Radford, J. W. Kim, C. Hallacy, *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning (ICML)*, 2021.

[36] N. Ravi, V. Gabeur, Y.-T. Hu, *et al.*, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024.

[37] T. Ren, S. Liu, A. Zeng, *et al.*, *Grounded sam: Assembling open-world models for diverse visual tasks*, 2024. arXiv: 2401.14159 [cs.CV].

[38] G. Schoettler, A. Nair, J. Luo, *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.

[39] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization.," in *Robotics: Science and Systems (RSS)*, 2013.

[40] I. Singh, V. Blukis, A. Mousavian, *et al.*, "Progprompt: Generating situated robot task plans using large language models," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[41] S. Sontakke, J. Zhang, S. Arnold, *et al.*, "Roboclip: One demonstration is enough to learn robot policies," in *Neural Information Processing Systems (NeurIPS)*, 2023.

[42] R. Tedrake, *Robotic Manipulation, Perception, Planning, and Control*. 2023. [Online]. Available: http://manipulation.mit.edu.

[43] L. Wang, Y. Ling, Z. Yuan, *et al.*, "GenSim: Generating robotic simulation tasks via large language models," in *International Conference on Learning Representations (ICLR)*, 2024.

[44] Y. Wang, Z. Sun, J. Zhang, *et al.*, "RL-VLM-F: Reinforcement Learning from Vision Language Foundation Model Feedback," in *International Conference on Machine Learning (ICML)*, 2024.

[45] J. Wei, X. Wang, D. Schuurmans, *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[46] J. Xiang, Z. Lv, S. Xu, *et al.*, "Structured 3d latents for scalable and versatile 3d generation," *arXiv preprint arXiv:2412.01506*, 2024.

[47] A. Xie, Y. Lee, P. Abbeel, and S. James, "Language-conditioned path planning," in *Conference on Robot Learning (CoRL)*, 2023.

[48] J. Yang, H. Zhang, F. Li, X. Zou, C. Li, and J. Gao, "Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v," *arXiv preprint arXiv:2310.11441*, 2023.

[49] W. Yu, N. Gileadi, C. Fu, *et al.*, "Language to rewards for robotic skill synthesis," in *Conference on Robot Learning (CoRL)*, 2023.

[50] W. Zhou and D. Held, "Learning to Grasp the Ungraspable with Emergent Extrinsic Dexterity," in *Conference on Robot Learning (CoRL)*, 2022.