Language-Conditioned Path Planning

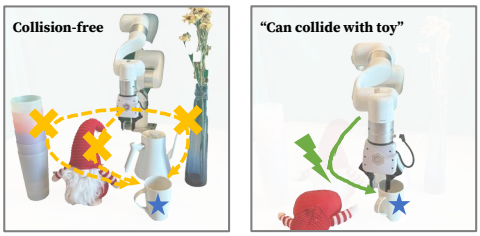
Anonymous Author(s)
Affiliation
Address
email

Abstract: Contact is at the core of robotic manipulation. At times, it is desired (e.g. manipulation and grasping), and at times, it is harmful (e.g. when avoiding obstacles). However, traditional path planning algorithms focus solely on collision-free paths, limiting their applicability in contact-rich tasks. To address this limitation, we propose the domain of Language-Conditioned Path Planning, where contact-awareness is incorporated into the path planning problem. As a first step in this domain, we propose Language-Conditioned Collision Functions (LACO), a novel approach that learns a collision function using only a single-view image, language prompt, and robot configuration. LACO predicts collisions between the robot and the environment, enabling flexible, conditional path planning without the need for manual object annotations, point cloud data, or ground-truth object meshes. In both simulation and the real world, we demonstrate that LACO can facilitate complex, nuanced path plans that allow for interaction with objects that are safe to collide, rather than prohibiting any collision.[1]

Keywords: Robotic Manipulation, Path Planning, Collision Avoidance, Learned Collision Function

1 Introduction

Collision checking is a fundamental aspect of path planning in robotics [1, 2, 3, 4, 5, 6, 7], aiming to find a path between initial and target robot configurations that avoids collisions with the environment. However, traditional collision-free path planning approaches fall short in scenarios where contact with the environment is necessary, such as when manipulating objects or interacting with the surroundings. In such cases, the strict "collision-free" constraint becomes impractical and inhibits the robot's ability to perform tasks effectively.



(a) w/o language (b) w/ language

Figure 1: Language-conditioned path planning enables finding a path, e.g., toward a cup, with acceptable collisions, e.g., a plush toy **(right)**, whereas typical path planning fails at finding a collision-free path **(left)**.

Traditional approaches for enabling contact in path planning [8, 9] often require manual adjustments, such as disabling collision checking for specific objects. However, these approaches rely on access to object state information, ground-truth object meshes, or extensive engineering efforts for each execution. This poses significant challenges, particularly in vision-based contact-rich robotic manipulation tasks.

To overcome this limitation, we propose the domain of **Language-Conditioned Path Planning (LAPP)**, which integrates contact-awareness into the path planning problem. In this domain, path planning is not solely concerned with avoiding collisions but also incorporates the ability to make

[1] Simulation and real-world videos: https://sites.google.com/view/lang-path-planning/home

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

informed decisions about contact with the environment. This enables robots to perform complex manipulation tasks that involve controlled interactions, such as holding a cup or opening a door. Figure 1 provides an illustration of a typical scenario where a robot encounters multiple obstacles and needs to interact with the environment.

To facilitate flexible and adaptive contact-aware path planning, we propose **Language-Conditioned Collision Functions (LACO)** as an initial step in the language-conditioned path planning domain. LACO learns to predict a collision based on a single-view image, language prompt, and robot configuration. By predicting collisions between the robot and the environment modulated by the language prompt, LACO enables the generation of path plans that can handle both desired and controlled collisions without requiring manual object annotations, point cloud data, or ground-truth object meshes. This approach empowers robots to interact with objects that are safe to collide, rather than rigidly avoiding all collisions.

In summary, our main contributions are threefold:

- We propose a novel domain of language-conditioned path planning that integrates semantic language commands to enhance the robot's understanding of how and what to interact with in the environment. By fusing language instructions, we enable more intelligent and context-aware planning.
- To enable language-conditioned path planning, we introduce Language-Conditioned Collision Functions (LACO), a collision function that incorporates language prompts to modulate collision predictions. LACO utilizes only a single-view camera, eliminating the need for object states or point clouds. This allows for easier application in real-world scenarios and facilitates zero-shot generalization to new language commands.
- We provide comprehensive demonstrations of LACO's effectiveness in various path planning tasks. Our experiments include simulations and real-world scenarios, highlighting the practicality and robustness of our language-conditioned collision function.

2 Related Work

Path planning [1, 2, 3, 4, 5, 6, 7] finds a collision-free path between initial and target task configurations – representing robot states and potentially environment states – by querying a collision function with states along a path. However, these collision-free path planning algorithms struggle at handling scenarios where collisions with the environment is necessary or desired. In this paper, we propose language-conditioned path planning, which aims to find a contact-aware path following a language prompt.

The recent success in **large language models** enables training multi-task robotic policies guided by a language instruction [10, 11] and high-level planning using a large language model [12, 13]. These language-conditioned "task" planning / solving approaches differ from our proposed language-conditioned "path" planning domain in that they directly solve a specific set of tasks while our language-conditioned path planning is agnostic to downstream tasks and can be used as a building block for many robotics tasks.

A **collision function** is a fundamental component of path planning in robotics. However, the collision function is assumed to be manually modelled or computed using state estimation. Instead of hand-engineering a collision function, many recent work have proposed to learn end-to-end motion planners [14, 15, 16] or collision functions [17, 18, 19] from synthetically generated data in simulation. To understand 3D configuration of environments, these approaches use point clouds to represent the scenes. In this paper, we propose to learn a collision function from only a single camera input, not requiring depth sensing nor precise camera calibration, which makes our method easily applied to many real-world applications. More importantly, our learned collision function conditions on language describing which objects to collide or not to collide, allowing acceptable or desired collisions for path planning.

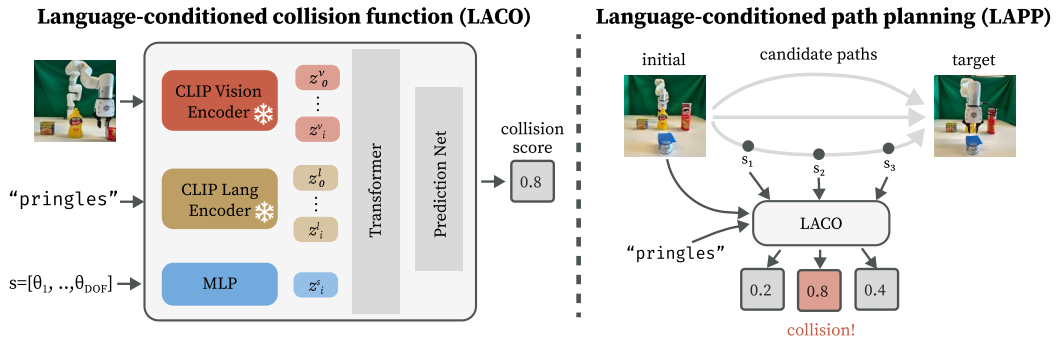


Figure 2: A language-conditioned collision function (LACO), $C(o, s, l)$, predicts whether a robot in a state, s, collides with objects other than collidable objects described in a language prompt, l, in a scene, o, e.g., $C(o, s, \texttt{'pringles'}) = 0$. Then, a path planning algorithm asks LACO to check whether any waypoint s_i of a path has collision objects except for the ones described in l.

3 Language-Conditioned Path Planning

In a cluttered real-world environment, a collision-free path can be highly sub-optimal or impossible to find. We introduce the problem domain of Language-Conditioned Path Planning, which extends traditional path planning to allow safe or desired collisions in a path via language, in Section 3.1. Furthermore, we present a proof-of-concept framework for language-conditioned path planning that first learns a language-conditioned collision function (Section 3.2) and leverages this learned collision function for language-conditioned path planning (Section 3.3).

3.1 Language-Conditioned Path Planning (LAPP)

In robotics, a path planning problem is about finding a connected, collision-free path of robot configurations (i.e. waypoints), $\mathbf{p} = (s_0, s_1, \ldots, s_g)$, starting from an initial configuration, s_0, to a final configuration, s_g, such that every configuration in the path has no collision with the environment, $\forall s_t \in \mathbf{p}, C(o_t, s_t) = 0$. o_t and s_t denote the environment and robot configurations at time t, respectively, and $C(o, s)$ denotes a collision function that outputs 1 if the robot configuration s has collision with the environment, o, and 0, otherwise.

In this paper, we propose a *Language-Conditioned Path Planning (LAPP)* problem, which relieves the strict collision-free constraint in path planning so that path planning can manage safe or desired contacts with the world, *especially guided by language*. LAPP can be formulated as finding a path, $\mathbf{p}$, between two configurations, s_0 and s_g, such that $\forall s_t \in \mathbf{p}, C(o_t, s_t, l) = 0$, where l is a language prompt modulating what needs to be considered as acceptable or desired collisions. For example, a language prompt, l, can be "a robot can collide with plush toys" to specify safe-to-collide objects, as illustrated in Figure 1, or "a robot can grasp a mug" to support contact-rich tasks.

3.2 Language-Conditioned Collision Function (LACO)

For language-conditioned path planning, a path planning algorithm should understand which collisions are acceptable and which must be avoided described in language. In this paper, we address this problem by adapting a collision function, $C(o, s)$, into a *language-conditioned collision function (LACO)*, $C(o, s, l)$, which takes a language prompt, l, into account. Specifically, LACO learns a collision function $C(o, s, l)$, where o is a single-view image observation of the environment, s is a queried robot joint state, and l is a language instruction corresponding to objects that allow for collisions. Note that a joint state, s, does not need to correspond to o; o is representing an environment configuration, which is agnostic to robot joint states, so that it can be used across multiple robot configurations, s.

(a) Training objects (b) Held-out objects (c) Real-world objects

Figure 3: We use ShapeNet objects for our simulated experiments. (a) The training dataset of ShapeNet objects includes: airplane, chair, pot, vessel, laptop, bus, cap, and bottle. (b) The held-out evaluation dataset includes: basket, mug, train, bag, and can. (c) We also perform real-world experiments with YCB objects: Spam, Cheez-it, Pringles, Windex, mustard, and bleach.

We train $C(o, s, l)$ using a dataset of $\mathcal{D} = \{(o, s, l, y^l, y)\}$, where y indicates whether the robot state, s, has any collision in the scene, o, (1) or not (0); and y^l indicates whether there is a undesirable contact under the language instruction, l. We optimize the cross-entropy loss for language-modulated collision prediction (target y^l) together with an auxiliary loss of predicting collisions regardless of language conditioning (target y):

$$\mathcal{L}_l = -\,\mathbb{E}_{(o,s,l,y^l,y)\sim\mathcal{D}}\Big[CE\big(y^l, C(o,s,l)\big) + CE\big(y, C_{\text{aux}}(o,s,l)\big)\Big], \tag{1}$$

where $C_{\text{aux}}(o, s, l)$ is an additional MLP head attached to the last layer of $C(o, s, l)$.

To take advantage of large vision-language models pretrained on a large corpus [20, 21], we use the vision and language encoders of CLIP [22] as the backbone networks for LACO. As illustrated in Figure 2, we tokenize an input image of size 256×256, o, with a frozen pretrained CLIP ViT encoder and a language prompt, l, with a frozen pretrained CLIP language model. We then get 197 visual tokens and 197 language tokens. For a robot state, s, we use an MLP to embed it to a single state token. All these tokens are fed into a transformer and the average of the transformer output tokens is used to predict the collision probability with an MLP. The hyperparameters for our model can be found in Appendix, Table 7.

3.3 Path Planning using LACO

Finally, we implement a language-conditioned path planning algorithm by simply replacing a collision checker in any path planning algorithm with our language-conditioned collision function (LACO). In this paper, we implement LAPP using a sampling-based method, LAPP-TrajOpt [7]. We can additionally integrate a sampling base method, LAPP-RRT* [6], or any other algorithms, as LACO is flexible to various styles of path planning. We use a custom implementation of TrajOpt [7]. Hyperparameters for TrajOpt are in Appendix, Table 9.

4 Experiments

In this paper, we introduce language-conditioned path planning (LAPP), and present a framework that combines existing path planning algorithms and our proposed language-conditioned collision function (LACO) as an initial step. Our evaluations are twofold: (1) we present a thorough investigation of LACO's performance in object- and language-level generalization (Section 4.3), and (2) we showcase the potential of LACO in the language-conditioned path planning domain (Section 4.5).

4.1 Environment Setups

For experiments, we use a XArm7, a low-cost 7-DOF robotic arm, and an Intel RealSense D435 camera, as shown in Figure 4. For the real world, we use 6 YCB objects [23], and for simulation, we

use ShapeNet v2 [24], which provides a taxonomy of diverse, realistic set of 3D meshes with their labels, as shown in Figure 3.

4.2 Data Collection

To train a language-conditioned collision function, we first collect data in both the simulation and real-world environments.

Figure 4: The real-world (left) and simulated (right) environments.

Simulation dataset. For the simulation data, we use PyRep [25] based on CoppeliaSim [26] to synthetically generate a diverse, language-annotated dataset. Each scene includes 2-5 randomly chosen objects in random poses on the table. Instead of randomly initializing a robot pose, a set of robot poses are generated by running CoppeliaSim's built-in RRT* motion planner for each scene. These smooth trajectories bias the dataset toward joint states likely to be queried by a path planner. Next, to generate language annotations for a scene with N objects, we sample combinations of $0, .., N-1$ objects to mask out. We recalculate collisions and procedurally generate a language annotation based on the ShapeNet name for each object. We generated 5000 unique scenes, which consist of unique combinations and positions of objects. Each scene contains about 40 joint states and 10 language annotations.

Real-world dataset. Learning a collision function from real-world data poses additional challenges: notably, the increased visual complexity of real images and the difficulty of collecting collision data. To address these issues, we train on a dataset collected from a domain-randomized twin simulator environment and then finetuning LACO on a small real-world dataset.

We collect data from 20 real-world scenes and 500 domain-randomized simulation scenes. For each scene, we extract 20-30 images using domain randomization in simulation, or camera perturbations in the real world. Similar to our simulated results, we vary the number of objects from 3 to 5 in each scene and vary positions of objects. We restrict our state to the (x, y)-position of the end-effector and the objects to 6 YCB [23] objects: bleach, Cheez-It crackers, Windex spray, Pringles, Spam, and mustard.

4.3 Collision Prediction Results Across Language Conditioning

LACO possess the ability to be modulated by language. In particular, any number of objects in the scene can be included in the language condition, allowing for flexibility in path planning. This ability is not found in built-in collision checkers or learned collision checkers, such as SceneCollisionNet [17], which are agnostic to desired and undesired collisions.

Table 1: We evaluate the language-conditioned collision prediction accuracy of LACO in simulation. For reference, we also evaluate SceneCollisionNet [17], which does not support language conditioning and "Built-In Sim Collision" refers to the ground-truth collision checker.

	Accuracy Per # Conditioned Objs (%)				
Method	0	1	2	3	4
Built-In Sim Collision	100.0	-	-	-	-
SceneCollisionNet	67.0	-	-	-	-
LACO	82.9	78.9	77.18	82.6	72.2

We evaluate the performance of LACO across different number of objects included in the language condition in Table 1. We find that LACO is robust to different numbers of conditioned objects, though it performs best when not conditioned on any objects. In this special case, LACO becomes a simple collision checker, without need to understand the semantics of the environment objects.

Even in the unconditional case, our method outperforms SceneCollisionNet, a point-cloud based learned collision function. We use the public implementation of SceneCollisionNet, which is trained

on a different simulator dataset. This distribution shift may be a reason for its poor performance in our environment. While our primary contribution is presenting a new paradigm of collision checking and path planning, the accuracy of our RGB-only method shows promise of collision checking without extensive camera setups.

4.4 Ablation Studies

Pre-trained observation encoder. We investigate the benefit of using pre-trained observation encoder by comparing: a CLIP pretrained vision encoder and a CLIP enconder trained from scratch. In Table 2, we find that a pretrained CLIP encoder consistently outperforms the one trained from scratch.

Table 2: Examines the benefits of using pre-trained observation encoders

	Accuracy Per # Conditioned Objs (%)				
Method	0	1	2	3	4
LACO	82.9	78.9	77.2	82.6	72.2
From scratch	82.1	72.7	77.7	75.3	71.1

Generalization to unseen objects. One advantage of LACO is its ability to be modulated by language, which is flexible, abstract, and simple. We may expect that with its pretrained vision-language backbone, LACO may generalize to objects and language that are unseen in the collision training dataset. To evaluate this, we include the collision accuracy for conditioning on one object. This allows for evaluation based on whether the language condition, which is standard ShapeNet class name, is successfully used for collision predictions. In Table 3, we present results across conditioning on seen and unseen objects across seen and unseen classes. The unseen classes have class names unseen during training; thus, increasing the difficulty of the task. Table 4 confirms that LACO generalizes to a language prompt not seen during training.

Table 3: Examines the generalization to new objects.

	Seen Class		**Unseen Class**
	Seen Object	Unseen Object	Unseen Object
LACO	78.9	70.6	54.7

Generalization to unseen language. We evaluate generalization to different language using seen objects but making an instruction with different names following these styles:

- **Synonym:** We use unseen synonyms or phrases describing the object, e.g., `beanie` for `hat`.
- **Description**: We use more complex phrases that describe the object. These phrases may not necessarily directly reference the object, e.g., `head-covering accessory` for `hat`.
- **Color** We choose a positive and negative language with the correct and incorrect color.

In Table 4, we find strong generalization for color, where the language references the seen class name. Generalization to descriptions is also strong. One hypothesis is that these descriptions, which are typically longer and contain many keywords relating to the object, may be more informative than just a synonym. Short language conditions like ”cap,” furthermore, may even be ambiguous, as ”cap” may refer to a bottle cap, a hat, or more. This suggests promise in future work of exploring stronger and more descriptive annotations, as we limit our language conditions to ShapeNet class names.

Table 4: Investigates the generalization to language.

	Default Name	Synonym	Descriptive Phrase	Color
LACO	78.9	63.9	71.4	77.0

Dataset size. In the previous experiments, we find that LACO is able to generalize to unseen objects of seen classes. This implies that training on a variety of objects leads to understanding of what each object class implies. However, there is a dip in accuracy for objects from unseen classes. This may arise because we are training on a limited dataset of objects. As the quality and quantity of 3D assets increases, we may expect improved performance by training on more than a limited set of classes. We verify our hypothesis by varying the dataset size in Table 5. The results show that there is an improvement with the increased size of the dataset, but the improvement is marginal.

Table 5: The dataset size ablation examines how performance varies as we alter the size of the dataset.

	Accuracy Per # Conditioned Objs (%)				
Dataset Size	0	1	2	3	4
50%	82.9	74.2	73.4	78.0	73.6
80%	81.2	70.0	72.6	72.2	65.8
100%	82.9	78.9	77.18	82.6	72.2

4.5 Path Planning Demonstrations

In this section, we showcase three tasks using language-conditioned path planning with LACO:

1. **Reach Target**: The objective of this task is to reach a target joint pose while avoiding obstacles. We do not condition on language, and this task resembles a traditional motion planning task.
2. **Reach Target (1 Lang)**: The objective is likewise to reach a target. However, 1 object is masked out, allowing for more flexibility in plans.
3. **Push Object (1 Lang)**: The objective is to push an object forward. For this task, collisions are in fact desired, showcasing the usefulness of LACO.

We report results in Table 6. The push object task is particularly well-suited for language-conditioned motion planning: TrajOpt can be initialized with a trajectory passing through the object and optimize collison constraints with other objects. Reaching targets and pushing objects can also be composed to perform more complex tasks, such as avoiding all obstacles before reaching the object the arm needs to push.

Table 6: Downstream task success rates.

	No Lang	1 Lang	
	Reach Target	Reach Target	Push Object
LAPP-TrajOpt	2/3	2/3	4/5

4.6 Real-World Results

We show real-world trajectories with LAPP-TrajOpt in Figure 5. Our LAPP-TrajOpt with the LACO model, pretrained on simulator data and finetuned with real-world data, is able to find plans in cluttered environments, using the language condition to discover a path that would otherwise be regarded as a trajectory with collisions.

5 Limitations

While LACO offers promising advancements in contact-aware path planning, there are several limitations that should be acknowledged:

Lack of Environment Dynamics. LACO does not currently explicitly consider environment dynamics. Once an object is hit, it may react by being pushed or knocked down, potentially affecting the configuration and positions of other nearby objects. This limitation restricts the ability of LACO to handle dynamic environments and may lead to suboptimal or unsafe motion plans in scenarios where object movements are significant.

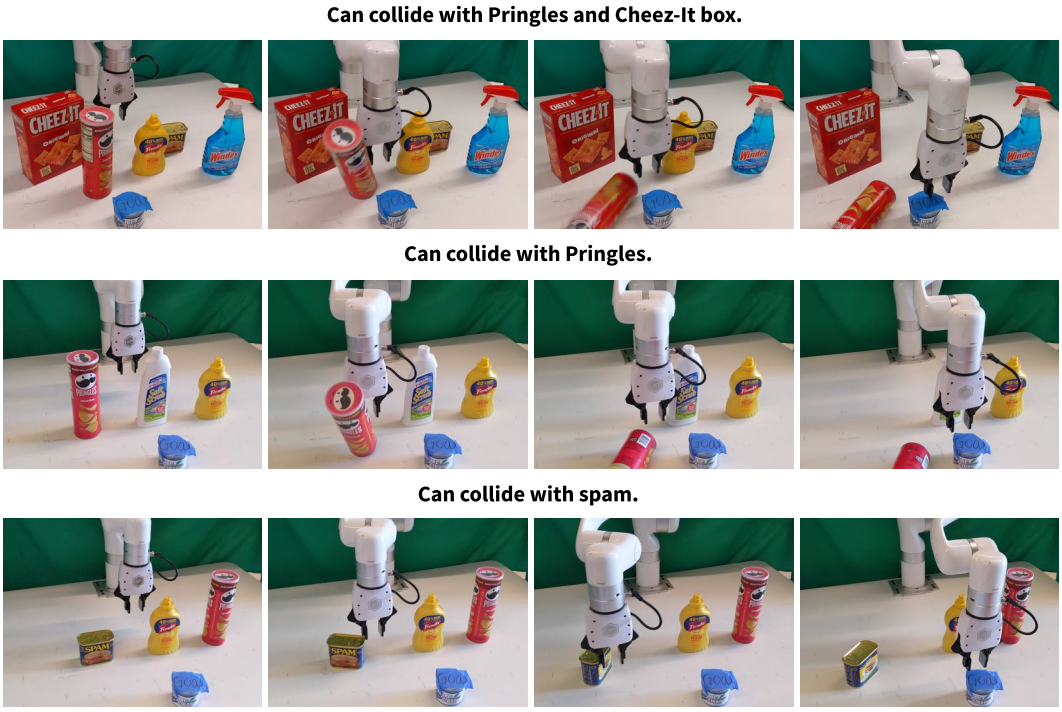


Figure 5: We demonstrate successful execution of the LAPP-TrajOpt path planning algorithm in the real-robot system. A path to the goal, specified by the blue jar, is blocked by diverse objects on the table, so the robot needs to collide with some object. We inform which object is safe and desired to collide using the language prompt above each of the three rollouts.

Limited Language Prompt Scope. The language prompt is limited to specifying objects that are desirable or safe to collide. While this provides valuable control over contact conditions, the current scope of language prompts may not cover the full range of instructions or interactions that a user may desire. Expanding the language prompt capabilities to include a wider variety of language instructions and specifications could enhance the versatility and adaptability of LACO.

Data Generation Efforts. The training process for LACO relies on a combination of synthetic data generated from simulations and manually collected data from the real world. Both approaches require some human engineering and labeling efforts, which can be time-consuming. Exploring advances in 3D asset availability [27] and sim2real [28, 29, 30, 31] techniques could alleviate this limitation and enable more efficient training of LACO using diverse and representative datasets.

6 Conclusion

In conclusion, our proposed domain of Language-conditioned Path Planning addresses the limitations of traditional collision-free path planning in contact-rich robotic manipulation tasks. By integrating contact awareness into path planning, it allows robots to make informed decisions about contact with the environment, enabling them to perform complex manipulation tasks effectively.

Language-Conditioned Collision Functions (LACO) serves as a first step towards Language-conditioned Path Planning, by learning to predict collisions based on visual input, language prompts, and robot configurations. This learned collision function eliminates the need for manual object annotations, point cloud data, or ground-truth object meshes, enabling flexible and adaptive path planning that incorporates both desired and controlled collisions. While the approach shows promise, there are areas for further improvement.

References

[1] M. Overmars. A random approach to motion planning. Technical Report RUU-CS-92-32, Department of Computer Science, Utrecht University, 1992.

[2] L. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration for fast path planning. In *IEEE International Conference on Robotics and Automation*, 1994.

[3] N. M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *IEEE International Conference on Robotics and Automation*, 1996.

[4] S. M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Iowa State University, 1998.

[5] J. J. Kuffner and S. M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 995–1001, 2000.

[6] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[7] 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*, 2013.

[8] F. Burget, A. Hornung, and M. Bennewitz. Whole-body motion planning for manipulation of articulated objects. In *IEEE International Conference on Robotics and Automation*, pages 1656–1662, 2013.

[9] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *IEEE International Conference on Robotics and Automation*, pages 639–646, 2014.

[10] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, 2021.

[11] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002, 2021.

[12] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, 2022.

[13] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022.

[14] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip. Motion planning networks. In *IEEE International Conference on Robotics and Automation*, pages 2118–2124, 2019.

[15] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip. Motion planning networks: Bridging the gap between learning-based and classical motion planners. *IEEE Transactions on Robotics*, 37 (1):48–66, 2020.

[16] H. Ha, J. Xu, and S. Song. Learning a decentralized multi-arm motion planner. In *Conference on Robot Learning*, pages 103–114, 2021.

[17] M. Danielczuk, A. Mousavian, C. Eppner, and D. Fox. Object rearrangement using learned implicit collision functions. In *IEEE International Conference on Robotics and Automation*, pages 6010–6017, 2021.

[18] A. Murali, A. Mousavian, C. Eppner, A. Fishman, and D. Fox. Cabinet: Scaling neural collision detection for object rearrangement with procedural scene generation. *arXiv preprint arXiv:2304.09302*, 2023.

[19] D. Son and B. Kim. Local object crop collision network for efficient simulation of non-convex objects in gpu-based simulators. In *Robotics: Science and Systems*, 2023.

[20] H. Liu, L. Lee, K. Lee, and P. Abbeel. Instruction-following agents with multimodal transformer. *arXiv preprint arXiv:2210.13431*, 2022.

[21] C. Wang, M. Chai, M. He, D. Chen, and J. Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022.

[22] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763, 2021.

[23] B. CALLI, A. SINGH, J. BRUCE, A. WALSMAN, K. KONOLIGE, S. S. SRINIVASA, P. ABBEEL, and A. M. DOLLAR. Ycb benchmarking project: Object set, data set and their applications. *Journal of The Society of Instrument and Control Engineers*, 56(10):792–797, 2017.

[24] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[25] S. James, M. Freese, and A. J. Davison. Pyrep: Bringing v-rep to deep robot learning. *arXiv preprint arXiv:1906.11176*, 2019.

[26] E. Rohmer, S. P. Singh, and M. Freese. V-rep: A versatile and scalable robot simulation framework. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326, 2013.

[27] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3d objects, 2022.

[28] S. James, A. J. Davison, and E. Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. In *Conference on Robot Learning*, pages 334–343. PMLR, 2017.

[29] J. Matas, S. James, and A. J. Davison. Sim-to-real reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 734–743. PMLR, 2018.

[30] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12627–12637, 2019.

[31] J. So, A. Xie, S. Jung, J. Edlund, R. Thakker, A. Agha-mohammadi, P. Abbeel, and S. James. Sim-to-real via sim-to-seg: End-to-end off-road autonomous driving without real data. In *Conference on Robot Learning*, 2022.