Minimalist Motion Planning Using Global Topological Guarantees

Claire Liang[1] and Ross Knepper[1]

***Abstract*— In this position paper, we articulate and motivate a reframing of the 2D path planning problem. This reframing does global planning topologically and local control geometrically. It avoids the computational inefficiencies of specifying a path down to the level of a trajectory by instead specifying a homotopy class. This paper illustrates several of the benefits from this reframing, as well as the corresponding research questions that must be answered to make the realization of this reframed path planning pipeline possible.**

I. INTRODUCTION

In this paper we will describe a reframing of the 2D path planning problem. This reframing is designed for efficient computation by trading off responsibilities from the path planning stage to the control stage to do global planning topologically and local control geometrically. In doing this, we do not specify a path down to the level of a trajectory, but rather, specify a homotopy class. This reframing is particularly suited for dynamic environments and applications with less precise sensor input.

II. THE PLANNER TODAY

Generating a motion plan is computationally expensive. The problem with traditional motion planners is that the motion plan specification output by the planner is overly precise. Even though there are many valid motion plans for any given problem, traditional motion planners specify down to one precise trajectory. Committing to a specific path runs the risk of the path becoming infeasible before it can even be executed by the robot.

Consider a robot in the room in Figure 1. The robot starts at the green point and the goal point is the red point. Before trying to navigate to the goal, the robot might first generate a global map of its space, or perhaps it is even provided with a map. Then, with this map, the robot must generate some precise trajectory through the space that will get the robot from the starting point to the goal point; this is often done with some sample based planner. Once this trajectory is computed, the robot's control must ensure that the robot sticks to this path as strictly as possible.

However, if that original provided map is discovered to be wrong at execution time such that the planned precise trajectory cuts through an obstacle by just a hair, then the robot would have to adjust its map and restart the process of selecting a trajectory before proceeding.

Or, consider a scenario where the map is entirely accurate, but only captures the static obstacles in the space. When the robot is actually navigating the space, it might observe an unaccounted-for dynamic obstacle. This problem may be resolved by allowing for reactive control, but the robot may lose guarantees about its ability to return to the precise, preplanned trajectory. Alternatively, the robot might replan a trajectory similarly to the prior case, but depending on the motion of the dynamic obstacle, the additional computation could become intractable rapidly.

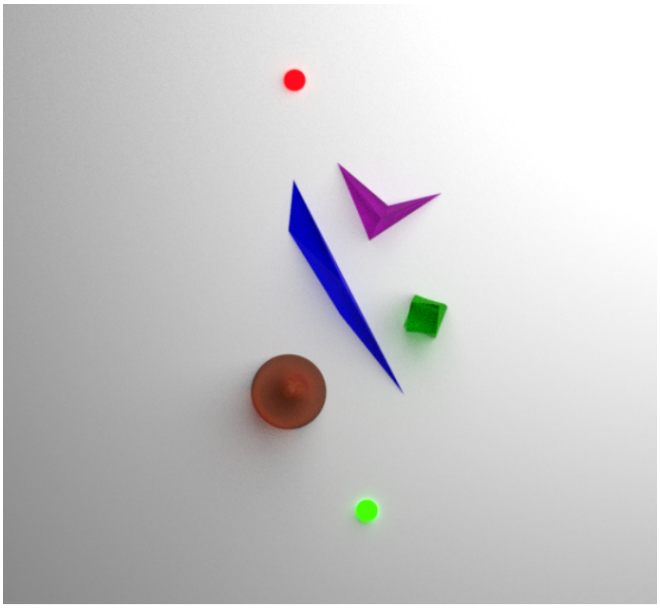

Fig. 1. An imagined room with obstacles to navigate. The starting point is the green point and the goal point is the red point.

A way to compensate for the rigidity of selecting a precise trajectory would be to select multiple trajectories, in case one does not work out [4]. The selection of this set of trajectories is important, because if one trajectory fails, it may be a monumental task to transition to another. For example, it is likely much easier to follow a new trajectory that lies within the same homotopy class as the original trajectory.

One could use topology aware sampling to generate a set of trajectories that fall in the same homotopy class [1]. However, the size of the set and sampling strategy would have enormous impact on the real performance of the robot and would need to be re-tuned per scenario.

III. THE ALTERNATIVE

To evade the issues that result from an overly precise plan specification output, we can instead specify to the level of homotopy classes rather than trajectories.

[1]Claire Liang and Ross Knepper are with The Department of Computer Science, Cornell University, Ithaca, NY, USA cyl48@cornell.edu, rak@cs.cornell.edu

Instead of adapting current planners to use metrically defined trajectories to capture the topology of the robot's space, we can avoid doing some of this inefficient work in the first place. There's no need to do the process of narrowing down all the way to one specific trajectory multiple times, then sorting through them to see if they satisfy our topological constraints. Instead, we could modify our planning and control to specify up to the homotopy class rather than a candidate trajectory within that homotopy class. Figure 2 illustrates the redundant costs of generating multiple candidate trajectories on the left versus generating a more generous, topologically-defined class specification on the right.

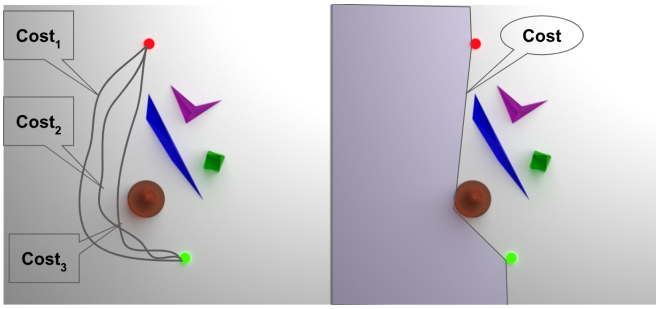

Fig. 2. Left: Generating multiple trajectories which each have the same cost of computation Right: Generating a homotopy-class level specification has a lower and non-redundant cost

As a consequence, the controller's job gets more difficult. Instead of following one trajectory, it now has much more freedom. However, as the controller operates during runtime, it is also the best equipped for handling the real-time unforeseen obstacles or errors that were not captured in the initial static map and adjusting along the way. For example, the trajectory depicted in Figure 3 would require no additional redundant computation and the robot would be free to avoid the three unforeseen obstacles depicted.

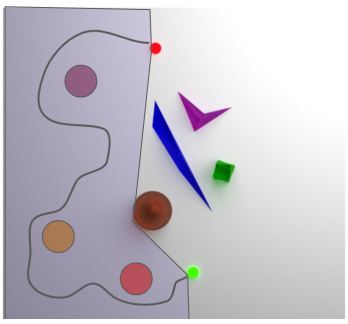

Fig. 3. The robot would be able to avoid the unexpected obstacles in the scene without having to recompute its path

In the following sections, we will motivate the benefits we gain from creating a path planner that generates a homotopy class rather than a trajectory. Furthermore, we will list the specific research questions that need to be answered to make this system possible.

IV. Benefits

1. If we reconsider the two scenarios where the original provided map is inaccurate or there are unexpected dynamic obstacles, having the path planner specify a homotopy class allows for the flexibility of a controller that can optimize and react in real time whilst proceeding along to the goal.

2. Specifying homotopy classes caters particularly well for environments where humans might be dictating directions to a robot. Human directions are often landmark based or topological in nature. For human users, a homotopy class specification may align better with their desired behavior for the robot than a single trajectory would [6].

3. Not requiring the metric specificity of a trajectory means that topological SLAM generated maps are particularly suited as input for planning (whereas they may not have been as suited for trajectory based planning, since they lacked the specificity necessary for obstacle avoidance [3]). A step beyond this, it may be possible to get away with less metric specificity in general, thus requiring less sensor input from the very beginning [5], [2].

V. Research Questions

1. The controller now has power to wander and roam, how does the planner impose the restrictions necessary to force the controller to adhere to a homotopy class (or set of classes)? We want to instead find a way to convert an entire homotopy class in a space into a control policy. This step is vital for the clean transition between a global topological specification and local metric behavior.

2. Furthermore, the specification of a homotopy class also needs to be practical. After all, one can loop around an object an unreasonable amount of times and easily fall into a specified homotopy class. If loops around obstacles are to be considered, is it best done at the path specification stage?

3. Since the planner does not need the metric precision to create a trajectory, how little sensing can we get away with for our map and still allow us to plan? What guarantees do we need from our map to be able to plan?

ACKNOWLEDGMENT

This material is based upon research supported by the National Science Foundation 1646417. We are grateful for this support.

References

[1] Subhrajit Bhattacharya, Vijay Kumar, and Maxim Likhachev. Search-based path planning with homotopy class constraints. In *Third Annual Symposium on Combinatorial Search*, 2010.

[2] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[3] Emilio Garcia-Fidalgo and Alberto Ortiz. Vision-based topological mapping and localization methods: A survey. *Robotics and Autonomous Systems*, 64:1–20, 2015.

[4] Ross A Knepper, Siddhartha S Srinivasa, and Matthew T Mason. An equivalence relation for local path sets. In *Algorithmic Foundations of Robotics IX*, pages 19–35. Springer, 2010.

[5] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J Leonard, David Cox, Peter Corke, and Michael J Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19, 2016.

[6] Illah R Nourbakhsh, Katia Sycara, Mary Koes, Mark Yong, Michael Lewis, and Steve Burion. Human-robot teaming for search and rescue. *IEEE Pervasive Computing*, (1):72–78, 2005.