Solid Freeform Fabrication 2023: Proceedings of the 34th Annual International Solid Freeform Fabrication Symposium – An Additive Manufacturing Conference
Reviewed Paper

Time-Optimal Path Planning for Heterogeneous Robots in Swarm Manufacturing

Ronnie F. P. Stone*, Junmin Wang*, Zhenghui Sha* [1]

* Walker Department of Mechanical Engineering, The University of Texas at Austin, TX 78712

Abstract

Time-optimal path planning is a critical problem in any system employing mobile robots. Much of the literature on this problem operates under two major assumptions: the environment is static, and the robots' effective shapes are constant and discretized either as a circle or a sphere, disregarding the impact of the robot's orientation on the overall task. These assumptions are restrictive in swarm manufacturing, where the environment is dynamically changing and manufacturing robots have geometric constraints for operation. For example, robots can not only move but also change their effective shapes depending on the parts they carry or the material deposited. These variations can significantly impact the path-planning solution. In this paper, we propose a methodology to find time-optimal paths in swarm manufacturing while explicitly considering the effective and changing shapes of mobile robots and the dynamic obstacles surrounding them.

Keywords: Autonomous Robotics Systems, Mobile Robots, Trajectory and Path Planning, Intelligent Manufacturing Systems, Swarm Manufacturing.

1. Introduction

The use of autonomous mobile robots (AMRs) in manufacturing is on the rise. Continuous advances in the field of path planning and the need for increased levels of automation in manufacturing processes are expanding the demand for the integration of AMRs into modern factories [1]. Integrating AMRs into common manufacturing environments, such as factory floors, promises to improve flexibility, stability, degree of automation, and efficiency in a myriad of processes [2]. These improvements are critical for modern manufacturing systems that must quickly adapt to sudden fluctuations in market demand, often requiring a flexible rearrangement of the production line [3]. The positive effects of this integration can already be seen in practice, as demonstrated by many existing mobile manufacturing systems [4], and particularly, swarm manufacturing (SM) (Fig. 1).

Swarm manufacturing (SM) is a novel manufacturing paradigm that employs a large number of heterogeneous mobile robots that can perform a wide range of manufacturing tasks autonomously and cooperatively [5]. The mobility of such robots within a SM process is fundamental because it allows them to operate at a high degree of cooperation. However, enabling cooperation requires efficient, time-optimal, and collision-free path planning for each robot.

Path planning in SM poses unique challenges that are not the focus of the existing literature available on the topic. One key assumption that is often made in path planning studies is to approximate the shape of all AMRs as n-spheres [6]. This is a restrictive assumption for SM, where knowing a robot's orientation may be essential in determining whether they can accomplish a given manufacturing task. Furthermore, a robot may be carrying an object that is much larger in one dimension than the other, meaning that a circular approximation (i.e., center of mass) would result in non-optimal or overly conservative path solutions.

[1] Corresponding author: zsha@austin.utexas.edu

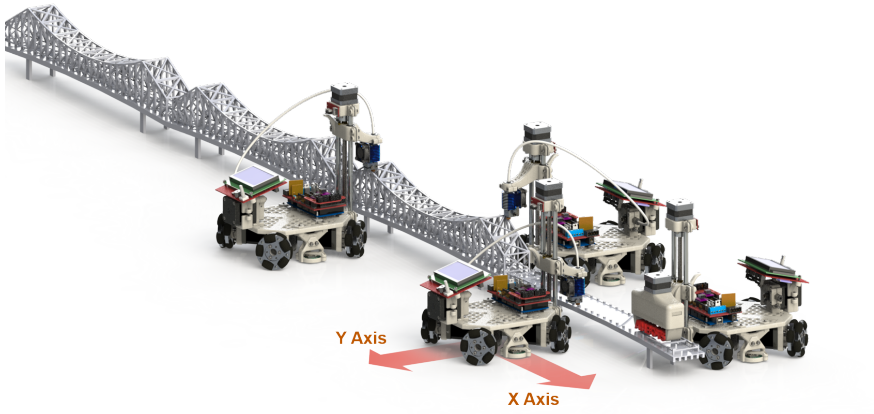

Figure 1: SM system where heterogeneous mobile robots manufacture large parts cooperatively.

Another assumption that is frequently made is that the surrounding environment is static [7]. By definition, an SM environment is necessarily dynamic, as not only there are multiple robots being employed simultaneously, but the parts being manufactured can increase in size and need to be moved over time. Therefore, in this paper, we attempt to address that gap by presenting a methodology for SM that can find time-optimal path solutions for arbitrarily-shaped mobile robots in dynamic environments.

The remainder of the paper is structured as follows. In Section 2 we discuss the relevant work available in the literature and how our paper addresses some of the identified gaps. Section 3 outlines the notation used throughout the paper and the scope of the problem we are studying. Our methodology is presented in Section 4. Finally, Section 5 describes our results, as well as future research directions.

2. Related Work

Although the field of SM is still fledgling, path planning of mobile robots has been widely studied [8]. Multiple methods for discretizing the workspace of AMRs have been proposed and they can be generally categorized as sampling-based or combinatorial [9]. Combinatorial methods are exact, since they do not rely on approximations of the workspace, and are provably complete, always succeeding in finding a solution if one exists or reporting failure otherwise. However, most combinatorial methods do not have practical applications because of their high computational and implementation complexity, as well as their poor scalability in the number of dimensions. It has been shown that a complete and exact algorithm for dynamic planning in a 3D space is at best NP-hard, making it intractable in a real engineering system [10].

Sampling-based methods, on the other hand, are satisfied with a weaker notion of completeness, and in return are able to significantly lower computational costs and find optimal solutions at a desired sampling resolution. Popular methods such as probabilistic roadmaps (PRMs) [11], and rapidly exploring random trees (RRTs) [12], have been extensively used in real systems, including manufacturing applications.

Modest attention was initially given to the path-planning problem in dynamic environments, as

it required adding time as an additional dimension to the configuration space of the AMRs. However, with the popularity of sampling-based methods, many novel algorithms for dynamic path planning were developed that relied on hybrid variants of PRM or RRT [13]. These methods, however, were based on a temporary approximation of dynamic obstacles as static obstacles, forgoing optimality and even resolution completeness. A major breakthrough came from the realization that the time dimensionality of the dynamic planning problem can be reduced only by considering safe intervals [14]. The premise is that, instead of dividing one static state into many states at each time step, one can couple the static state with a finite number of safe intervals, which is generally much smaller than the number of time steps. This approach was expanded to a more efficient algorithmic framework, now known as safe interval path planning (SIPP), and successfully tested in a real robotic system [15].

Although SIPP allowed finding efficient time-optimal path planning solutions in dynamic environments, there were still a number of restrictive assumptions in the framework such as the disregard for the robot's acceleration limits, the robot's shape discretization as an open disk, and the grid-like spatial representation of the workspace. Recent attempts have been made to relax these assumptions, but not with the focus of making SIPP-based planners viable in SM applications [16]. Therefore, we propose a novel methodology where safe intervals are calculated using a configuration space approach that inherently considers the shape and orientation of entities in the SM environment. It is also worth noting that other researchers have also leveraged SIPP-based planners for controlling swarm behavior in multi-agent scenarios with hundreds of entities, but still assuming circular entities [17].

3. Problem Definition

Given the current knowledge gap pertaining to the use of AMRs in swarm manufacturing scenarios, as highlighted in previous sections, we propose a methodology that can simultaneously address three critical challenges. The first is finding a feasible path for the single-agent problem without assuming the robot shape as a center of mass. The second is to allow the same framework to be applicable in dynamic environments, where obstacles have known trajectories and effective shape variations. Finally, we would also like for the framework to be scalable to multi-robot scenarios where more than one collision-free path must be found in reasonable time. In this section, we first introduce the notation that is used throughout the paper and then specify the scope of the theoretical analysis for each of the aforementioned challenges.

3.1. Preliminaries and Notations

In this paper, we assume that all entities exist in a two-dimensional Euclidean space as the abstraction of a factory floor. We use the word *entity* to encompass AMRs, dynamic obstacles, and static obstacles. We can define the configuration of any entity as $\vec{q} \in \mathbf{R}^4$, representing the x-position, y-position, orientation, and effective area of the entity with respect to a given Cartesian spatial frame $\{F_A\}$. Note that the term *effective area* is used here to denote the 2D-projection of an AMR, including any parts it may be carrying, onto the factory floor. Each entity also has its own Cartesian body frame, which allows its orientation θ with respect to the spatial frame to be well-defined. We will denote $\{F_B\}_i$ as the body frame of the ith entity, where $i \in \mathbf{Z}^+$. Following the same idea, we will use $\vec{q}_i$ to denote the configuration of each entity. The factory floor is represented by a rectangular area of width w and height h, where the spatial frame $\{F_A\}$ is attached to its bottom-

left corner (Fig. 2). Depending on the class of the entity, we can enforce how their configuration is allowed to change.

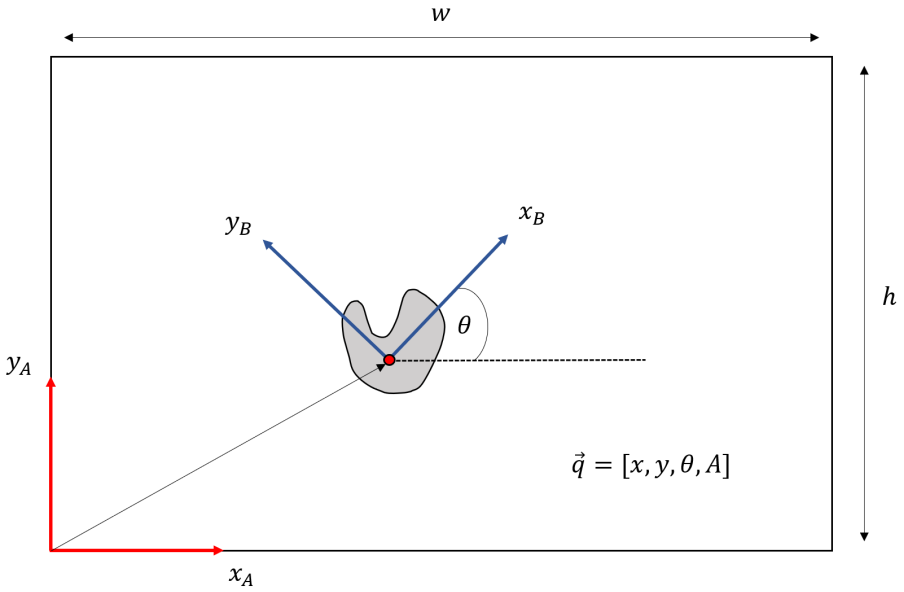

Figure 2: General representation of an arbitrarily-shaped entity on the manufacturing floor.

- Static obstacles must have a constant configuration throughout the process.

$$\vec{q_i} = [x_i, y_i, \theta_i, A_i] \tag{1}$$

- AMRs can vary their pose but not their effective area.

$$\vec{q_i}(t) = [x_i(t), y_i(t), \theta_i(t), A_i] \tag{2}$$

- Dynamic obstacles can vary all dimensions of their configuration.

$$\vec{q_i}(t) = [x_i(t), y_i(t), \theta_i(t), A_i(t)] \tag{3}$$

Note that in this study, we assume that we know, a priori, what the configurations for the static and dynamic obstacles are throughout the entire process. This means that we have perfect knowledge of the environment, which is a reasonable assumption for manufacturing applications in which we have full control of the processes. The goal is to find a time-optimal and collision-free sequence of configurations for arbitrarily-shaped AMRs, starting at an initial configuration $\vec{q_o}$ and ending at a final configuration $\vec{q_f}$, in an environment with any combination of arbitrarily-shaped static and dynamic obstacles. We assume that the AMRs are holonomic and that once a path-planning solution is found, the mobile robots will follow it exactly.

3.2. Path Planning Adaptations for Swarm Manufacturing

The first adaptation is to be able to explicitly consider the geometry of the AMRs in the path-planning algorithm. This is particularly important for manufacturing applications, since the robot's

orientation is crucial in determining whether a job can be accomplished or not. Consider, for instance, a mobile robot with an effective U-shape, which can be its true shape or a product of the part it is carrying, and it needs to reach a very specific final configuration (Fig. 3). If we fully consider its geometry, a solution can be found. However, if it were to be discretized as a two-dimensional center of mass, the desired configuration would be unattainable.

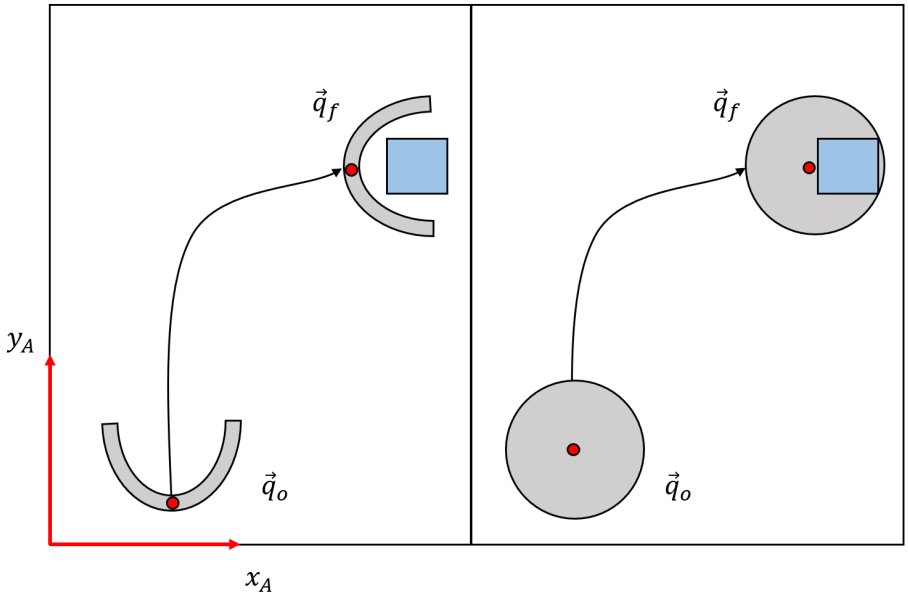

Figure 3: Representing the AMR using its actual effective area (left diagram) vs. discretizing it as a circular agent (right diagram).

The second adaptation is to extend the same path planning solution used to overcome the first challenge to also deal with dynamic environments. It would be too restrictive to assume that there are no moving entities on a factory floor. This is particularly important for SM, where the main idea is to employ an army of mobile robots that can perform a range of manufacturing tasks. We assume that the manufacturing environment can contain any combination of AMRs, static, and dynamic obstacles (Fig 4). Hence, we expect our method to be able to deal with any entity as long as its configuration can be described within the mathematical framework established in Section 3.1.

The final manufacturing-oriented adaptation that we would like our path-planning framework to have is the ability to find optimal solutions for multi-AMR scenarios. In this context, the notion of optimality must be more carefully defined, since this problem can be approached either from a decentralized or centralized manner. If the framework is centralized, then we can guarantee a global time-optimal solution in return for higher computational costs. However, if the framework is decentralized, then a global optimum is not guaranteed. The advantage of the decentralized approach is the low computational budget requirement, making it more appropriate for SM applications that employ hundreds or thousands of agents. In this work, we use a decentralized approach, but our framework is also valid for centralized planning, if desired. This trade-off is discussed further in Section 4.3.

4. Methodology

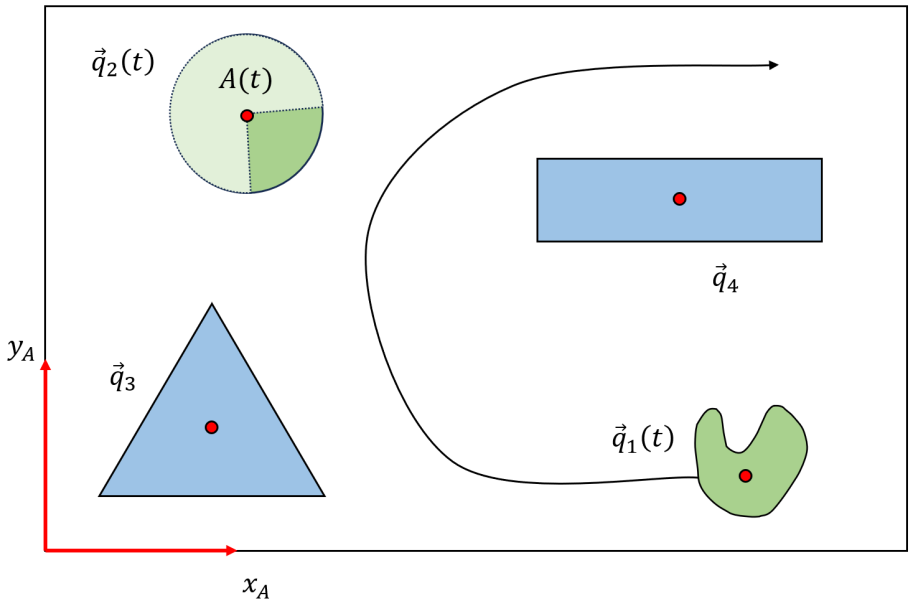

Figure 4: General manufacturing environment with a combination of arbitrarily-shaped static obstacles ($\vec{q}_3$ and $\vec{q}_4$) and dynamic obstacles ($\vec{q}_1(t)$ and $\vec{q}_2(t)$) travelling along known trajectories or varying their effective areas.

Our methodology follows a sequence of steps to achieve a time-optimal path planning solution (Fig. 5). First, we perform a general shape discretization, where the area of all entities is converted into a simple polygon. We then construct a map of the environment by building a state graph using a modified probabilistic roadmap and projecting the dynamic obstacles onto it to calculate the safe intervals for each node. Finally, we use a modified A* algorithm, as described in [15], to find the optimal AMR configuration function. Additionally, if more than one AMR is present, we adopt a decentralized approach where the path solution for each robot is found sequentially, where the ones whose solutions were found first are then treated as dynamic obstacles to the subsequent ones. For inputs, all we require are the configurations $q_i(t)$ of each obstacle present in the environment.

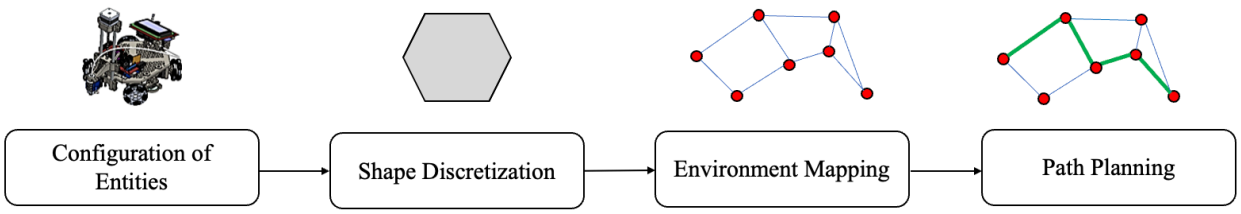

Figure 5: Overview of the proposed methodology.

4.1. Shape Discretization

The key idea in this process is to produce shapes whose collision-checking procedures will be efficient and that also faithfully represent the original geometry of the entities. Suppose that the geometry of one of the entities is provided to us. We can accurately discretize its shape, without

resorting to the conservative circular approximation, by sampling a number of control points along its boundary and then connecting them with straight edges (Fig. 6). The choice of both the number and location of control points is left to the user. Naturally, the more control points, the higher the computational cost, but with increased mode fidelity. The resulting shape will be a simple polygon (not necessarily convex), which allows us to perform efficient collision-checking with standard techniques such as ray-casting [18].

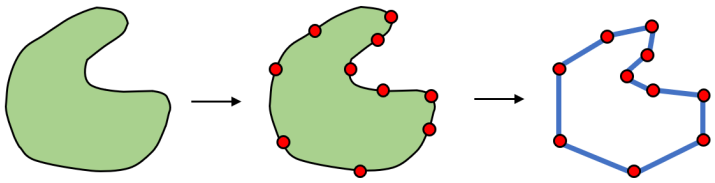

Figure 6: Entity discretization process based on straight edges and control points, turning any shape into a simple polygon.

4.2. Environment Mapping

Once all the shapes of the entities have been discretized, we build a map of the environment using a modified PRM. How refined the spatial discretization is, again, is left to the user, who can weigh the trade-off between desired solution quality and available computational resources. The initial roadmap constructed with PRM only needs to take into consideration the static obstacles, therefore only accepting configurations that are admissible with respect to the static obstacle space. Assuming a octagon-shaped agent that is only allowed to translate, and given a set of arbitrary static obstacles, we can discretize its free space (Fig. 7). It is also common practice to enlarge the static obstacles by a small dilation factor ϵ_{dil}, as seen by the faded red outline on the margins of the obstacles. This is done to ensure a safety margin and generate solutions where the agent does not just barely scrape by the obstacles.

The graph generated by the PRM procedure depends on the geometry of both the AMR and static obstacles. Note that we are not constraining our analysis to agents that can only translate. When the orientation of the agent is allowed to change, a new degree of freedom is included and the graph generated by the PRM procedure would be three-dimensional and it would be impossible to overlay it on top of the AMR's workspace, prohibiting a visual display of the problem [19]. We can define the free configuration space of the agent to be the set of all admissible configurations $\vec{u}$ discretized by the sampling procedure:

$$\vec{C}_{free} = \{\vec{u}_1, \vec{u}_2, ..., \vec{u}_n\} \tag{4}$$

Once $\vec{C}_{free}$ is defined, the safe intervals need to be calculated taking into account the geometry of the dynamic obstacles, which move according to known trajectory functions $\vec{q}_i(t)$. First, a desired time-step Δt must be chosen so that the trajectory functions can be sampled. We then project the covered areas of these dynamic obstacles at each time step into the configuration space of an agent. If a collision is detected for any of the originally admissible nodes in $\vec{C}_{free}$, then it is marked as

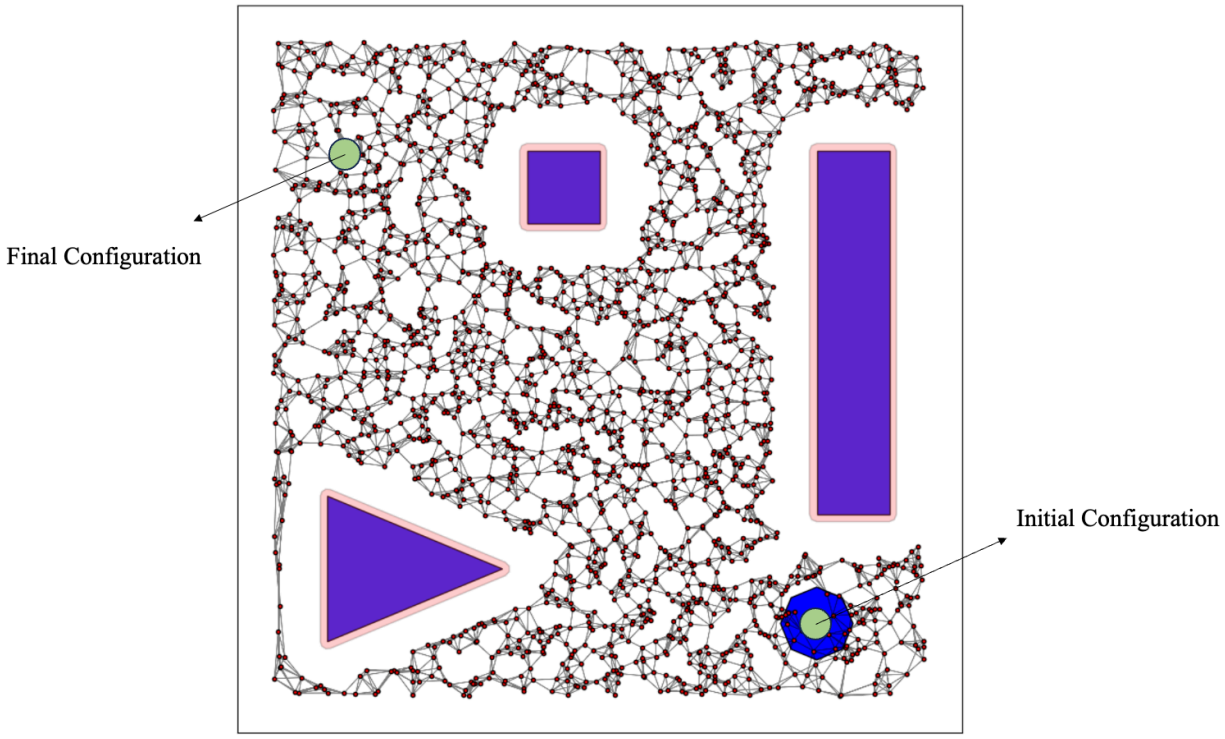

Figure 7: Environment discretization process. The agent is colored blue, the static obstacles are colored purple with a small dilation factor highlighted in red, and the final PRM graph overlaid on top.

unsafe at that given time instance (Fig. 8). For any node at an arbitrary time $n\Delta t$, the safety condition can be mathematically represented as:

$$\bigcap_{i=1}^{N} A(n\Delta t)_i = \emptyset, \ \forall i \leq N, \tag{5}$$

assuming a total of N entities in the environment, where $A_i(n\Delta t)$ is the effective area of the i^{th} entity at time-step $n\Delta t$, $n \in \mathbf{Z}_0^+$.

The safe instances for each configuration are then collapsed into intervals, significantly reducing the dimensionality of the problem. For instance, if a certain configuration is collision-free throughout the entire process, it will have a corresponding safe interval of $[0, \infty]$. This reduces the number of states for that specific configuration from N_t to 1, where N_t is the total number of time-steps generated and is given by:

$$N_t = \frac{T}{\Delta t}, \tag{6}$$

where T is the total process time. Note that N_t could be in the order of thousands if the desired time discretization is small, making the interval-based reduction of the dimensionality of the problem significant. Finally, the configuration graph must be converted into a state graph. Each configura-

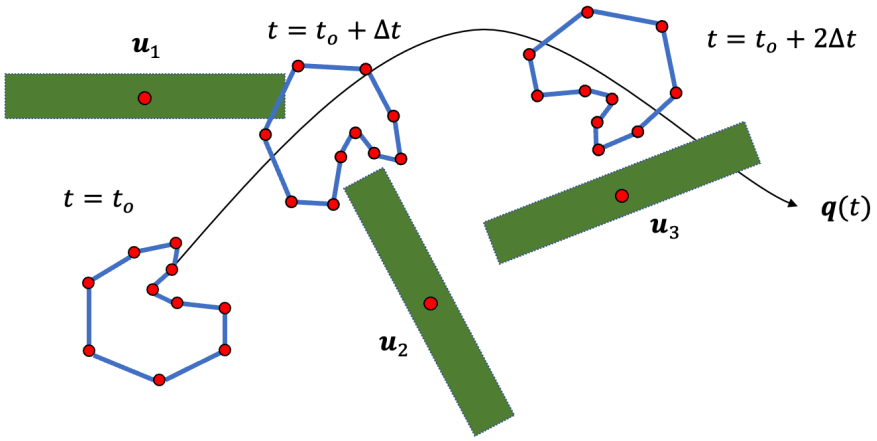

Figure 8: Safe intervals calculation. The dynamic obstacle is projected at each time step and collision is checked against nearby configurations. For instance, configuration $\vec{u}_1$ is safe at $t = t_o$ and $t = t_o + 2\Delta t$, but not at $t = t_o + \Delta t$.

tion $\vec{u}$ in $\vec{C}_{free}$ will generate k states $\vec{s}$, where k is the number of safe intervals that configuration $\vec{u}$ has (Fig. 9). There will always be at least as many states as configurations.

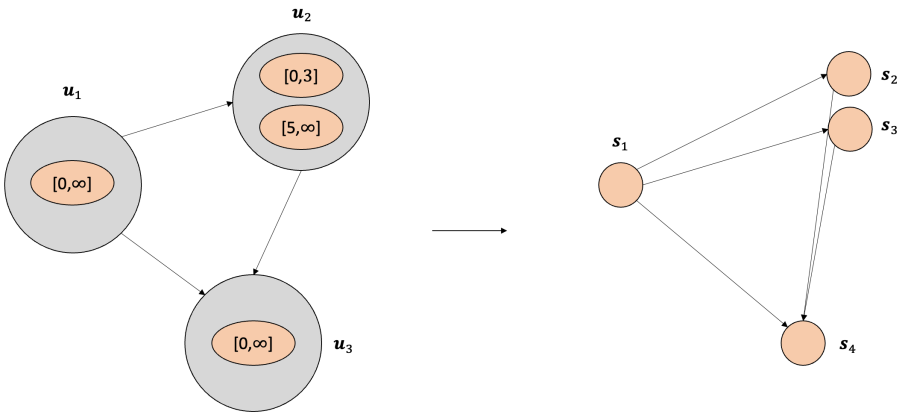

Figure 9: Configuration graph to state graph transition. The connectivity in both graphs is maintained and represent the range of possible motions between nodes.

4.3. Path Planning

At this point, the problem has been reduced to a graph traversal problem where the cost of each edge is the time taken to traverse between the states. If only translation is allowed, the state graph is two-dimensional, whereas if rotations are allowed, it is three-dimensional. As previously mentioned, it can be solved using a modified variant of the popular A* algorithm [15]. As long as the heuristic chosen is admissible, we are guaranteed to find the optimal path.

For the multi-AMR case, a decentralized approach is employed. Given any number of AMRs,

a state graph is built for one of them, and an optimal path is found using the modified A* algorithm. The solution for the first AMR is then framed as a configuration $q(t)$, which is then treated as a dynamic obstacle for the second AMR. This process continues until a path is found for all robots.

5. Results

In order to validate our framework, we generated four distinct test cases. First, we wanted to assess the impact of varying the size of the agent (Test Cases 1 and 2). Then, we chose a geometry that is not easily discretized as a circle, and saw how considering orientation would influence the solution (Test Case 3). Finally, we wanted to demonstrate our multi-robot approach while also highlighting that our framework can deal with variable-area dynamic obstacles (Test Case 4). All test cases have the same layout in terms of static obstacles (Fig. 7) and sample a thousand configuration nodes with a minimum of six edges each. They are listed below and classified by the number and type of each entity. Additionally, we provide the total time expected for each path. Again, we assume that the AMR is holonomic and travels at a constant speed. Shorter paths do not necessarily mean shorter times, since agents are allowed to wait in a state as long as necessary.

- 1) Small octagon-shaped agent, no rotations allowed, rectangular dynamic obstacle moving in a circle (Fig. 10). Expected time of 11.67s.
- 2) Large octagon-shaped agent, no rotations allowed, rectangular dynamic obstacle moving in a circle (Fig. 11). Expected time of 14.76s.
- 3) Rectangular agent, rotations allowed, rectangular dynamic obstacle moving in a circle (Fig. 12). Expected time of 13.26s.
- 4) Two small octagon-shaped agents, no rotations allowed, growing rectangular dynamic obstacle (Fig. 13). Expected time of 12.44s.

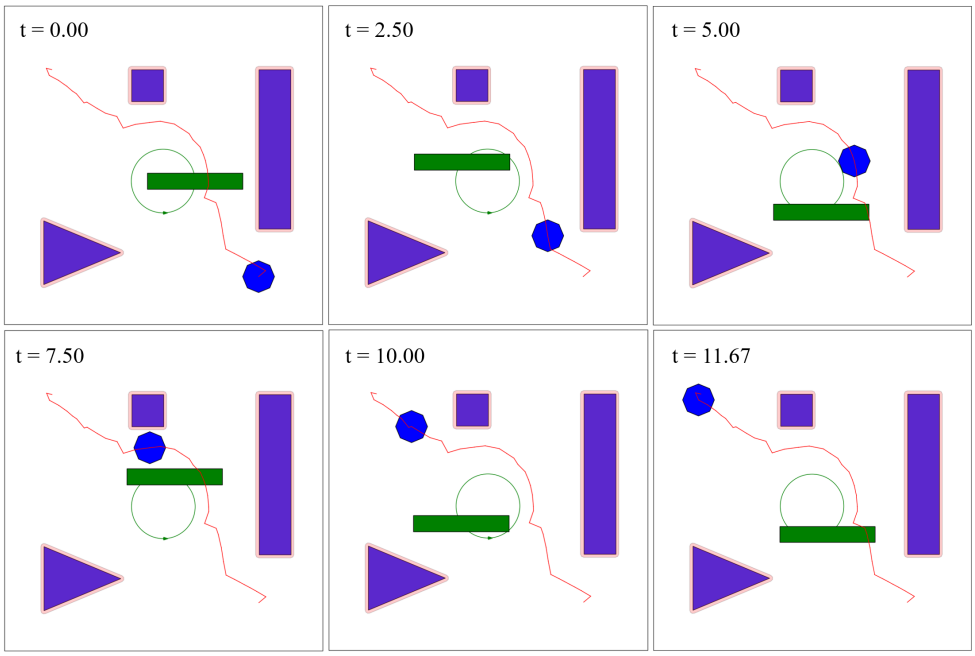

Figure 10: Test Case 1 Snapshots

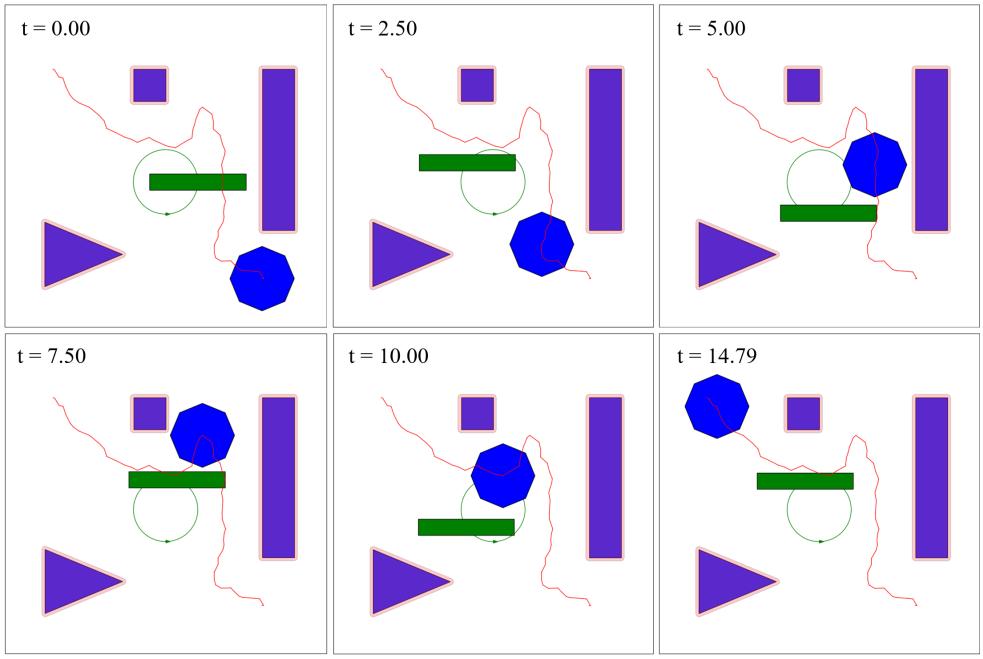

Figure 11: Test Case 2 Snapshots

The average computational time to find the path after the graph was built for each of the test cases was approximately 0.01 s. All tests were performed on a Windows 11 computer with an AMD Ryzen 7 4700U CPU with 32GB of RAM. Finally, the snapshots of the path planning solutions were all generated using a custom Python script with Matplotlib [20].

6. Discussion & Future Research

Once the state graph was completed, a collision-free path was found for the AMRs in each scenario in a couple of milliseconds, showing the potential of the framework for real-time applications. The first test case shows that the agent intelligently moves around the moving dynamic obstacle, showing a curved path. As soon as the agent size is increased, as in the second test case, that path is not possible anymore, and it adapts by first moving it into a corner and then waiting for the dynamic obstacle to move away before proceeding to the goal. The third test case highlights the power of considering orientation in the path planning solution. If the rectangular agent would only be allowed to translate, a solution would not exist since it does not move fast enough to safely bridge the gap in the middle. However, by allowing rotations, it can intelligently adjust itself to continuously move towards the goal. The final test case shows that the agents can also behave intelligently with respect to each other, developing a form of cooperation where one of them waits for the second one to pass through a narrow segment. The multi-robot test case also shows that the AMRs are aware of the growing effective area of the dynamic obstacle. It is important to note that the algorithm is also complete in the sense that it can determine whether a solution exists or not based on the resolution of the graph. This is relevant, for instance, in a scenario where one needs to determine how many AMRs can be included in a single factory until their tasks become impossible to complete due to lack of space for motion planning.

Although the results look promising, there are still limitations that could prevent them from

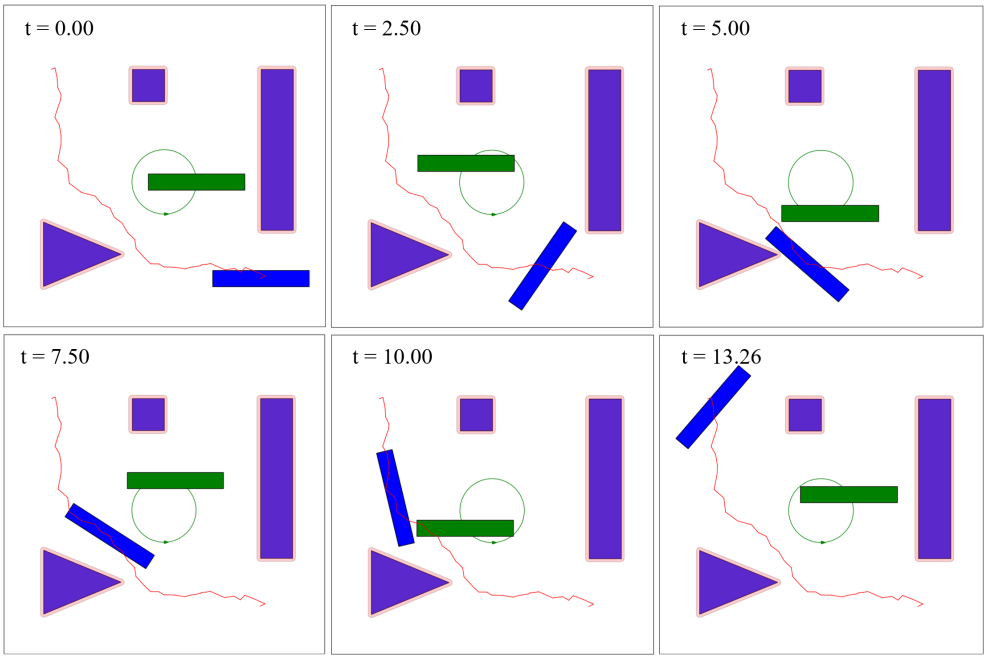

Figure 12: Test Case 3 Snapshots

being used successfully in SM applications. For example, graph building, depending on the number of sampled nodes and desired connectivity density, could take up to one second to complete. If the graph is not pre-built, which could be a possibility in a manufacturing environment, more work needs to be done in speeding up the mapping of the environment. Since we employ a PRM approach, one possibility is to modify it using lazy techniques to evaluate the collision-checking during the path planning stage [21].

Regarding the multi-robot solution, although the decentralized approach does not guarantee a global optimum, it is computationally scalable for swarm manufacturing applications, in which we envision hundreds to thousands of robots working simultaneously. Furthermore, since the time taken to complete manufacturing tasks is many orders of magnitude higher than the time taken to complete a path, for instance, 3D printing a large part, it is unlikely that there will be too many path-planning problems that require multiple AMRs to be considered. Hence, the small loss in optimality in choosing a decentralized approach is well-justified in SM applications.

An interesting approach to allow for true centralized time-optimal planning would be to subdivide the manufacturing environment into areas. This is a common strategy on factory floors where different areas are responsible for different parts of the manufacturing process. In this way, only the agents in a given area are considered for path planning purposes, reducing the dimensionality of the problem and allowing a feasible real-time centralized solution.

7. Conclusion

In this paper, we successfully developed and validated a time-optimal collision-free path planning framework for heterogeneous robots in swarm manufacturing. The framework is general in that it is agnostic to the shapes of the entities, allows for dynamic obstacles, and accommodates si-

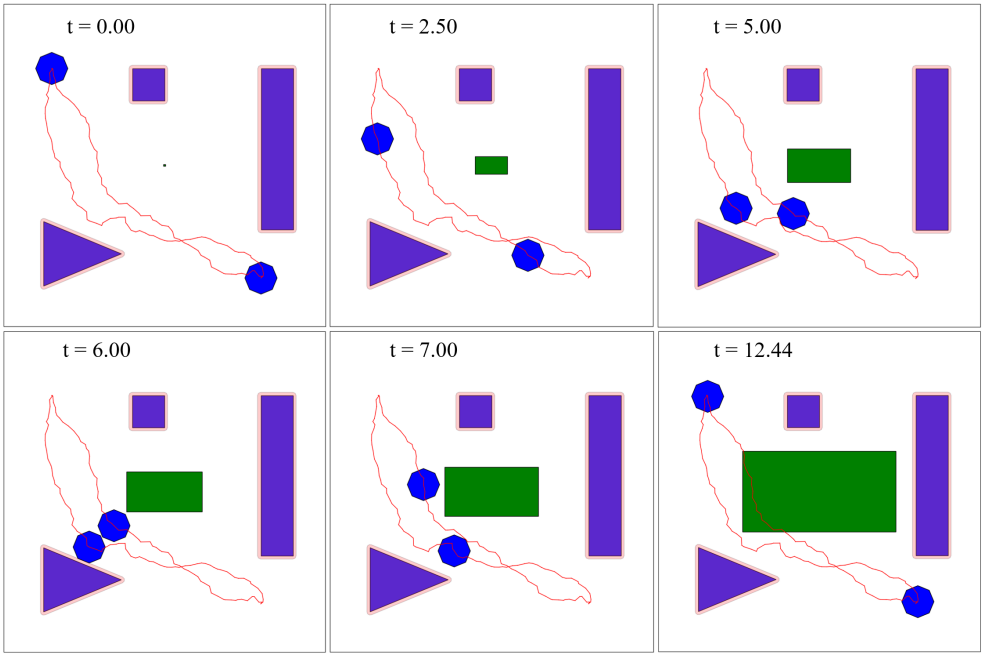

Figure 13: Test Case 4 Snapshots

multaneous planning for multiple AMRs. The results showed that the framework is promising and we look forward to exploring the research directions stemming from the challenges and insights presented in the discussion section.

References

[1] G. Ullrich and T. Albrecht, "History of automated guided vehicle systems," in *Automated Guided Vehicle Systems: A Guide-With Practical Applications-About The Technology-For Planning*, pp. 1–26, Springer, 2022.

[2] H. Unger, T. Markert, and E. Müller, "Evaluation of use cases of autonomous mobile robots in factory environments," *Procedia Manufacturing*, vol. 17, pp. 254–261, 2018.

[3] G. Fragapane, D. Ivanov, M. Peron, F. Sgarbossa, and J. O. Strandhagen, "Increasing flexibility and productivity in industry 4.0 production networks with autonomous mobile robots and smart intralogistics," *Annals of operations research*, vol. 308, no. 1-2, pp. 125–143, 2022.

[4] K. Azadeh, R. De Koster, and D. Roy, "Robotized and automated warehouse systems: Review and recent developments," *Transportation Science*, vol. 53, no. 4, pp. 917–945, 2019.

[5] L. Poudel, L. G. Marques, R. A. Williams, Z. Hyden, P. Guerra, O. L. Fowler, Z. Sha, and W. Zhou, "Toward swarm manufacturing: Architecting a cooperative 3d printing system," *Journal of Manufacturing Science and Engineering*, vol. 144, no. 8, p. 081004, 2022.

[6] R. Stern, "Multi-agent path finding–an overview," *Artificial Intelligence: 5th RAAI Summer School, Dolgoprudny, Russia, July 4–7, 2019, Tutorial Lectures*, pp. 96–115, 2019.

[7] G. Kyprianou, L. Doitsidis, and S. A. Chatzichristofis, "Towards the achievement of path planning with multi-robot systems in dynamic environments," *Journal of Intelligent & Robotic Systems*, vol. 104, no. 1, p. 15, 2022.

[8] B. Patle, G. Babu L, A. Pandey, D. Parhi, and A. Jagadeesh, "A review: On path planning strategies for navigation of mobile robot," *Defence Technology*, vol. 15, no. 4, pp. 582–606, 2019.

[9] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006. Available at http://planning.cs.uiuc.edu/.

[10] J. Reif and M. Sharir, "Motion planning in the presence of moving obstacles," *Journal of the ACM (JACM)*, vol. 41, no. 4, pp. 764–790, 1994.

[11] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[12] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[13] L. Jaillet and T. Siméon, "A prm-based motion planner for dynamically changing environments," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 2, pp. 1606–1611, IEEE, 2004.

[14] J. P. Van Den Berg and M. H. Overmars, "Roadmap-based motion planning in dynamic environments," *IEEE transactions on robotics*, vol. 21, no. 5, pp. 885–897, 2005.

[15] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *2011 IEEE international conference on robotics and automation*, pp. 5628–5635, IEEE, 2011.

[16] K. Yakovlev and A. Andreychuk, "Towards time-optimal any-angle path planning with dynamic obstacles," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 31, pp. 405–414, 2021.

[17] K. Kasaura, M. Nishimura, and R. Yonetani, "Prioritized safe interval path planning for multi-agent pathfinding with continuous time on 2d roadmaps," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10494–10501, 2022.

[18] K. Hormann and A. Agathos, "The point in polygon problem for arbitrary polygons," *Computational Geometry*, vol. 20, no. 3, pp. 131–144, 2001.

[19] T. Lozano-Perez, *Spatial planning: A configuration space approach*. Springer, 1990.

[20] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[21] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, vol. 1, pp. 521–528, IEEE, 2000.