DREU Summer 2020 Final Report

Collision Avoidance Templates for Multi-Agent Motion Planning

Report written by:
Jose Carrillo
Texas A&M University
College Station
jcarri877@tamu.edu

Graduate Student Mentor:
Irving Solis
Texas A&M University
College Station
g02005760@tamu.edu

Faculty Mentor:
Dr. Nancy Amato
University of Illinois Urbana-Champaign
namato@illinois.edu

Abstract

CBS-MP [1] is a recent variant of Conflict-Based Search (CBS) [2] which works in continuous space. One item our lab has been working on is creating methods to 'repair' conflicts in CBS-MP, where repair is defined as using local changes to conflicting robot paths outside of the graph representation to resolve the conflict and thereby avert the creation of a new Conflict Tree node. In this project, we begin explore one type of local repair, which we call Collision Avoidance Templates (CATs). CATs are solutions to atomic subproblems, which can be repeatedly pasted into the roadmap to repair conflicts, with the intent of finding a solution faster for complex problems. We begin by working with a simple and specific type of Multi-Agent Motion Planning (MAMP) problem to lay ground work for CATs. At the time of writing this report, the implementation able to solve this simplified problem is nearly complete, but not at a stage where experimental data can be collected to analyze effectiveness of CATs.

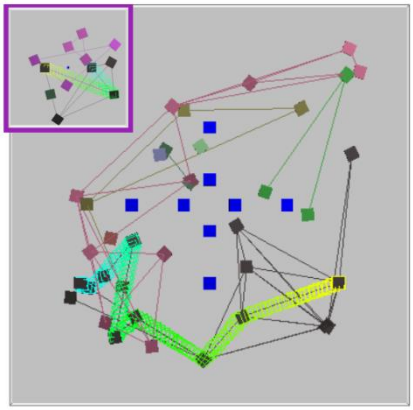

Figure 1: A template pasted into a PRM. In the upper-left image, the template path is shown. In the main image, the repaired path is shown, with the zig-zag in the path near the beginning being the repaired location where the template was pasted.

Background

Motion planning (MP) problems are defined as finding a motion a robot can take from point A to point B in a continuous workspace without colliding with static obstacles. Naively searching the entire continuous space is intractable because of the time required to check each configuration's validity is relatively high, so algorithms such as Probabilistic Roadmaps (PRMs) take samples of configurations to try and find from sample to sample [3].

Multi-agent motion planning (MAMP) is a generalization of MP where there are multiple robots which also must not collide with each other. The discrete environment version of this problem is multi-agent path finding (MAPF), where each robot occupies a vertex on a graph and cannot share vertices or cross the same edge with another robot at the same timestep. Conflict-based

search (CBS) [2] and variants of it have been applied to both these problems with much success. CBS works by planning a path for each robot without regard for the other robots, then checking for collisions on those paths. If a collision occurs between a pair of robots, the algorithm will then consider the two possible resolutions to the conflict, where either robot acquiesces and avoids that space at that time, then repeats the planning with that new constraint. This dichotomic conflict resolution is abstractly stored and represented in a Conflict Tree (CT), where each node of the tree stores the constraints, and other information, and each leaf of the tree can be evaluated by determining the paths with the constraints, finding a conflict, and create two children nodes corresponding to the two ways a conflict can be resolved. This leads to exponential growth rate with regards to the number of conflicts. Recent work has been done in applying CBS to MAMP problems, such as CBS-CT [4] and CBS-MP [1].

To improve the performance of CBS methods for MAMP, attempting to resolve conflicts locally before adding a constraint and expanding the CT, which may speed up the time to find a solution. We call this 'local repair', and there are different ways to do it. One way is to simply do additional sampling to the region and then run a MAMP solver in the local region with just the pair of robots, and has been implemented in our lab prior to the start of this project.

Part of the inspiration for this work is Interaction Templates, which are used in Task-and-Motion Planning problems to exploit the symmetry of hand-offs and other inter-robot interactions [5]. We seek to also employ the concept of templates to exploit symmetry, in our case to speed up conflict resolution.

Project Objective

The objective of this summer project was to implement a new type of local repair which uses conflict-free templates of motion for a pair of robots, which we call Collision Avoidance Templates (CATs). Some intuitive templates types include waiting for one robot to pass, moving behind the other robot, or aligning with the other robot to fit though a passage. To have a starting point to work from in the code, we simplified the problem to be that robots are rigid, jointless bodies which can freely translate and rotate at a given speed. This project used a pseudo-2D environment, where thin wafer-like robots navigate a similarly thin 3D space, and static obstacles stretch from the floor to the ceiling of this space[1]. The long-term goal is to find a generic method that can be applied to MAMP problems which can improve the speed at which sub-optimal paths are found; and this simplified problem will help build an understanding of how it would work and develop the code infrastructure needed before generalizing.

[1] The choice of pseudo-2D instead of 2D is because the code this project was built on is focused on 3D problems, and it would have me taken time to adapt it to be true 2D. The choice of 2D instead of 3D was to make the problem easier to visualize and understand.

Project Work Performed

The method developed has multiple stages, which will be discussed in more detail in the following sections. First, before the search begins, templates must be generated. Then CBS-MP begins to run, finds a conflict, and generates a "window". Now it must determine which templates to try via "pre-pasting". Pre-pasting returns a heuristic of how effective the paste will be, and from that we can then attempt to paste a template, trying templates ordered by heuristic until we find one that works or exhaust all valid templates. If we succeed in pasting a template, CBS-MP resumes as if there was no conflict; otherwise, it defaults normal CBS-MP behavior and creates a CT node.

Template Generation For this project, we created templates by setting up miniature environments of pairs of robots and running a MAMP solver on it. These environments were: head-on collision (robots had to swap places), orthogonal 'T-bone' collision (one robot going up, one robot going left) and the mirrored version, parallel motion (two robots very close together moving same direction). The MAMP solver we ran in these environments was a simple hierarchical planner, where the first robot's path was generated, followed by the second robot who had to avoid the first. In the future, being able to import existing templates for robots can be used to reduce the run time if available.

Pasting Window Before we get into the details of pasting, we need to outline some of the design decisions made for this project that frame the implementation later.

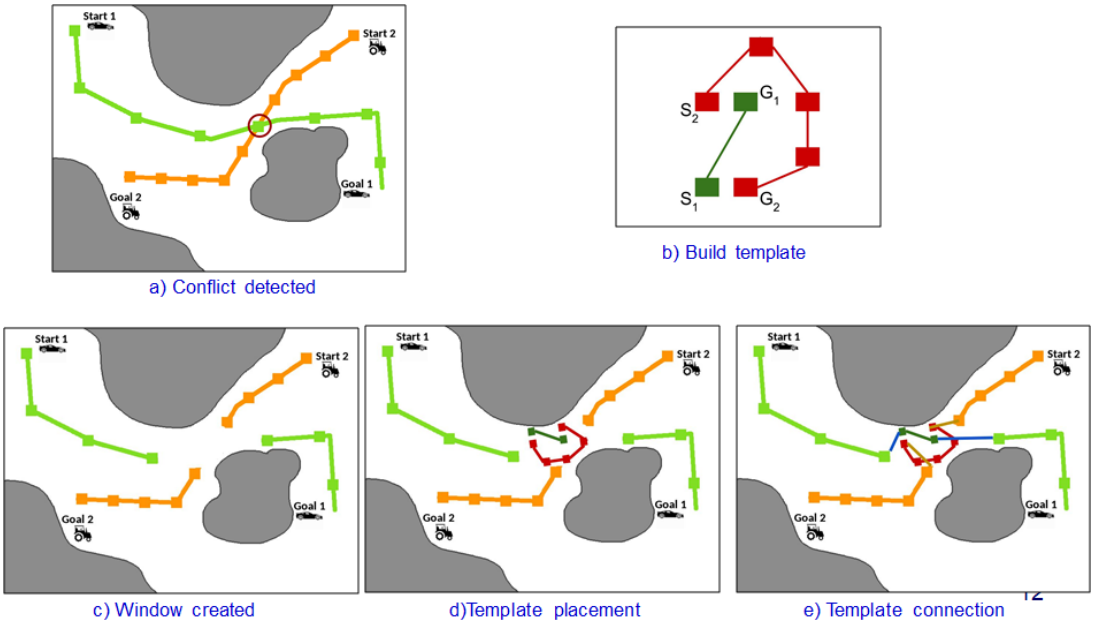

Figure 2: Cartoon depiction of how the pasting window works.

In order to use the template to resolve a conflict, the template must be merged into the robots' global paths. There are several ways that this could be done, such as taking the point of (hypothetical) collision for both the robots and template and using that as an anchor point to place the template translationally, then rotating the template into place and connecting it into the rest of the roadmap somehow. However, this makes it difficult to ensure that the distances to get to the template are the same, which goes against the decision discussed in the previous paragraph. Instead, we create a "window" of time around the conflict, and try to find a solution inside of that window that connects to the configurations in the path on the border of that window. We do this by following the edge where the conflict occurs w timesteps before, and placing a node there; then do the same stepping w timesteps after conflict. We place these pairs of nodes for both conflicting robots, ending up with 4 configurations, 2 in each configuration space corresponding to start and end of the window. Now when we want to paste our template, we simply have to consider these 4 configurations and try to match them to the 4 corresponding configurations in the template. When we paste our template, we can then connect these configurations between window and template to get a conflict-free path for the pair of robots guaranteed up until the end of the window after the template.

Pre-Pasting At this point, a conflict has been identified, and we have a list of templates which could be applied. Now we need to figure out how to transform the template into the right place to resolve the conflict, and then evaluate how good of a paste it will be. This needs to be done quickly, since it will be run for every template each time there is a conflict. We split pre-paste off from paste because paste contains the slower operations of actually checking collisions so should be called less often, and is sufficiently different in what it does to be considered something different.

As discussed before, we require that both robots arrive at the start of the template at the same time. Let S_1, S_2 denote the configurations of robots 1 and 2 at the start of the window, and let S_1' S_2' denote the configurations of robots 1 and 2 at the start of the pasted template. Assuming that the time it takes to get from one configuration to another is dependent solely on the Euclidean distance[2], and knowing that by the definition of the window that both robots will be at S_i at the same time, then we can determine a paste satisfying this constraint using simply geometry. As Figure 3 shows, if we translate the template so that the midpoint of the two starting configurations is coincident with the midpoint M of the starting configurations of the window, this condition is satisfied as S_1 to S_1' is equally long as S_2 to S_2'. Then we simply must get a good enough angle to rotate this template, and we have a valid paste. Algorithm 1 shows this implemented.

[2] This assumption is certainly not generalizable, but works for these particular robots.

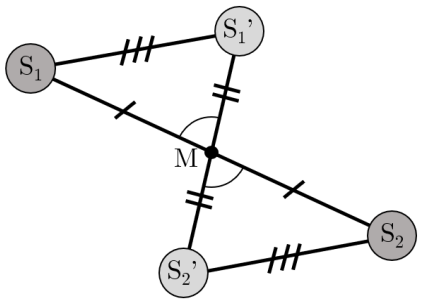

Figure 3: Geometric proof that this transformation ensures equal distance for both robots from start of window to start of pasted template.

Algorithm 1: PrePaste

---

1. Get the midpoints of startpoints of the template as M_T
2. Get the midpoints of startpoints of window as M_W
3. Get the linear transformation T_L that maps M_T to M_W
4. Get the POBs of window and template as POB_W and POB_T respectively
5. While we haven't found a valid pair of POBs
    a. If either POB is invalid, continue
    b. Get $POB_\Delta = POB_T - POB_W$ and break loop
6. If we couldn't find a valid pair of POBs, return failure
7. Take POB_Δ as a rotational transformation T_R around origin
8. Return the composed transformation $T = T_L \circ T_R$

In Algorithm 1, we define a Position Orientation Basis (POB)[3] to be some angle indicative of the direction the window and template are facing. We try a few different ways of getting this:

- The angle from the midpoint of startpoints to the midpoint of endpoints
- The angle from startpoint 1 to startpoint 2
- The angles from startpoint i to endpoint i for robots $i \in \{1, 2\}$

The first two methods are better heuristics as they account for both robots' average motion, but may fail if the difference between the two points the angle is being calculated from is zero[4]. If we then rotate the template so that its POB matches the window's POB, we have a decent heuristic for the angle to reduce the distances from the end of the template configurations to the end of the window configurations.

Algorithm 1 is easily extendable to higher dimensional space by simply using more POBs.

---

[3] There is probably a clearer name for this, but I cannot come up with one.

[4] If this is also close enough to zero, we consider it a failure as well, since estimating the average motion based on a tiny interval is likely going to be poor.

To evaluate how well pasting using this transform may go, we look at the 4 configurations of the window and the respective 4 configurations of the template (with the transform applied) and use some metric. We decided to use the inverse sum of the Euclidean distances from each window to template configuration as the metric.

Pasting Now that we have a transformation, it is relatively straight-forward. We iterate through each small discrete timestep, and check each interpolated configuration at that time against both static obstacles, and optionally against the paths of the other robots not in the pair. If there is a collision with a static obstacle or outsider robot, we consider the paste a failure and return to the loop of trying out other templates. Upon failure, all of the vertices added to the roadmap so far, and any other changes, should be reverted.

We have not yet studied what benefits and detriments of checking against the other robots, but by the fact that not checking may result in additional conflicts that CBS-MP must resolve, it seems that this could slow it down in some cases, or perhaps even cause some looping behavior with dramatic effects.

Project Results

As of the time of writing this report, template repair fails to resolve a conflict >90% of the time with the parameters tested. Time to compute a solution is very similar to the case without repair, most likely as a result of this. I have not yet begun to investigate why this is the case, but suspect it to be a bug in the code that checks if the paste collides with other robot's current paths. Alternatively, it may be that the templates used require too much space to maneuver and actually do fail to fit where they are pasted, but this seems less likely at this time.

The main contribution is the coding infrastructure created. I expect that once the remaining issues are ironed out, a performance boost will be observed. The code generated solving this toy problem will allow future work to be done with more abstract or complex templates, which could potentially speed up problems that cannot yet be solved efficiently enough.

In working through implementation, problems and improvements have been raised such as tardy-robustness and the separation of pre-pasting from pasting. Having this knowledge moving forward will allow more clarity when thinking of template repair, which is crucial if it is to be generalized in future work.

Future Work

The next step is to complete the work described in this paper so that experiments can be performed to verify that this theory does hold water. Extending the problem to higher dimensions is trivial, so that will likely be implemented and tested as well.

Beyond completing this particular project, there are many ways that this can be expanded upon assuming that is verified to work. Generalizing to robots with joints, robots that slowly turn, and/or non-holonomic robots can have important practical uses, but is quite complex to figure out how. The reason we limited this problem to rigid, jointless, fast-turning robots is because there is a clear intuition as to how templates of these robots look and work. Because the translational component of the C-Space is a direct map to the environment coordinates, we can translate a template by simply adding the displacement to each configuration. Similarly, we can rotate the template by rotating the positions of each configuration around the template's center point, and then adding the rotation to the orientation of the configuration. When this intuition is removed, it is more difficult to understand what the operators and constraints needed to position valid templates are. An analysis of the symmetries of both the robots and the motions involved may shed some light on this.

It may also be beneficial to apply template repair beyond just CBS-MP. Clearly, other centralized multi-agent planners should be able to adopt similar techniques to the one described in this paper, but even beyond that it may be fruitful to consider using templates as shared knowledge between decentralized motion planning.

Citations

[1] I. Solis, R. Sandström, J. Motes, and N. M. Amato, "Representation-Optimal Multi-Robot Motion Planning using Conflict-Based Search," Mar. 2020, Accessed: Aug. 17, 2020. [Online]. Available: http://arxiv.org/abs/1909.13352.

[2] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 219, pp. 40–66, Feb. 2015, doi: 10.1016/j.artint.2014.11.006.

[3] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996, doi: 10.1109/70.508439.

[4] L. Cohen, T. Uras, T. K. S. Kumar, and S. Koenig, "Optimal and Bounded-Suboptimal Multi-Agent Motion Planning," p. 8.

[5] J. Motes, R. Sandström, W. Adams, T. Ogunyale, S. Thomas, and N. M. Amato, "Interaction Templates for Multi-Robot Systems," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2926–2933, Jul. 2019, doi: 10.1109/LRA.2019.2923386.