2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

Multirobot Sequential Composition

Glenn Wagner, Howie Choset, Avinash Siravuru

***Abstract*— Conventional path planning algorithms compute a single path through the configuration space. There is no guarantee that a physical robot will be able to track the trajectory while avoiding collisions, particularly in the presence of environmental perturbations and errors in the process model. Sequential composition combines planning and control by computing a sequence of controllers to execute rather than a single trajectory, offering greater safety guarantees. In this paper, we apply sequential composition to multirobot systems in a scalable fashion using M*, an advanced multirobot path planning algorithm. Controllers will vary in size and geometry, and thus take different amounts of time to execute. To handle these differences, we introduce the time augmented joint prepares graph and the approximate time augmented joint prepares graph which simplifies implementation by discretizing time. We validate our approach in a mixed reality test framework.**

I. Introduction

Conventional path planning algorithms compute a single trajectory in the configuration space of the system. Environmental perturbations and errors in the process model can cause the system to depart from the planned trajectory. Running a trajectory-tracking controller can provide robustness to small perturbations. However, because the planner only considers the nominal trajectory, there is no guarantee that the trajectory-tracking controller will avoid collisions while recovering from larger perturbations (Figure 1a). *Sequential composition* seeks to produce paths that are robust to perturbations by planning not in the configuration space of the system, but rather over a set of controllers [3, 4, 5, 9, 10, 11]. The resulting plan is a sequence of controllers, chosen such that the goal set of each controller lies in the domain of attraction of the next controller (Figure 1b), defining a plan over a "thick" region of the configuration space, instead of a single "thin" trajectory. As long as the system remains within the domains of the planned controllers safety is guaranteed. Furthermore, the system can readily detect when it is subject to a large enough perturbation that the original sequence of controllers can no longer guarantee safety, triggering replanning.

Computing a sequential composition plan is a multi-step affair. First, a set of controllers with well defined domains must be deployed in the environment. The domain of a controller is an invariant subset[1] of the domain of attraction of the controller that lies entirely within the free configuration space. Controllers should be deployed so that the union of their domains cover as large a fraction of the free configuration space as possible.

Glenn Wagner is a post-graduate scholar at the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 15213 gswagner@cmu.edu
Howie Choset is a professor at the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 15213 choset@cs.cmu.edu
Avinash Siravuru is a PhD student in Mechanical Engineering at Carnegie Mellon University, Pittsburgh PA 15213 avinashs@cmu.edu

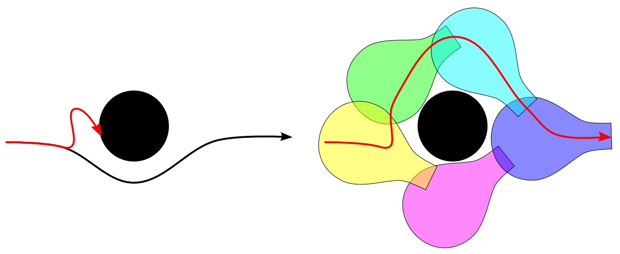

(a) Path planning (b) Sequential composition

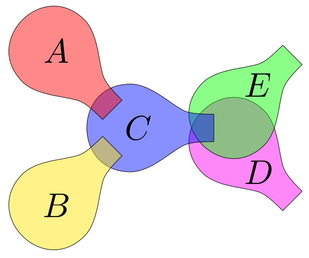
Fig. 1: **(a)** Conventional path planning algorithms compute a single trajectory in configuration space (black arrow). When a robot is perturbed from its planned trajectory, there is no guarantee that a trajectory tracking controller will avoid collisions. **(b)** Sequential composition planners [4, 5] compute a plan consisting of a sequence of controllers. In this example the initial path is yellow, purple, blue. Safety is guaranteed as long as the robot remains within the domains of the planned controllers, and perturbations large enough to invalidate the plan can be easily detected and trigger replanning.

Second, the order in which controllers can be executed is described by a *prepares graph*. Controller A is said to *prepare* controller B, denoted $A \succ B$,[2] if the goal set of A is a subset of the domain of B. A single controller can prepare multiple controllers. Once a robot reaches the goal set of its current controller, it will be in the domain of all the prepared controllers. The robot then uses a precomputed path or policy to choose which controller to execute next. The prepares graph is a directed graph that captures the prepares relationship between controllers (Figure 2). Each controller is represented as a vertex in the prepares graph whose out-neighbors are the prepared controllers. Thus a feasible sequence of controllers can be computed by finding a path in the prepares graph.

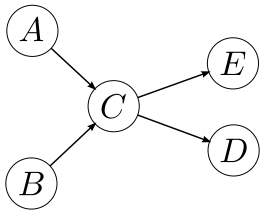
Prior work on sequential composition has primarily focused on single robot systems, which are characterized by small prepares graphs. A policy over the entire prepares graph can then be computed using standard graph-search algorithms [3, 4, 5, 9, 10, 11], resulting in a control policy

[1] If a robot starts in the domain of a controller it will remain within the domain of the controller, although in an abuse of terminology we allow the robot to leave the domain of the controller if it does so by passing through the goal set of the controller.

[2] The prepares relation superficially resembles a partial ordering but may contain cycles. The direction the prepares symbol points follows the usage of [4] and is opposite the standard partial ordering symbol.



(a) Controllers



$$\begin{tikzcd} A \arrow[dr] & & E \\ & C \arrow[ur] \arrow[dr] & \\ B \arrow[ur] & & D \end{tikzcd}$$

(b) Prepares graph

Fig. 2: **(a)** An example of a controller deployment where $A \succ C$, $B \succ C$, $C \succ D$ and $C \succ E$. **(b)** The associated prepares graph.

defined over a much larger fraction of the free configuration space than would be possible with a single conventional controller.

The prepares graphs associated with multirobot systems are exponentially larger than those for single robot systems. Ayanian and Kumar [1, 2] were able to apply sequential composition to multirobot systems by constructing multirobot controllers then finding a single path through the prepares graph using A*, rather than computing a full policy over the entire graph, but were still limited to small numbers of robots due to the exponential complexity of finding even a single path. While not using the language of sequential composition, Ulusoy et al. [12] describe an approach for using Linear Temporal Logic (LTL) solvers to coordinate robots executing single robot primitives, explicitly allowing for the primitives to be controllers. They also introduced region automata to handle primitives with different durations. However, the computational cost of the LTL solver limited the approach to a system of 2 robots.

In this paper, we present an approach to robust multirobot path planning based on combining the concepts of sequential composition with high-performance multirobot path planning algorithms. Our basic approach is to compute a prepares graph for each robot, then take the direct product to define a *joint prepares graph* for the system as a whole. We then use a multirobot path planning algorithm called M* [14, 15] to find a path in the joint prepares graph which assigns a sequence of single robot controllers to each robot.

M* assumes that all robots move synchronously, i.e. that the actions associated with each edge in the search graph take the same amount of time to execute. However, robots will take varying amounts of time to execute controllers with different geometries and sizes. We therefore introduce the *Time Augmented Joint Prepares Graph* (TAJPG), inspired by region automata [12], which captures differences in average execution time for different controllers, thus improving the feasibility of paths computed by M*. We then describe the *Approximate Time Augmented Joint Prepares Graph* (ATAJPG) that simplifies the integration of TAJPG with multirobot path planning algorithms such as M* by discretizing time. We validate our approach on a mixed-reality test bed, combining physical and simulated robots.

II. Multirobot sequential composition

Consider system of n potentially inhomogeneous robots r^i, $i \in \{1, \ldots n\}$. Robot r^i has a configuration space Q^i and dynamics $\mathcal{F}^i : Q^i \times \mathcal{U}^i \to \mathcal{T}Q^i$, where $\mathcal{U}^i$ is the space of control inputs of r^i and $\mathcal{T}Q^i$ is the tangent bundle of Q^i that describes the possible velocities of r^i. A controller for r^i fixes the control input at every point in its domain which, combined with the dynamics function, produces a mapping $\mathcal{C}^i_k : \mathcal{D}^i_k \to \mathcal{T}Q^i$ where $\mathcal{D}^i_k \subset Q^i_{\text{free}}$ is the domain of the controller in the free configuration space. The flow of the controller $\Phi_{\mathcal{C}^i_k} : Q^i \times \mathbb{R} \to Q^i$ maps the configuration q^i of r^i at time zero to the configuration it would occupy $\Phi_{\mathcal{C}^i_k}\left(q^i, t\right)$ at time t under the influence of controller $\mathcal{C}^i_k$. Let the set of controllers for r^i be denoted $\mathfrak{C}^i$.

Each controller $\mathcal{C}^i_k$ has a goal set $\mathcal{G}^i_k \subset \mathcal{D}^i_k$. The flow of valid controller takes every point in its domain to the goal set in finite time

$$\forall q^i \in \mathcal{D}^i_k \; \exists T \geq 0 \; s.t \; \Phi_{\mathcal{C}^i_k}\left(q^i, T\right) \in \mathcal{G}^i_k \wedge \\ \forall t \in [0, T] \; \Phi_{\mathcal{C}^i_k}\left(q^i, t\right) \in \mathcal{D}^i_k$$

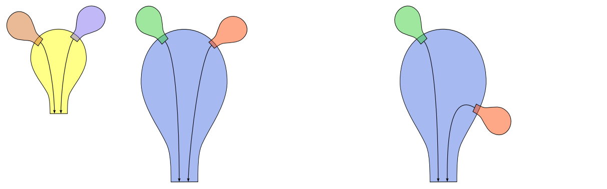
A controller $\mathcal{C}^i_k$ prepares a different controller $\mathcal{C}^i_\ell$, $k \neq \ell$, if $\mathcal{G}^i_k \subset \mathcal{D}^i_\ell$, denoted $\mathcal{C}^i_k \succ \mathcal{C}^i_\ell$. $\mathcal{C}^i_k$ prepares itself if $\mathcal{G}^i_k$ is an invariant set or if a robot executing $\mathcal{C}^i_k$ will always revisit $\mathcal{G}^i_k$ after finite time without leaving $\mathcal{D}^i_k$. We assume that each controller prepares at least one controller, potentially itself, so that a robot will always have at least one valid controller to execute. We term the controllers that prepare a given controller $\mathcal{C}^i_k$ the *preparing controller*s of $\mathcal{C}^i_k$.

The prepares relations between a set of controllers $\mathfrak{C}^i$ for a robot r^i are described by the prepares graph G^i, where each vertex G^i represents a controller in $\mathfrak{C}^i$. A directed edge connects $\mathcal{C}^i_k$ to $\mathcal{C}^i_\ell$ if and only if $\mathcal{C}^i_k \succ \mathcal{C}^i_\ell$.

The joint prepares graph of the multirobot system is the direct product of the individual prepares graphs $G = \prod_{i=1}^n G^i$. Each vertex v_k in G is associated with a tuple of controllers $\left(\mathcal{C}^1_k, \ldots, \mathcal{C}^n_k\right)$, which assigns r^i to execute $\mathcal{C}^i_k$. An edge connects $\left(\mathcal{C}^1_k, \ldots, \mathcal{C}^n_k\right)$ to $\left(\mathcal{C}^1_\ell, \ldots, \mathcal{C}^n_\ell\right)$ if and only if $\mathcal{C}^i_k \succ \mathcal{C}^i_\ell$ for all $i \in \{1, \ldots n\}$. Let v_f be associated with a tuple of controllers that stabilize the robots at their ultimate goal states.

Once the joint prepares graph has been constructed, we use a multirobot path planning algorithm called M* [14] to find a path in the joint prepares graph that describes the sequence of controllers to be executed by each robot. M* treats robots as colliding if they simultaneously execute potentially conflicting controllers. Two controllers potentially conflict if there is at least one pair of configurations in the domains of the controllers that would lead to a robot-robot collision.

Specifying the vertex v_s that represents the initial state of the system requires care. Each robot may start in the domain of multiple controllers. Thus, the system could start from any element of a set of exponentially many vertices in the joint prepares graph which may be too large to explicitly construct if the number of robots is large. Instead, the initial vertex v_s is associated with a tuple of dummy controllers


(a) Different controller geometries (b) Different preparing controller

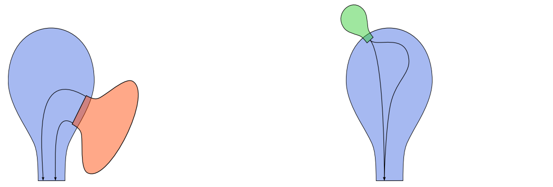

(c) Different positions in goal set of preparing controller (d) Different path due to noise and modeling errors

Fig. 3: Sources of variation in controller duration. **(a)** Larger controllers will typically have longer durations than smaller controllers. **(b)** The duration of a single controller will depend on the preparing controller and **(c)** where in the goal set of the preparing controller the robot starts executing the controller. **(d)** Finally, perturbations of the trajectory during execution ensure that the duration will not be exactly the same even if the robot starts twice in exactly the same spot.

$(\mathcal{C}^1_{\text{dummy}}, \ldots, \mathcal{C}^n_{\text{dummy}})$ where $\mathcal{C}^i_{\text{dummy}}$ prepares every controller $\mathcal{C}^i_k \in \mathfrak{C}^i$ for which $q^i_s \in \mathcal{D}^i_k$. M* will then use heuristics based on the cost of paths for individual robots to construct only a necessary subset of the possible initial combinations of controllers.

III. Synchronization Issues

The construction of the joint prepares graph implicitly contains an assumption that each robot will transition from one controller to its successor at the same time, which would imply that every controller takes the same amount of time to be executed. However, differences in controller geometry and execution errors ensure that the time required to traverse a controller, termed the *duration* of a controller, not only differs among controllers, but has stochastic variation. There are several reasons for differences in duration

1) Controllers have domains of different shapes and command different velocities, and thus require different amounts of time to traverse (Figure 3a).
2) The goal set of different preparing controllers will be at different distances from the goal set of the controller. Therefore, the duration of a controller will depend upon the preparing controller (Figure 3b).
3) The time a robot requires to execute a controller from different configurations within the goal set of the same preparing controller will be different (Figure 3c).
4) A robot will take different amounts of time to execute the same controller from the same starting point due to environmental perturbations, noise in position estimates, and other stochastic influences (Figure 3d).

Reasons 1 and 2 are deterministic in that their contribution to the duration is due solely upon the geometry of the controller and its preparing controllers.[3]

Reasons 3 and 4 are inherently stochastic. Reason 3 is inherent to the fact that sequential composition reasons about controllers, rather than specific robot configurations, while 4 is inherent to non-ideal robots. Properly handling stochastic variation in controller duration requires scalable multirobot path planning in the face of uncertainty, which is left to future work.

IV. Time Augmented Prepares Graph

The Time Augmented Joint Prepares Graph (TAJPG) is intended to account for deterministic differences in the duration of controllers (Section III) in a similar fashion to the region automata of Ulusoy et al. [12]. To do so, the TAJPG tracks the time until each robot is expected to complete its assigned controller, and only assigns new controllers to robots when they complete their previously assigned controllers. To this end, we define the *nominal duration* $t_{nom} : \mathfrak{C}^i \times \mathfrak{C}^i \to \mathbb{R}^+$ over the set of all deployed controllers $\mathfrak{C}^i$ for r^i, where $t_{nom}\left(\mathcal{C}^i_k, \mathcal{C}^i_\ell\right)$ is the time that r^i is expected to require to execute controller $\mathcal{C}^i_k \in \mathfrak{C}^i$ when prepared by $\mathcal{C}^i_\ell$.

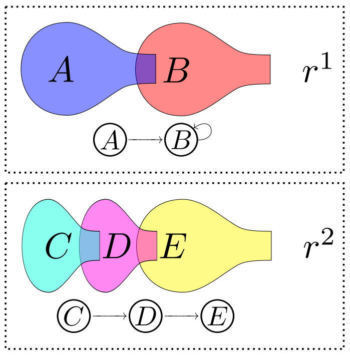
A vertex v_k in the TAJPG is a set of n ordered pairs $\left(\mathcal{C}^i_k, t^i_k\right)$, where $\mathcal{C}^i_k \in \mathfrak{C}^i$ is the controller assigned to r^i, and t^i_k is the expected time-to-go before r^i will finish executing $\mathcal{C}^i_k$. The neighbors of v_k assign new controllers to the robots that will finish their current controllers first, and update the time-to-go for the robots that will take longer to finish executing their assigned controllers. More precisely, denote the minimal time-to-go for any robot as $\delta t_k = \min_i t^i_k$. The neighbors of v_k are then

$$\left\{ v_\ell \;\middle|\; \begin{array}{ll} v^i_\ell = \left(\mathcal{C}^i_m, t_{nom}\left(\mathcal{C}^i_m, \mathcal{C}^i_k\right)\right) & t^i_k = \delta t_k \wedge \mathcal{C}^i_k \succ \mathcal{C}^i_m \\ v^i_\ell = \left(\mathcal{C}^i_k, t^i_k - \delta t_k\right) & t^i_k > \delta t_k \end{array} \right\}. \tag{1}$$

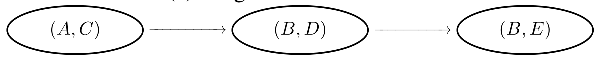
An example of the TAJPG is given in Figure 4. The controllers for robot r^1 have the prepares relation $A \succ B$, and the controllers for r^2 have the relation $C \succ D \succ E$ (Figure 4a). Controllers A, B, and E have a nominal duration of 2 units, while controllers C and D have a nominal duration of 1 unit. The resulting joint prepares graph is given by $(A, C) \to (B, D) \to (B, E)$ (Figure 4b). The transition $(A, C) \to (B, D)$ is not feasible, because A takes longer to execute than C. The TAJPG avoids such problems. The initial vertex v_1 is given by $((A, 2), (C, 1))$. Robot r^2 will finish executing C in 1 unit of time, while r^1 requires 2 units of time to execute A. Therefore $\delta t_1 = 1$. According to equation 1, the neighbor of v_1 is $v_2 = ((A, 1), (D, 1))$, as the nominal duration of D is 1 unit, while r^1 has already been executing controller A for 1 unit of time out of a nominal duration of 2 units. Thus, $\delta t_2 = 1$, with r^1 and r^2 finishing their current controllers at the same time. Therefore, the out-neighbor of v_2 is $v_3 = ((B, 2), (E, 2))$.

While M* could be run on the TAJPG, all existing implementations of M* assume that the graph describing the full system is the product of single robot graphs, which is

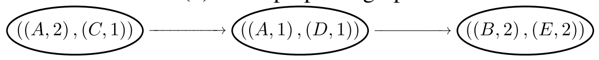
[3]Assuming that the goal set of each controller is small.



(a) Single robot controllers



(b) Joint prepares graph



(c) Time augmented joint prepares graph

Fig. 4: **(a)** Robots r^1 and r^2 have different controller sets and prepares graphs. The larger controllers A, B, and E have a nominal duration of 2, while the smaller controllers C and D can be executed in 1 time unit. **(b)** The joint prepares graph is the direct product of the single robot prepares graphs, and thus ignores differences in nominal duration of the controllers. The transition $(A, C) \rightarrow (B, D)$ is not realistic, because A takes longer to execute than C. **(c)** The time augmented joint prepares graph is formed by augmenting each controller with a time-to-go.

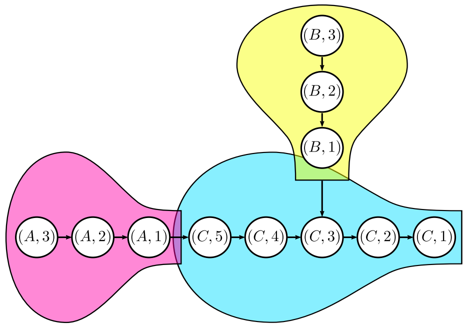
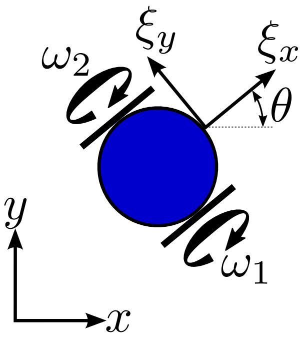


Fig. 5: Example of a approximate time augmented prepares graph with a time resolution of 1 unit. The nominal duration of controller C is 5 units when prepared by controller A and 3 units when prepared by controller B.

not the case for the TAJPG. To simplify implementation, we introduce the *Approximate Time Augmented Joint Prepares Graph* (ATAJPG), which represents the execution of controllers using a fixed temporal resolution. The ATAJPG is the direct product of single robot *approximate time augmented prepares graph*s, in which each controller is represented by a number of vertices determined by its largest nominal duration, recalling that the nominal duration of a controller depends upon its preparing controller. If a controller has a maximal nominal duration of 5 seconds and the time resolution is 1 second, then the controller is represented in the approximate time augmented prepares graph by 5 vertices (Figure 5). The vertices representing a single controller are

(a)



(b)

Fig. 6: **(a)** Parallax Scribbler robot as used in experiments. **(b)** To simplify control design, the body frame was placed at the front of the robot. The scribbler can then be treated as a fully actuated robot in position with heading left uncontrolled.

arranged in a chain, and annotated with the time to go. The last vertex associated with controller $\mathcal{C}_k^i$ (smallest time to go) is connected to the vertices representing controllers $\mathcal{C}_\ell^i, \mathcal{C}_k^i \succ \mathcal{C}_\ell^i$ with a time-to-go equal to $t_{nom}\left(\mathcal{C}_\ell^i, \mathcal{C}_k^i\right)$.

The ATAJPG has up to a constant factor more vertices than the TAJPG and joint prepares graph, where the factor is the maximum number of vertices used to represent a single controller. However, the branching factor of the graph does not increase because all but one of the vertices that represent a given controller have a single neighbor. As a result the worst case computational complexity of finding paths with M* in the ATAJPG will be within a constant factor of finding paths in the TAJPG or joint prepares graph. However, the run time of M* on a specific problem is sensitive to changes in the representation of the problem, even if the representations are functionally equivalent. As a result, the run time of M* is likely to vary significantly and inconsistently when run on the joint prepares graph, TAJPG and ATAJPG.

V. Experimental Setup

The purpose of applying sequential composition to multi-robot path planning is to provide robustness in the face of perturbations during plan execution, which makes validation on real robots essential. Logistical constraints limited the number of physical robots that could be run simultaneously and the size of the available workspace. Therefore, experiments were run in a mixed reality framework, combining physical and simulated robots.

A. Robots

The physical robots were Parallax Scribblers, circular differential drive robots with a radius of 7.2 cm (Figure 6a). Each robot was connected via Bluetooth to a central computer that performed all planning and control. Localization was provided by an overhead camera tracking Aruco tags attached to the top of each robot [6]. The server could optionally synchronize the motion of the robots by reducing the velocity of the robots that were running fast by a factor of σ^η, where σ is the *synchronization factor* and η is the number of steps in the plan the robot is ahead of the slowest robot.

The synchronization factor can be thought of as turning multiple, single robot controllers into a simple multirobot controller.

The purpose of this paper is to explore planning for multiple robots, rather than controller design. Therefore to simplify controller design we follow [2] and place the body frame at the front of the robot, instead of directly between the wheels as in the unicycle model (Figure 6b). With the offset body frame the Scribbler can be treated as a fully-actuated planar robot, leaving θ uncontrolled. The kinematics of the robot in its body frame are then given by

$$\begin{bmatrix} \dot{\xi}_x \\ \dot{\xi}_y \end{bmatrix} = \rho_{wheel} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} \tag{2}$$

where $\dot{\xi}_x$ and $\dot{\xi}_y$ are the body velocities in the x and y directions, ρ_{wheel} is the radius of a wheel and ω_1 and ω_2 are the angular velocities of the right and left wheels, respectively.

B. Controller Design

Controllers are velocity fields in the world frame that are defined over square domains following the work of Habets and Van Schuppen [8]. An affine velocity field over a triangular domain is fully defined by the velocity at the vertices of the triangle. A continuous, piecewise-affine field can be generated over a convex polygon by triangulating the polygon and specifying the velocity at each vertex of the polygon. To ensure that the robot will only exit the domain through the goal face, the velocities at the vertices of the polygon must be chosen to have a positive component in the direction of the outward pointing normal of the goal face while pointing into the polygon at every face not adjacent to the goal face. We choose to rescale the resulting velocity fields to have constant magnitude.

wait controllers stabilize a robot at the centroid of the domain using linear attractive fields with magnitude proportional to the distance from the centroid. Each wait controller is assigned a nominal duration. When a robot begins executing a wait controller, it starts a clock and will not begin executing the next controller in its plan until it has spent at least the nominal duration in the wait controller.

C. Test Environment

The Scribblers were deployed in a 1.6x1.2 meter workspace. The workspace was covered with a regular square grid, where each cell was 0.15 meters on a side, slightly more than one Scribbler diameter. Up to five controllers were deployed in each square, four controllers each using a different face of the square as a goal face, and one wait controller. Controllers were only constructed if they would prepare at least one other controller, *i.e.* the goal face of the controller must be shared with at least one other square. A controller prepares every controller in the square that shares its goal face, except for the controller which would immediately return the robot to its original square. The simulated robots were deployed in a workspace centered on the physical workspace and twice as large.

Controllers are deemed to interfere if they share a vertex. This guarantees a clearance of two Scribbler radii between the origins of the body frames of the robots. To guarantee that no collisions occur a clearance of four radii would be required. Unfortunately, the workspace for the physical robots is not large enough to require a four radii clearance. From a practical standpoint, we have never observed a collision between robots that were executing controllers that did not share a vertex.

D. Calibration

Constructing the ATAJPG requires knowledge of the nominal duration of each controller. Calibration was performed by driving the physical and simulated robots through 10 laps of a closed course. Non-wait controllers were divided into two categories depending on the relative geometry of the controller and its preparing controller: straight and turn. A controller was labeled a straight controller if its goal face was parallel to the goal face of its preparing controller, and was otherwise labeled as a turn controller. Note that the same controller may be assigned both labels when prepared by different controllers. The physical robots took 0.85 ± 0.12 seconds to execute a straight controller, and 0.87 ± 0.16 seconds to execute a turn controller. The velocity of the simulated robots was adjusted so that they executed a straight controller in an average of 0.88 ± 0.03 seconds and a turn controller in 0.59 ± 0.10 seconds. The simulated robots were more responsive, and could thus cut corners more aggressively, resulting in reduced time to execute turn controllers.

We choose to use a time resolution of 0.3 seconds for the ATAJPG, and set the nominal duration of each controller to the mean duration for robots with the same duration and round to the nearest whole number of vertices. The nominal duration for physical robots executing straight and turn controllers and for the simulated robots executing straight controllers was 0.9 seconds, and so these controllers were represented by three vertices. The simulated robots took approximately 0.6 seconds to execute a turn controller, and so those were represented by two vertices.

VI. Results

We tested multirobot sequential composition on problems involving up to four physical robots and four simulated robots (Figure 7). The physical robots were placed at the vertices of a rectangle within the workspace of the physical robots. The simulated robots were placed at the vertices of an outer rectangle. The goal for each physical (respectively simulated) robot was to swap positions with the diametrically opposite physical (resp. simulated) robot. This induced a double 'X' pattern, leading to mutual interaction between all robots.

M* uses a heuristic to guide path planning. The heuristic can be weighted by a factor $\epsilon \geq 1$. Doing so generally allows a plan to be found much more quickly, but may increase the cost of the resulting path by a factor of up to ϵ [15]. Previous work on M* showed that the increase in path cost

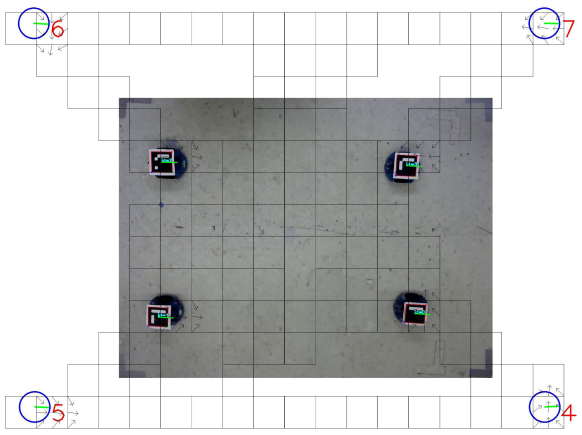
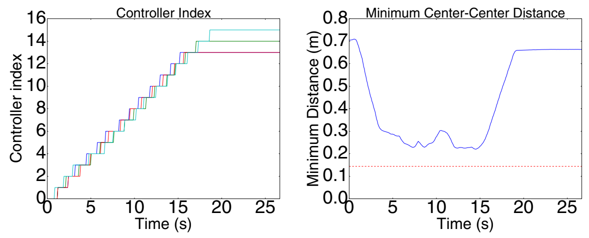
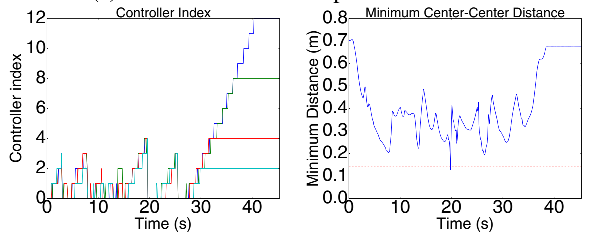


Fig. 7: Initial configuration of the eight robots. The empty blue circles with a red number are the simulated robots. Robots seek to move to the diametrically opposite position. The squares are the domains of the controllers used in one solution to the problem.



(a) 4 robot trial without perturbations



(b) 4 robot trial with large perturbations

Fig. 8: **(a)** The controller indices and minimum distance between any two robots without shoving/swapping robots and **(b)** with shoving/swapping robots. The controller index is the index of the controller in a robot's plan that the controller is executing at a given time. If the robots were perfectly synchronized, they would always have the same controller index. If robots are closer together than 2 Scribbler radii (red dashed line) they are in collision. The one apparent collision in **(b)** occurred while manually swapping robot positions, and does not represent a failure of the algorithm.

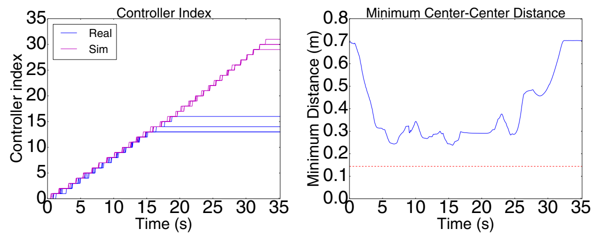
was generally small, so we used $\epsilon = 4$ for the mixed reality tests.

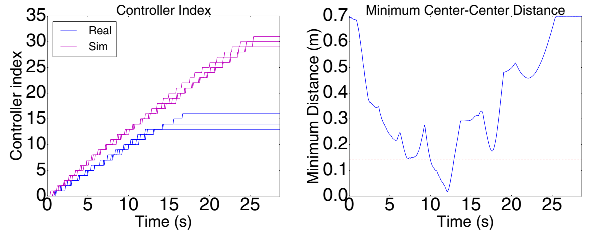
The first experiment focused on the robustness of multirobot sequential composition in the face of large environmental perturbations. In the first trial, we ran just the four physical robots without external disturbances, using a synchronization factor of 0.7. The experiment was then repeated, but during plan execution the experimenter first shoved the robots with a pole, and then later in the trial picked up and swapped the positions two pairs of robots (see the video attachment). Multirobot sequential composition was able to accurately identify when the perturbations were large enough to invalidate the previously computed plans and immediately planned a new path. As a result, the robots safely avoided collisions, despite robots being unexpectedly displaced across the whole breadth of the workspace. We plot the index in the path of the controller being executed by each robot at every instant in time and the minimal distance between the centers (not body frames) of the robots (Figure 8). When replanning occurred the controller indices went to zero. There is one apparent collision in the large perturbation case (Figure 8b), but that occurred while the robots were being manually swapped and not while the system was under control of the planner.

The second experiment investigated the impact of the ATAJPG using all of the physical and simulated robots. We tested three conditions; planning on the joint prepares graph with a synchronization factor of 0.3, planning on the joint prepares graph with no synchronization, and planning on the ATAJPG with no synchronization. We ran each condition three times, however the results for each trial were very similar so we only plot results for the first trial. Computing a plan in the joint prepares graph took an average of 0.4 seconds. The plans generated by M* would be safe if executed exactly as planned. To track how closely the robots adhere the plan we plot the index in the path of the controller being executed by each robot at every instant in time and the minimal distance between the centers (not body frames) of the robots (Figure 9).

When M* plans in the joint prepares graph and robots are run with a synchronization factor of 0.3 the robots remain almost perfectly synchronized (Figure 9a), and as a result stay a safe distance from one another. When the synchronization is removed, the simulated robots execute their plans significantly faster than the physical robots (Figure 9b). The resulting synchronization errors leads to one grazing collision between a real and simulated robot at 7 seconds and a serious collision where a simulated robot almost completely overlapped a real robot for several seconds starting 10 seconds into the run. We believe that the difference in speed comes from the simulated robot being able to execute turn controllers more quickly than physical robots, whereas planning on the joint prepares graph implicitly assumes that the time required to execute each controller is the same.

The ATAJPG accounts for deterministic differences in execution time for different controllers, including whether they are executed by real or simulated robots. As a result, even without explicit synchronization the robots are almost as coordinated when executing a plan computed in the ATAJPG (Figure 9c) as they were when executing a plan computed on the joint prepares graph with active synchronization (Figure 9a). We note that the physical robots showed great consistency; the paths followed in all three replicates of a given trial were nearly identical which likely contributed to the efficacy of planning on the ATAJPG. Planning on the ATAJPG took 4.5 seconds on average, 11 times as long


(a) M* planning on the joint prepares graph with 0.3 synchronization factor


(b) M* planning on the joint prepares graph with no synchronization

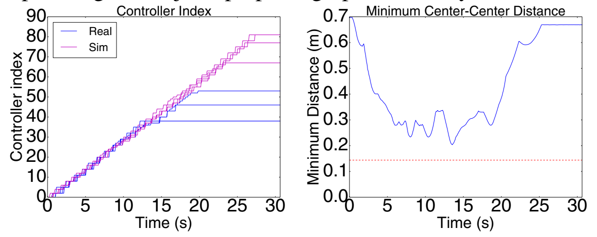
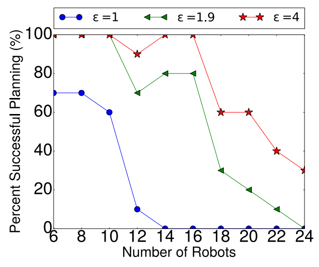

(c) M* planning on the ATAJPG with no synchronization

Fig. 9: The controller indices and minimum distance between any two robots. The controller index is the index of the controller in a robot's plan that the controller is executing at a given time. If the robots were perfectly synchronized, they would always have the same controller index. If robots are closer together than 2 Scribbler radii (red dashed line) they are in collision.


(a) Planning Performance

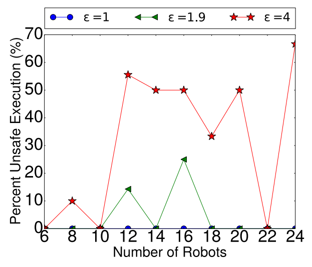

(b) Trials with Collisions

Fig. 10: Performance of M* and the ATAJPG in scaling experiments of between 6 and 24 simulated robots. **(a)** shows the percentage of trials for which M* was able to find a solution within 10 minutes using a heuristic weight ϵ of 1, 1.9, and 4.

as planning in the joint prepares graph even though each controller was represented by at most 3 vertices in the ATAJPG. We attribute the extra run time to M* identifying and resolving more potential collisions when planning in the ATAJPG, taking more time to plan but producing more robust paths.

We then ran experiments to examine how M* and the ATAJPG scale as the number of robots increase. We generated 10 sets of random start and goal configurations for systems of between 6 and 24 simulated robots using the same controllers as the simulated robots in the mixed reality experiments. Only simulated robots were used to facilitate running many trials and because the workspace for the physical robots was too small to accommodate large numbers of robots. We then ran M* with ϵ values of 1, 1.9, and 4, limiting initial planning time to 10 minutes. The percentage of trials for which M* was able to find a solution within the time limit is plotted in Figure 10a. As expected, higher weighting factors allowed M* to solve substantially larger problems.

We then plot the percentage of trials that feature at least one robot-robot collision (Figure 10b). The plans generated by M* with a large heuristic weight are substantially less safe than those with a low heuristic weight. Recall that robots are not allowed to occupy adjacent controllers. As a result a robot must take two steps to one side to get out of the way of another robot. When $\epsilon = 4$ M* operates in a near-greedy fashion, and it is sometimes locally cheaper for one robot to push a second robot backwards, rather than having both robots take a step to the side to pass around each other. This results in a longer path in which the robots remain in close proximity to one another for a prolonged period of time. As a result, non-deterministic variation in controller duration as well as modeling errors in the ATAJPG become more important leading to collisions between robots. In contrast, when ϵ is smaller M* is less greedy and spends more time looking for lower-cost paths, producing shorter paths where robots remain in proximity to one another for shorter periods of time. These better paths reduce the importance of non-deterministic variation in controller duration, leading to greater safety.

VII. Conclusions

In this paper, we showed how to combine planning and control for systems of multiple robots using the sequential composition framework and M*. We adapted region automata to the multirobot sequential composition framework to address deterministic differences in the duration of controllers, producing the TAJPG. To allow direct reuse of existing implementations of M* we show that the TAJPG can be approximated as the direct product of single robot approximate time augmented prepares graphs, producing the ATAJPG.

We validate our results on experiments involving eight agents, of which four were physical robots and four were simulated robots. We show that multirobot sequential composition is robust to large perturbations, including swapping the positions of robots on opposite sides of the workspace. Furthermore, planning on the ATAJPG generated plans that could be safely executed open loop in the absence of large perturbations.

There are two major issues with the proposed approach that were not addressed in this paper. The proposed approach does not address stochastic variation in controller duration and when designing controllers there is a robustness/path length trade-off. The ATAJPG only captures deterministic differences in duration between controllers. Capturing stochastic variation in duration requires a multirobot path planner that can reason about uncertainty. We propose to leverage prior work on single robot belief-space planning [7, 13] to enable M* to plan with uncertainty. In belief space planning, the position of each robot is represented by a belief distribution rather than a single coordinate, and constraints are placed on the maximal likelihood of collision. Running a path planner in belief space can produce a solution more quickly than solving the full partially observable Markov decision process.

In general, using controllers with larger domains will produce more robust paths as they will tend to increase the separation between robots and increase the size of perturbations required to force a robot out of its assigned controller. However, maintaining large separations will lead to longer paths and may make finding paths difficult or impossible if multiple robots must pass through a narrow bottleneck in both directions. One possible resolution would be to deploy overlapping controllers with varying sizes. If traversing the same distance using large controllers is made cheaper than traversing the same distance using small controllers then M* will automatically prefer to use large, robust controllers when possible. However, M* will still be able to fall back on the smaller controllers if necessary to find a path or if using the larger controllers would require excessively long detours. Adjusting the relative costs of the large and small controllers would allow tuning of the robustness/path length trade-off.

This work was supported by the MURI ANTIDOTE program and the ONR Subcontract to the Applied Physics Lab entitled Autonomous Unmanned Vehicles Applied Research Program Prime Contract Number N00024-13-D-6400.

REFERENCES

[1] Nora Ayanian and Vijay Kumar. Decentralized feedback controllers for multi-agent teams in environments with obstacles. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1936–1941. Ieee, may 2008. ISBN 978-1-4244-1646-2.

[2] Nora Ayanian and Vijay Kumar. Decentralized feedback controllers for multi-agent teams in environments with obstacles. *IEEE Transactions on Robotics*, 26(5): 878–887, oct 2010.

[3] Calin Belta, Volkan Isler, and George J Pappas. Discrete abstractions for robot motion planning and control in polygonal environments. *IEEE Transactions on Robotics*, 21(5):864–874, 2005.

[4] Robert R Burridge, Alfred A Rizzi, and Daniel E Koditschek. Sequential composition of dynamically dexterous robot behaviors. *International Journal of Robotics Research*, 18(6):534–555, 1999.

[5] David C Conner, Howie Choset, and Alfred A Rizzi. Integrated planning and control for convex-bodied nonholonomic systems using local feedback control policies. In *Robotics: Science and Systems II*, pages 57–64, Philadelphia, PA, aug 2006. MIT Press.

[6] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco J Madrid-Cuevas, and Mamuel J Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, jun 2014. ISSN 00313203.

[7] Juan Pablo Gonzalez and Anthony Stentz. Planning with Uncertainty in Position: an Optimal Planner. In *IEEE International Conference on Intelligent Robots and Systems*, 2005.

[8] Luc Habets and Jan H Van Schuppen. A control problem for affine dynamical systems on a full-dimensional polytope. *Automatica*, 40:21–35, 2004. ISSN 00051098.

[9] Joseph Moore and Russ Tedrake. Control synthesis and verification for a perching UAV using LQR-Trees. In *IEEE Conference on Decision and Control*, pages 3707–3714. Ieee, dec 2012. ISBN 978-1-4673-2066-5.

[10] Umashankar Nagarajan, George Kantor, and Ralph Hollis. Integrated motion planning and control for graceful balancing mobile robots. *The International Journal of Robotics Research*, 32(9-10):1005–1029, jul 2013. ISSN 0278-3649.

[11] Russ Tedrake, Ian R Manchester, Mark Tobenkin, and John W Roberts. LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification. *The International Journal of Robotics Research*, 29(8):1038–1052, apr 2010. ISSN 0278-3649.

[12] Alphan Ulusoy, Stephen L Smith, and Calin Belta. Optimal Multi-Robot Path Planning with LTL Constraints : Guaranteeing Correctness Through Synchronization. In M. Ani Hsieh and G Chirikjian, editors, *Distributed Autonomous Robotic Systems*, volume 104 of *Spring Tracts in Advanced Robotics*. 2014. ISBN 9783642551468.

[13] Jur van den Berg, Pieter Abbeel, and Ken Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research*, 30(7):895–913, 2011. ISSN 0278-3649.

[14] Glenn Wagner and Howie Choset. M*: A Complete Multirobot Path Planning Algorithm with Performance Bounds. In *IEEE International Conference on Intelligent Robots and Systems*, sep 2011.

[15] Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219(0):1–24, 2015. ISSN 0004-3702.