Safe Distributed Motion Coordination for Second-Order Systems with Different Planning Cycles

Kostas E. Bekris, Devin K. Grady, Mark Moll and Lydia E. Kavraki*

Abstract

When multiple robots operate in the same environment, it is desirable for scalability purposes to coordinate their motion in a distributed fashion while providing guarantees about their safety. If the robots have to respect second-order dynamics, this becomes a challenging problem, even for static environments. This work presents a replanning framework where each robot computes partial plans during each cycle, while executing a previously computed trajectory. Each robot communicates with its neighbors to select a trajectory that is safe over an infinite time horizon. The proposed approach does not require synchronization and allows the robots to dynamically change their cycles over time. This paper proves that the proposed motion coordination algorithm guarantees safety under this general setting. This framework is not specific to the underlying robot dynamics, as it can be used with a variety of dynamical systems, nor to the planning or control algorithm used to generate the robot trajectories. The performance of the approach is evaluated using a distributed multi-robot simulator on a computing cluster, where the simulated robots are forced to communicate by exchanging messages. The simulation results confirm the safety of the algorithm in various environments with up to 32 robots governed by second-order dynamics.

1 Introduction

It is becoming increasingly commonplace for multiple robots with complex dynamics to operate in the same environment. For instance, consider teams of quadrotors or unmanned ground vehicles navigating through a cluttered environment. Such robots have complex dynamics and cannot stop instantaneously. Glancing collisions will still cause catastrophic damage in these applications. Guaranteeing safety by making sure that none of the robots will collide is therefore of the utmost importance, but the underlying dynamics complicate this challenge. The complexity of planning safe motions for multiple robots in a centralized fashion grows exponentially with the number of robots and quickly becomes impractical. It thus becomes necessary to develop practical decentralized solutions for multi-robot systems with complex dynamics.

This paper focuses on cooperative robots which do not necessarily collaborate to solve a common global task. It presents a distributed motion coordination framework where robots replan their trajectories independently. Although this work does not consider other moving systems beyond the cooperating robots, replanning allows to consider multiple trajectories during each cycle and provides flexibility in changing environments as it has been demonstrated in the literature (Ferguson et al., 2006). During each cycle, a robot executes a previously computed trajectory while planning new ones for the consecutive cycle. To achieve coordination, this work utilizes communication. Towards the end of each cycle, a robot communicates trajectories with its neighbors. Based on the responses of its neighbors, a robot selects a trajectory that is guaranteed to be safe over an infinite time horizon. Each robot tries to make progress towards its own goal, while also allowing other robots to make progress. An example simulation is shown in Figure 1. The duration of a planning cycle can vary per robot and is also allowed to vary from one cycle to the next. The robots are therefore not synchronized and communication of plans can happen at any point. Thus, the robots need to operate safely in the presence of partial information about the plans of their neighbors. An asynchronous, distributed framework is developed that guarantees the safety of all robots in this quite general setup. The framework does not depend on the dynamics of specific systems or the properties of specific planning or control algorithms. Simulations confirm the

*Corresponding authors are Kostas Bekris and Lydia Kavraki. Bekris is with the Department of Computer Science and Engineering at the University of Nevada at Reno, bekris@cse.unr.edu. Grady, Moll, and Kavraki are with the Department of Computer Science at Rice University, {dkg1,mmoll,kavraki}@rice.edu.

safety of the approach under different setups for up to 32 second-order, car-like and airplane-like systems, which have a combined total of 160 degrees of freedom.

1.1 Background

Safety Safety issues for dynamical systems have been studied for many years. Collision-free states that inevitably lead to collisions, regardless of which controls are applied, have been referred to as Obstacle Shadows (Reif and Sharir, 1985), Regions of Inevitable Collision (LaValle and Kuffner, 2001), or Inevitable Collision States (ICS) (Fraichard and Asama, 2004). Efforts to deal with ICS resulted in conservative approximations of ICS sets (Fraichard and Asama, 2004), integration of these approximations with replanning schemes (Petti and Fraichard, 2005) and ICS checkers for various systems (Martinez-Gomez and Fraichard, 2009). The same line of research also provided the following criteria for motion safety (Fraichard, 2007):

(i) a robot must consider its dynamics,
(ii) a robot needs to reason about the environment's future behavior, and
(iii) reason over an infinite-time horizon.

These efforts to characterize ICS sets, however, have not dealt with coordinating robots as the current paper does.

Myopic Approaches A class of methods that can enable a robot to avoid collisions for unknown or dynamic environments are classified as myopic (or reactive) methods. Such methods include the Dynamic Window Approach (Fox et al., 1997; Brock and Khatib, 1999) and Velocity Obstacles (Fiorini and Shiller, 1998; Large et al., 2005). Many of the popular myopic methods do not satisfy the three criteria for motion safety specified in the previous paragraph (Fraichard, 2007). Experimental comparisons have also shown the practical importance of reasoning about ICS in the context of myopic approaches (Martinez-Gomez and Fraichard, 2009). Among myopic methods, path deformation techniques compute a flexible path that is adapted on the fly so as to avoid moving obstacles (Lamiraux et al., 2004; Yang and Brock, 2010), but they do not deal with the ICS issue. The work on Reciprocal Velocity Obstacles (RVOs) (Van den Berg et al., 2008) and Optimal Reciprocal Collision Avoidance (ORCA) (Van den Berg et al., 2009) involves multiple agents which simultaneously avoid collisions one with another and with obstacles. RVOs and ORCA can be used to simulate thousands of moving agents without collisions and achieve this objective without communication. Acceleration-Velocity Obstacles (AVOs) (Van den Berg et al., 2011), as the name suggests, extend RVOs by reasoning over second-order dynamics. While very efficient in practice, ORCA requires to set the preferred velocity to zero in order to guarantee collision avoidance without any communication between the agents. Collisions and deadlocks may also still arise in the recently proposed Hybrid Reciprocal Velocity Obstacles (Snape et al., 2011), although it may require many agents to observe collisions in practice. A related approach (Lalish and Morgansen, 2008) deals with second-order models of a planar unicycle but does not provide guarantees in the presence of obstacles.

Multi-robot coordination has also been approached from a control-theoretic point of view. In many such approaches, obstacles other than the robots themselves are usually ignored. Air-traffic control is an application where such approaches have been applied. For instance, hybrid control methods (Tomlin et al., 1998) provided policies with safety guarantees (Ghosh and Tomlin, 2000; Tomlin et al., 2001). More recent work on conflict resolution in multi-vehicle

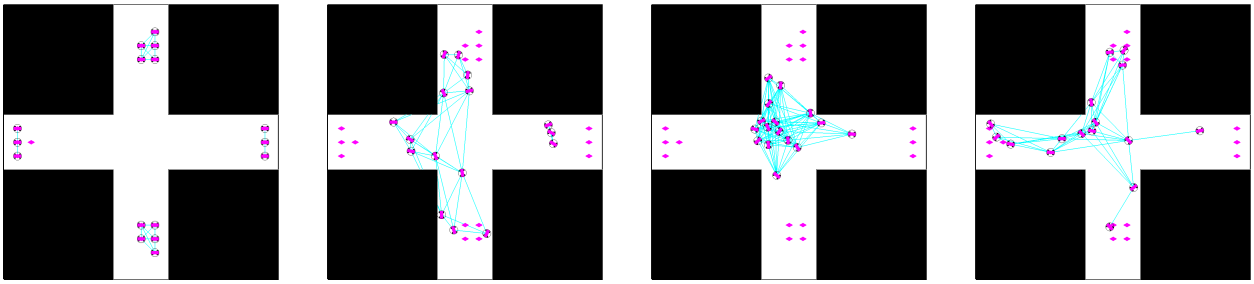


Figure 1: A sample run in an "intersection" environment (left to right). Each robot is trying to reach a different goal location without colliding with other robots. Links show communicating robots at each time step.

systems focused on decentralized, communication-less solutions (Pallottino et al., 2007). Under some assumptions, such policies (Pallottino et al., 2007) guarantee not only safety, but can also provide arguments regarding liveness. Liveness means that the solution will not result in a deadlock or a livelock and that the approach can eventually solve the problem. These methods, however, can guarantee liveness only under certain sparseness assumptions for the goals of individual robots. Hybrid control architectures have also been integrated with prioritized schemes to resolve local conflicts but without liveness guarantees (Azarm and Schmidt, 1997). The paradigm of navigation functions (Rimon and Koditschek, 1992) has led to the development of decentralized feedback control strategies (Dimarogonas and Kyriakopoulos, 2007; Loizu et al., 2004), including techniques for aircraft-like vehicles (Roussos et al., 2010). There are also methods that utilize a receding horizon approach (Li and Cassandras, 2006; Du Toit and Burdick, 2010) or solutions to mixed integer linear programs (Pallotino et al., 2004; Earl and D'Andrea, 2005).

Predictive Approaches In this paper, and in contrast with the above myopic approaches, the focus is on *planning* directly safe paths. Planning has a longer horizon than myopic methods so it does not get stuck in local minima as easily and extends to higher degrees-of-freedom systems. Multi-robot planning can be addressed either with coupled or decoupled approaches. *Coupled* approaches guarantee completeness but with an exponential dependence on the number of robots. Robot networks can utilize coupled planners opportunistically by using them to plan for each connected component of ad-hoc communication networks (Clark et al., 2003). This work did not consider, however, second-order systems or the ICS challenge. *Decoupled* approaches allow robots to compute their own paths and then resolve conflicts. They are usually incomplete but can compute solutions orders of magnitude faster than coupled alternatives. In prioritized planners, high priority robots are considered moving obstacles that must be avoided by lower priority ones (Erdmann and Lozano-Pérez, 1987). The choice of priorities has an impact on solution quality (Van den Berg and Overmars, 2005), and searching the space of priorities can improve performance (Bennewitz et al., 2002). Another decoupled approach tunes the velocities of robots along precomputed paths (Kant and Zucker, 1986; O'Donnell and Lozano-Perez, 1989). Recent variants compute collision-free trajectories for systems with dynamics (Peng and Akella, 2005). Coordination graphs can provide safety and reduce the occurrence of deadlocks (Li et al., 2005; Bekris et al., 2007). Multi-robot planning has also been cast as a search problem in a dynamically changing maze (Lumelsky and Harinarayan, 1997). While most of the above approaches optimize a scalar function, it has been argued that Pareto-optimal solutions may be more appropriate. A characterization of such solutions for multi-robot planning has been provided in the related literature (LaValle and Hutchinson, 1998).

The work in this paper uses a decoupled approach, but in contrast to velocity tuning, it weakly constraints the motion of a robot before considering interactions with neighbors since it considers multiple alternative paths for each robot at each cycle. At the same time, it does not impose a predefined priority to the robots. Instead, robots respect their neighbors in a way that emerges naturally from the lack of a synchronized operation.

Reasoning about safety during the planning process allows a planner to focus on the safe part of the state space. In this work planning and replannning with second-order dynamics is achieved using a sampling-based tree planner (LaValle and Kuffner, 2001; Hsu et al., 2002; Bekris and Kavraki, 2007). Alternatives for the planning process could include, among others, navigation functions (Dimarogonas and Kyriakopoulos, 2007; Philippsen et al., 2008) and lattice-based approaches (Pivtoraiko et al., 2009; Likhachev and Ferguson, 2009).

The literature on contingency planning in static environments shows that braking maneuvers are sufficient to provide safety and were used within a control-based scheme (Wikman et al., 1993) and in sampling-based replanning (Bruce and Veloso, 2006; Bekris and Kavraki, 2007). In the case of dynamic environments most of the existing work considers a relaxation of ICS. For instance, the notion of τ-safety was used in a real-time kinodynamic planner for dynamic environments (Frazzoli et al., 2002). The τ-safety notion guarantees no collision for τ seconds in the future for each node of a sampling-based tree but it does not reason over an infinite-time horizon. A kinodynamic sampling-based planner was tested on real air-cushioned robots moving in dynamic environments, where an escape maneuver was computed when the planner failed to find a complete trajectory to the goal (Hsu et al., 2002). Learning-based approximations of an ICS set can also be found (Kalisiak and Van de Panne, 2007), as well as approximations for computing space×time space obstacles (Chan et al., 2008). Other works focus on the interaction between planning and the sensing limitations of a robot, and point out that it is necessary to limit planning within the robot's visibility region, since the visibility boundary may be hiding moving obstacles (Alami et al., 2002; Vatcha and Xiao, 2008).

The current paper extends earlier work of the authors (Bekris et al., 2009), who pursued the integration of a

kinodynamic sampling-based planner with ICS avoidance schemes (Bekris and Kavraki, 2007) to safely plan for multiple robots that formed a network and explored an unknown environment. In the previous work, the robots followed a synchronous planning operation, which simplified the coordination process. It also extends the most recent work along this direction (Grady et al., 2011) as described in the next section.

In summary, myopic and predictive approaches exhibit different characteristics. On one hand, myopic approaches are closer to achieving real-time behavior as they can provide updated control inputs with high frequency. Nevertheless, most of these methods are designed for specific systems and while they exhibit very good behavior for the systems they have been designed for, they cannot be easily used for general dynamic models. Furthermore, in the context of motion coordination, they tend to minimize the information needed in order to achieve collision avoidance. On the other hand, predictive methods can indeed utilize general dynamic models but require longer computation times to return an updated plan for a system. Due to a longer horizon, they are typically able to better avoid local minima in the planning process compared to myopic alternatives.

1.2 Contributions

The paper deals with independent but communicating second-order robots that move to reach their destinations in an otherwise known environment. A general framework is presented that does not depend on specific robot dynamics. The framework is fully distributed and relies on communication between the robots where the robots' replanning cycles are not synchronized. Furthermore, the robots have no knowledge about their clock differences, and no access to a global clock. The framework is based on the exchange of contingency plans between neighboring robots that are guaranteed to be collision-free. While contingency plans have been used in the past to provide safety for individual agents (Hsu et al., 2002; Petti and Fraichard, 2005), this line of research shows the usefulness in communicating these plans in multi-robot scenarios and studies specifically the case where the robots are not synchronized. A proof that shows that the proposed scheme guarantees ICS avoidance is provided. The framework has been implemented on a distributed simulator, where each robot is assigned to a different processor and message passing is used to convey plans. The experiments consider various scenarios involving 2 to 32 robots and demonstrate that safety is indeed achieved in scenarios where collisions are frequent if the ICS issue is ignored. The experiments also evaluate the efficiency and the scalability of the approach.

This paper extends recent work by the authors (Grady et al., 2011) in several ways. Most significantly, this paper further generalizes the motion coordination algorithm to the case where the length of planning cycles is different per robot and can be changed dynamically. In short, the motion coordination algorithm described in this paper can guarantee safety with respect to both static obstacles and other agents for heterogeneous robots that can differ in almost any aspect: dynamics, planning algorithm, and planning cycle length. A new proof of safety is provided under these conditions. The motivation for allowing the planning cycles to change comes from work on replanning for individual robots that has shown that adapting the duration of cycles can provide completeness guarantees (Hauser, 2010). Secondly, this paper also improves the upper bound on the robots' velocity required for safety. The bound is now computed per robot depending on their varying dynamic cycle rather than globally. This allows robots to move at higher velocities. Thirdly, the algorithmic framework now includes a pairwise payoff scheme that allows neighboring robots to vote on several plans proposed by a robot, whereas in the previous work robots could only accept or reject one proposed plan. This change allows robots to choose plans that are locally sub-optimal, but closer to globally optimal. Finally, the simulation results have been significantly enhanced. They demonstrate that the motion coordination algorithm still maintains safety for robots with different and dynamic planning cycles. This paper also includes results for plane-like vehicles, which require more complex contingency maneuvers than the car-like vehicles considered previously.

2 Problem Statement

Consider robots operating in the same known workspace with static obstacles. Each **robot** A exhibits drift and must satisfy non-holonomic constraints expressed by differential equations of the form:

$$\dot{x} = f(x, u), \quad g(x, \dot{x}) \leq 0, \tag{1}$$

where $x \in \mathcal{X}$ represents a **state**, $u \in \mathcal{U}$ is a **control** and f, g are smooth. The set $\mathcal{U}$ corresponds to the **control space** of the robot. The subset of the **state space** $\mathcal{X}$ that does not cause a collision with static obstacles is denoted as $\mathcal{X}_{free}$. Throughout the paper, the superscripts i, j will be used to refer to the parameters of specific robots A^i and A^j (e.g., state $x^i(t)$ denotes the state of robot A^i at time t). The specific robot models used in the experiments can be found in Section 5 and involve acceleration controlled car-like systems, including versions with minimum positive velocity.The algorithms discussed in this work do not require all the robots to follow the same dynamics.

2.1 Multi-robot Planning with Dynamics

The following notation will be useful for the following discussion:

- A **trajectory** $\pi[t, \Delta t] : \ [t, t + \Delta t] \subset \mathbb{R}^+ \rightarrow \mathcal{X}$ assigns states to the time interval from t to $t + \Delta t$. A trajectory is **feasible** as long as its derivative satisfies the constraints in Equation 1.
- A feasible trajectory $\pi^i[t, \Delta t]$ for robot A^i is **collision-free** with respect to the static obstacles if:
$$\forall \ t' \in [t, t + \Delta t] : \quad \pi[t, \Delta t](t') \ \in \ \mathcal{X}^i_{free}.$$
- The expression $x^i(t) \asymp x^j(t)$ means that A^i in state $x^i(t)$ does not collide with A^j at state $x^j(t)$. Such states will be called **compatible states**.
- Two trajectories $\pi^i[t^i, \Delta t^i]$ and $\pi^j[t^j, \Delta t^j]$ for A^i and A^j are **compatible trajectories**, denoted as $\pi^i[t^i, \Delta t^i] \asymp \pi^j[t^j, \Delta t^j]$, if:
$$\forall \ \ t \ \in [max(t^i, t^j) : min(t^i + \Delta t^i, t^j + \Delta t^j)] : \ \ \pi^i[t^i, \Delta t^i](t) \asymp \pi^j[t^j, \Delta t^j](t).$$
Based on this definition, if two trajectories do not overlap in time, then they are by default compatible.

Given the above notation, the problem of **multi-robot planning with dynamics** can be defined as follows: Consider m robots operating in the same workspace among obstacles. Each robot's motion is governed by second-order dynamics, which need not be the same for all of them, specified by differential constraints similar to Equation 1. Furthermore:

- At time 0, A^i is located at x^i_0, where $x^i_0 \in \mathcal{X}^i_{free}$ and $\forall i, j : \ x^i_0 \asymp x^j_0$, i.e., the robots are initially collision-free.
- There are also goal regions $\mathcal{X}^i_g$ for each robot A^i, where $\mathcal{X}^i_g \subset \mathcal{X}^i_{free}$ and
$$\forall i, j : \forall x^i_g \in \mathcal{X}^i_g \text{ and } \forall x^j_g \in \mathcal{X}^j_g : x^i_g \asymp x^j_g,$$
i.e., the goal regions do not bring the robots into collision with obstacles and are pair-wise disjoint.

Then the objective is to compute feasible trajectories $\pi^i[0, T]$ of finite duration T with the following properties:

– $\pi^i[0, T](0) = x^i_0$ and $\pi^i[0, T](T) \subset \mathcal{X}^i_g$, i.e., the plan brings the robot from its start location to its goal region within finite time T,

– $\forall \ i, \ \ \forall t \ \in [0 : T] : \ \ \pi^i[0, T](t) \in \mathcal{X}_{free}$, i.e., the resulting trajectories are collision-free with static obstacles,

– and $\forall \ i, j : \ \ \pi^i[0, T] \asymp \pi^j[0, T]$, i.e., the resulting trajectories are pairwise compatible from the beginning and until all the robots reach their goals.

2.2 Decentralized Replanning Version

As discussed in Section 1, this paper adopts a decentralized approach for scalability purposes. Instead of velocity tuning and fixed prioritization, the robots coordinate on the fly within a replanning framework. Towards this objective, each robot's operation is broken into time intervals $([0 : t_1], [t_1, t_2], \ \ldots, [t_n : t_{n+1}], \ldots)$ called cycles. A cycle $[t_{n-1} : t_n]$ will also be identified as: δ_n. The duration of a cycle will be denoted as: $|\delta_n| = (t_n - t_{n-1})$.

The robots will not be synchronized. In other words, the cycles among different robots do not coincide, i.e., it is *not* necessarily the case that $\forall \ i, j, n, \exists \ m : t^i_n = t^j_m$. Furthermore, this paper allows the duration of the various cycles of each robot to vary, i.e., it is *not* necessarily the case that $\forall n, m |\delta^i_n| = |\delta^i_m|$. Throughout this paper, subscripts will be used to differentiate between cycles of the same robot. Also, the following notation will be useful:

- A trajectory that returns the states robot A^i goes through during the cycle δ_n^i will be abbreviated as π_n^i.
- The trajectory A^i follows during the cycle δ_n^j of a different robot A^j will be denoted as $\pi^i[\delta_n^j]$.

The robots are equipped with an omnidirectional, range-limited communication ability, which can be utilized for coordination. The set of all robots within communication range of A^i at time t is defined as the neighborhood $N^i(t)$. When robot A^i broadcasts a message in time t, then this message is forwarded to all robots in $N^i(t)$. The algorithm presented in this work does not require that messages arrive on time or that they arrive at all. Thus, the approach can deal with a certain level of latency, i.e., delay in the arrival of a message. It is important, however, for two vehicles to know that they should be able to communicate if they are within communication range. This can be achieved through sensing. Thus, if two systems are within communication range and cannot communicate because their messages are dropped, the control algorithm can follow a conservative approach and take measures so as to guarantee safety.

Furthermore, the execution of the trajectories by the robots is assumed perfect. This means that if A^i has decided to execute the trajectory π_n^i, then it already knows at time t_n^i the state $x^i(t_n^i)$ it is going to reach at the end of the planning cycle δ_n^i. State $x^i(t_n^i)$ is the initial state for the consecutive cycle δ_{n+1}^i.

Given the decomposition of time into replanning cycles and the availability of communication, each robot has to solve the following **decentralized replanning version of the multi-robot planning with dynamics problem**:

During each cycle δ_n^i, robot A^i must select a feasible trajectory π_{n+1}^i with the following properties:

- $\pi_{n+1}^i(t_n^i) = x^i(t_n^i)$,
- $\pi_{n+1}^i \in \mathcal{X}_{free}$,
- $\forall\ j : \pi_{n+1}^i \asymp \pi^j[\delta_{n+1}^i]$.

As with the original problem: $\pi_0^i(0) = x_0^i$. Furthermore, the desired outcome is that eventually there is a point in time T so that $\forall t \geq T : x^i(t) \subset \mathcal{X}_g^i$. Nevertheless, convergence cannot be guaranteed in the general case for a decentralized setup.

3 A Simple Framework without Safety Guarantees

Given the above setup, Algorithm 3.1 outlines a straightforward decentralized and incremental approach for the single cycle operation of each robot to solve the multi-robot planning with dynamics problem. The reader should be aware that this algorithm is used to present the challenges faced in decentralized motion coordination and a modified version of this framework will be presented in the next section. Each cycle δ_n^i of a robot is split into two parts: (a) a planning part of duration $(|\delta_n^i| - \epsilon)$ and (b) a compatibility check part of duration ϵ.

During the first, planning part of the cycle (lines 3–7), robot A^i generates a set of alternative, collision-free partial trajectories Π^i (lines 4–5) for the consecutive cycle δ_{n+1}, given knowledge of the future initial state $x^i(t_{n+1}^i)$. For example,

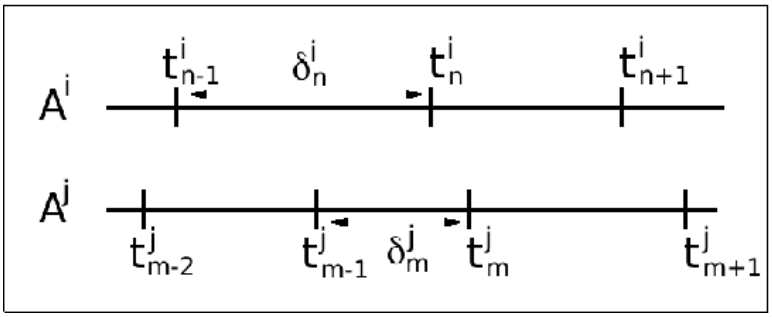


Figure 2: An illustration of how time is decomposed into planning cycles of different duration for different robots. δ_n^i is the n-th planning cycle for the i-th robot of duration $|\delta_n^i| = t_n^i - t_{n-1}^i$.

Algorithm 3.1 Simple but Unsafe Operation of A^i During Cycle $\delta_n^i = [t_{n-1}^i : t_n^i]$

1: $\Pi^i \leftarrow \emptyset$ (set of candidate trajectories for A^i)
2: $\Pi^{N^i} \leftarrow \emptyset$ (set of communicated trajectories by robots in the neighborhood N^i)
3: **while** $t < t_n^i - \epsilon$ **do**
4: $\quad \pi_{n+1}^i \leftarrow$ a new collision-free trajectory from a single-robot planner for δ_{n+1}
5: $\quad \Pi^i \leftarrow \Pi^i \cup \pi_{n+1}^i$
6: $\quad$ **if** $A^j \in N^i$ is transmitting a trajectory π^j intersecting the time interval δ_{n+1} **then**
7: $\quad\quad \Pi^{N^i} \leftarrow \Pi^{N^i} \cup \pi^j$
8: **for all** $\pi^i \in \Pi^i$ **do**
9: $\quad$ **for all** $\pi^j \in \Pi^{N^i}$ **do**
10: $\quad\quad$ **if** $\pi^i \not\asymp \pi^j$ (incompatible trajectories) **then**
11: $\quad\quad\quad \Pi^i \leftarrow \Pi^i - \pi^i$
12: $\pi^{*i} \leftarrow$ trajectory in Π^i which brings A^i closer to the goal given a metric
13: Transmit π^{*i} to all neighbors in N^i and execute π_*^i during next cycle

a sampling-based planner can be used in order to produce these trajectories. In parallel, A^i listens for messages from robots in neighborhood N^i that contain their selected trajectories (line 6). The neighbors' trajectories are inserted into a set Π^{N^i} (line 7).

Then, during the second part of the cycle and when time passes $(t_n^i - \epsilon)$ (lines 8–13), A^i executes first a compatibility check between trajectories in Π^i and Π^{N^i} (lines 8–11). Note that the planning cycles of different robots may not coincide and may be changing dynamically, so A^i has no knowledge when the states returned by Π^{N^i} occur in time. Thus, in order to execute the compatibility check of line 10, i.e., to find out whether $\pi^i \in \Pi^i \not\asymp \pi^j \in \Pi^{N^i}$ is true or not, the following conservative operation is executed:

$$\pi^i \not\asymp \pi^j \text{ if } \exists\, t^i, t^j \text{ so that } \pi^i(t^i) \not\asymp \pi^j(t^j). \tag{2}$$

In other words, if there exists a state along trajectory π^i that brings robot A^i in collision with robot A^j in a state along trajectory π^j, the two trajectories are considered incompatible, even if in reality these states occur in different points in time. Every trajectory $\pi^i \in \Pi^i$ found to be incompatible even with one trajectory $\pi^j \in \Pi^{N^i}$ using Equation 2 is removed from the set Π^i (line 11). Thus, such incompatible trajectories will never be considered by the neighbors of robot A^i. Among the remaining compatible trajectories, which at this point are both collision-free and compatible with the neighbors' communicated trajectories, the algorithm selects the trajectory π^{*i} that brings the robot closer to its goal (line 12). The time ϵ has to be enough for the robot to be able to execute the compatibility check and the selection of trajectory π^{*i}. It corresponds to a lower limit for the duration of a cycle $|\delta_n^i|$.

If such a trajectory π^{*i} is found and there is progress to the goal during each iteration, then the problem is eventually solved by this straightforward planning-based approach. The following discussion will focus on the safety properties of this high-level decentralized and incremental approach. A discussion in Section 6 will focus on the completeness properties of this framework.

4 A Safe Solution to Distributed Motion Planning with Dynamics

A robot following the above approach might fail to find a trajectory π^{*i} because the set Π^i might end up being empty by the time that a solution trajectory must be selected. This section describes a similarly decentralized algorithm that guarantees the existence of at least one collision-free, compatible trajectory for all robots at every cycle.

4.1 Safety Considerations: Inevitable Collision States

It is possible that the set Π^i is empty because the single-robot planner failed to find collision-free trajectories. This is guaranteed to happen when the state $x^i(t_n^i)$ is an `ICS` with regards to static obstacles. State $x(t)$ is an `ICS` with regards to

static obstacles if:

$$\forall \text{ feasible } \pi[t:\infty] \text{ so that } \pi[t:\infty](t) = x(t) : \exists\, t' \in [t,\infty) \text{ so that } \pi[t:\infty](t') \notin \mathcal{X}_f.$$

Computing whether a state $x(t)$ is ICS is intractable, since it requires reasoning over an infinite horizon for all possible trajectories out of $x(t)$. It is sufficient, however, to consider conservative methods that identify states that are *not* ICS (Fraichard and Asama, 2004; Bekris and Kavraki, 2007). The approximation reasons over a subset of predefined maneuvers. The corresponding **contingency trajectories** generated by executing a contingency maneuver at a state $x(t)$ will be denoted as $\gamma[t:\infty]$, where $\gamma[t:\infty](t) = x(t)$. The set of all contingency trajectories at state $x(t)$ will be denoted as $\Gamma[t:\infty]$. If A^i can avoid future collisions with static obstacles at a state $x^i(t_n^i)$ by following a collision-free contingency trajectory $\gamma^i[t_n^i:\infty] \in X_{free}^i$ over an infinite time horizon past t_n^i, then $x^i(t_n^i)$ is not ICS with regards to static obstacles:

$$x^i(t_n^i) \text{ is not ICS iff } \exists\, \gamma^i[t_n^i:\infty] \in \Gamma^i[t_n^i:\infty] \text{ so that:}$$
$$\gamma^i[t_n^i:\infty](t_n^i) = x^i(t_n^i) \text{ and } \forall t' \in [t_n^i:\infty] : \gamma^i[t_n^i:\infty](t') \in \mathcal{X}_{free}^i.$$

For cars, braking maneuvers are sufficient to provide safety as contingency maneuvers. Periodical maneuvers (e.g., circling maneuvers) can be used for systems with minimum velocity limits, such as airplanes. For such maneuvers, braking or periodical, it is computationally feasible to reason over an infinite time horizon whether they lead to collisions with static obstacles.

Multiple moving robots pose additional challenges. Two trajectories π_n^i and $\pi^j[\delta_n^i]$ may be compatible for cycle δ_n^i, but the corresponding robots A^i and A^j may reach states $x^i(t_n)$ and $x^j(t_n)$ that will inevitably lead them in a future collision. Thus, safety notions have to be extended into the multi-robot case. It is still helpful for computational reasons to be conservative and focus only on a predefined set of contingency maneuvers.

To describe how contingency plans are used to prove that a state is safe in the multi-robot case, it is useful to define a **trajectory concatenation** with a contingency plan. Assume a robot A^i is following a trajectory $\pi^i[\delta_n^i]$ during a cycle δ_n. After the completion of $\pi^i[\delta_n^i]$ at time t_n, A^i switches to a contingency trajectory $\gamma^i[t_n^i:\infty]$. Then the trajectory concatenation $\pi\gamma^i[\delta_n^i]$ is the trajectory A^i is following from time t_{n-1} up to infinity so that:

$$\pi\gamma_n^i[\delta_n^i](t) = \begin{cases} \pi^i[\delta_n](t), & \forall t \in \delta_n^i, \\ \gamma^i[t_n^i:\infty](t), & \forall t \geq t_n^i. \end{cases} \tag{3}$$

As with simple trajectories, the abbreviation $\pi\gamma_n^i$ will be used to denote $\pi\gamma^i[\delta_n^i]$.

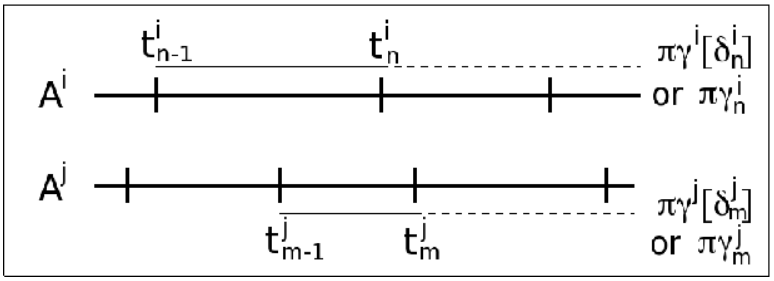


Figure 3: The trajectory $\pi\gamma^i[\delta_n^i]$ corresponds to the trajectory $\pi^i[\delta_n^i]$ followed by the contingency $\gamma^i[t_n^i:\infty]$. It is also abbreviated as $\pi\gamma_n^i$. Similarly for $\pi\gamma^j[\delta_m^j]$ and $\pi\gamma_m^j$.

Consider m robots $\{A^1, A^2, \ldots, A^m\}$ at states $\{x^1(t), x^2(t), \ldots, x^m(t)\}$. Furthermore, assume that the robots are committed at that point to execute trajectories $\{\pi^1[\delta^1], \pi^2[\delta^2], \ldots, \pi^m[\delta^m]\}$. The cycles δ^j are different in duration but all of them overlap at time t. Then state $x^i(t)$, for a different robot A^i not in the set of m robots, is considered a **safe state** if $\exists\, \gamma^i[t:\infty] \in \Gamma^i[t:\infty]$ so that:

a) $\gamma^i[t:\infty](t) = x^i(t)$ and $\forall\, t' \in [t,\infty) :\ \gamma^i[t:\infty](t') \in \mathcal{X}_f$ and

b) $\forall\ j \in [1,m],\ \exists\ \gamma^j[t:\infty]:\ \ \gamma^i[t:\infty] \asymp \pi\gamma^j[\delta^j]$.

The above expression requires that the state $x^i(t)$ is not an ICS relative to static obstacles and that there is a contingency trajectory $\gamma^i[t:\infty]$ out of this state which is compatible over an infinite horizon with the currently selected trajectories $\pi^j[\delta^j]$ of the other robots, as well as with their future contingencies that can be applied after the execution of their current trajectories. In addition to the requirement that the starting states are compatible in the original formulation of the problem, the following discussion will assume that the initial states of all the robots are also safe states. Then an algorithm is needed that also maintains the following invariant for each robot and planning cycle:

Safety Invariant The selected trajectory π^i_n:

a) Must be collision-free with obstacles: $\forall t' \in \delta^i_n : \pi^i_n(t') \in \mathcal{X}_{free}$.

b) Must be compatible with all other robots, during the cycle δ^i_n:

$$\pi^i_n \asymp \pi^j[\delta^i_n],\ \ \forall j \neq i.$$

c) And the resulting state $x^i(t^i_n)$ is safe for all possible future trajectories $\pi^{*j}[t^i_n:\infty]$ selected by other robots ($j \neq i$) using the same algorithm. In other words, the concatenation of trajectory π^i_n with a contingency $\gamma^i[t^i_n:\infty]$ must be compatible with the concatenations of trajectories π^j_m of other vehicles with their contingencies $\gamma^j[t^j_m:\infty]$, where the cycle δ^j_m is the one containing time t^i_n:

$$\forall j \neq i:\ \text{when } t^i_n \in \delta^j_m : \pi\gamma^i_n \asymp \pi\gamma^j_m.$$

Point c) above means that A^i has a contingency plan at $x(t^i_{n+1})$, which can be safely followed given the other robots' choices when they follow the same algorithm. If the invariant holds for all the robots, then they will always be safe. If for any reason a robot cannot find a plan that satisfies these requirements, then it can revert to its contingency that guarantees its safety.

4.2 Safe and Asynchronous Distributed Solution

Algorithm 4.1, in contrast to Algorithm 3.1, maintains the above safety invariant. The protocol follows the same high-level framework and still allows a variety of planning techniques to be used for producing trajectories. Figure 4 provides a diagram of the communication taking place between the two agents. The differences with the original algorithm can be summarized as follows:

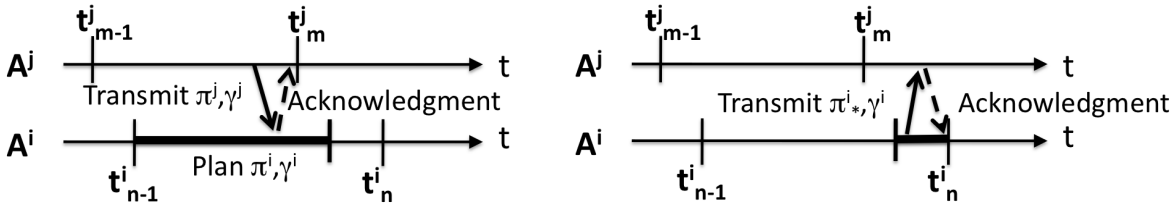


Figure 4: The communication taking place between two agents under Algorithm 4.1. A robot A^j before the end of cycle δ^j_m transmits not only the selected trajectory π^j_{m+1} for the consecutive planning cycle δ^j_m but also attaches the future contingency $\gamma^j[t^j_{m+1}:\infty]$. In order to actually follow π^j_{m+1}, A^j needs to receive acknowledgments from all of its neighbors before time t^j_m. If these acknowledgments do not arrive on time, then A^j reverts to a contingency $\gamma^j[t^j_m:\infty]$.

- The algorithm stores the last messages received from neighbors during the previous cycle in the set $\Pi^{N^i}_{prev}$ (lines 2 and 4–5). Note that the robots transmit the selected trajectory together with the corresponding contingency (lines 11, 14, 16, 18, 24).

Algorithm 4.1 Safe and Asynchronous Operation of A^i During Cycle $\delta_n^i = [t_{n-1}^i : t_n^i]$

$\Pi^i \leftarrow \emptyset$ (set of candidate trajectories for A^i)
$\Pi_{prev}^{N^i} \leftarrow \emptyset$ (set of previously communicated trajectories by robots in the neighborhood N^i)
$\Pi_{new}^{N^i} \leftarrow \emptyset$ (set of newly communicated trajectories by robots in the neighborhood N^i)
for all $A^j \in N^i$ **do**
    $\Pi_{prev}^{N^i} \leftarrow \Pi_{prev}^{N^i} \cup \pi\gamma_m^j$
    (i.e., include the latest communicated trajectories and attached contingencies of neighbors in $\Pi_{prev}^{N^i}$)
while $t < t_n^i - \epsilon$ **do**
    $\pi_{n+1}^i \leftarrow$ a new collision-free trajectory from a single-robot planner for δ_{n+1}
    $\pi\gamma_{n+1}^i \leftarrow \pi_{n+1}^i + \gamma[t_{n+1}^i : \infty]$ (concatenation with the contingency trajectory)
    if $\forall\, t \in [t_{n+1}^i : \infty) :\ \pi\gamma_{n+1}^i(t) \in \mathcal{X}_f$ **then**
        $\Pi^i \leftarrow \Pi^i \cup \pi\gamma_{n+1}^i$
        for all $\pi\gamma_m^j \in \Pi_{prev}^{N^i}$ **do**
            if $\pi\gamma_{n+1}^i \not\asymp \pi\gamma_m^j$ **then**
                $\Pi^i \leftarrow \Pi^i - \pi\gamma_{n+1}^i$
    if $A^j \in N^i$ is transmitting a trajectory and an attached contingency **then**
        Transmit acknowledgment to A^j
        $\Pi_{new}^{N^i} \leftarrow \pi\gamma_{m+1}^j$
for all $\pi\gamma_{n+1}^i \in \Pi^i$ **do**
    for all $\pi\gamma_{m+1}^j \in \Pi_{new}^{N^i}$ **do**
        if $\pi\gamma_{n+1}^i \not\asymp \pi\gamma_{m+1}^j$ **then**
            $\Pi^i \leftarrow \Pi^i - \pi\gamma_{n+1}^i$
if Π^i empty or if a message was received during compatibility check **then**
    $\pi^{*i} \leftarrow \gamma[t_n : \infty]$ (i.e., follow the available contingency for next cycle)
else
    $\pi^{*i} \leftarrow$ trajectory and attached contingency in Π^i which brings A^i closer to the goal given a metric
Transmit π^{*i} to all neighbors in N^i
if acknowledgments received by all neighbors in N^i **then**
    Execute π^{*i} during next cycle
else
    Execute $\gamma[t_n : \infty]$ and communicate it to the neighbors

- A contingency plan $\gamma[t^i_{n+1} : \infty]$ is attached to every collision-free trajectory π^i_{n+1} and the trajectory concatenation $\pi\gamma^i_{n+1}$ is generated (line 8). Note that potentially multiple different contingencies can be attached to the trajectory π^i_{n+1}. Each resulting trajectory concatenation is treated individually by the algorithm.
- The trajectory $\pi\gamma^i_{n+1}$ is added to Π^i only if it is collision-free with static obstacles for an infinite time horizon (lines 9–10), thus guaranteeing that $x^i(t^i_{n+1})$ is not `ICS`.
- $\pi\gamma^i_{n+1}$ can be rejected later, however, if it is not compatible with all the trajectories and contingencies of neighbors. There are two compatibility check procedures (lines 11–13 and 17–20), and A^i needs to check against the last transmitted trajectory and contingency of a neighbor as well as any new message that arrives during the planning process.
- The final change (lines 21–29) addresses the case that Π^i is empty or when a message arrives while A^i executes its compatibility check. If any of the two is true, then A^i selects to follow the contingency $\gamma[t^i_n : \infty]$, which was used in the previous cycle to prove that $x(t^i_n)$ was safe (line 22). Otherwise, A^i selects among the set Π^i the trajectory π^{*i} that brings it closer to the goal according to a desired metric (line 24). This trajectory is transmitted to the neighbors. If an acknowledgment is received on time by all the neighbors, then trajectory π^{*i} will be executed during the next cycle (line 27). Otherwise, the robot must still resort to $\gamma[t^i_n : \infty]$ (line 29).
- The while loop (lines 4–13) is executed as long as time t is less than the end of the planning cycle (t^i_n) minus an ϵ time period. Time ϵ should be sufficient for the robot to complete the compatibility check (lines 17–20) and the selection process (lines 21–29). If the robot is running out of time, the robot should immediately select a contingency in order to guarantee safety. In a real robot implementation, this can be achieved through an interrupt or a signal that stops execution and enforces the contingency.

Overall, each robot selects a trajectory π^i_{n+1} and a contingency $\gamma^i[t^i_{n+1} : \infty]$ that respect the trajectories and contingencies of other robots that have been selected before time t^i_n. If no such plan is found or there is no time to check against newly incoming messages, then the contingency $\gamma^i[t^i_n : \infty]$ is selected to be executed during cycle δ^i_{n+1}.

Computational Complexity The complexity of the algorithm depends on the number of neighboring robots N^i, which in the worst case could be the total number of robots N. In order to represent the cost of operations involving trajectories, it is important to consider a representation for a trajectory. One way to represent a trajectory is through a discrete sequence of states, which are selected at a given temporal resolution Q (i.e., the technique becomes resolution-safe in this case). Let S denote the upper limit in the number of states used to represent each trajectory concatenation. For the case of braking maneuvers, this upper limit can be computed as

$$S = \frac{|\delta|}{Q} + \frac{\frac{v_{max}}{\alpha}}{Q}.$$

The first term on the righthand-side corresponds to the number of states used to represent a collision-free trajectory for a planning cycle of duration $|\delta|$. The second term is the upper limit in the number of states needed to represent a braking contingency, where v_{max} is the maximum velocity of the system and α its maximum de-acceleration. A similar upper limit can be computed for periodical contingencies. Finally, let the upper limit in the number of plans considered during each planning cycle for the current agent to be denoted as P.

Given the above notation, the complexity of the algorithm's various operations is as follows:

(a) *Lines 4–5:* $S \times N$,

(b) *Lines 7–10:* $P \times S$,

(c) *Lines 11–13:* $P \times N \times S^2$ (if the states in a trajectory are not accompanied by a global timestamp) or $P \times N \times S$ (if the states are tagged with a global timestamp),

(d) *Lines 17–20:* Same as in c,

(e) *Lines 21–24:* P, assuming constant time for computing a cost-to-go metric for each trajectory,

(f) *Line 25:* $N \times S$.

Overall, the worst-case complexity is: $P \times N \times S^2$. Note that for robots with limited communication, the parameter N is reduced. Furthermore, coarser resolution in the representation of trajectories helps computationally but relaxes the safety guarantees. Similarly, considering fewer plans reduces computational complexity but reduces the diversity of solutions considered at each time step. Finally, lower maximum velocity or higher maximum de-acceleration also assist computationally in the case of braking maneuvers.

4.3 Guaranteeing Maintenance of the Safety Invariant

This section provides a proof that Algorithm 4.1 maintains the safety invariant of Section 4.1 given the assumption that all robots are always able to communicate. This assumption will be waived later. Through the use of acknowledgments, the following theorem proves safety even when messages fail to reach the neighbors of a robot or when there is significant latency.

Theorem 1. *For two robots A^i and A^j, which are always able to communicate and which execute the proposed coordination protocol, when they start at safe states $x^i(t_0^i)$ and $x^j(t_0^j)$ it is true $\forall \delta_n^i, \delta_m^j$, where $\delta_n^i \cap \delta_m^j \neq 0$, that:*

$$\pi\gamma_n^i \asymp \pi\gamma_m^j.$$

Proof. The theorem can be proved by induction. For the *base case* and without loss of generality assume that $t_0^j \leq t_0^i$. This means that robot A^j will start planning first. During their first planning cycle, δ_1^j and δ_1^i correspondingly, the robots will be executing a contingency until a trajectory is computed by the algorithm at times t_1^j and t_1^i correspondingly. What is necessary to show for the base case is that $\pi\gamma_1^i \asymp \pi\gamma_n^j, \forall \delta_n^j$, where $\delta_n^j \cap \delta_1^i \neq 0$. Given that $x^i(t_0^i)$ is a safe state, then from the definition of a safe state and for the planning cycle δ_m^j where $t_0^i \in \delta_m^j$ the following is true: $\gamma^i[t_0^i : \infty] \asymp \pi\gamma_n^j$. But for the very first cycle: $\pi\gamma_1^i = \gamma^i[t_0^i : \infty]$, since during the cycle δ_1^i robot A^i will be executing the contingency trajectory $\gamma^i[t_0^i : \infty]$ and the contingency at time x_1^i, is the same trajectory. So the base case is true as long as the initial states of the robots are safe states.

For the *inductive step*, it is necessary to show that if:

$$\pi\gamma_{n-1}^i \asymp \pi\gamma_m^j, \forall \delta_m^j, \text{ where } \delta_{n-1}^i \cap \delta_m^j \neq 0,$$

then it is also true that:

$$\pi\gamma_n^i \asymp \pi\gamma_{m'}^j, \forall \delta_{m'}^j, \text{ where } \delta_n^i \cap \delta_{m'}^j \neq 0.$$

There are two choices available to A^i at time t_{n-1}^i: (1) Either A^i is able to follow a trajectory π_n^i during δ_n^i different than a contingency trajectory that has been acknowledged by the neighbors, or (2) for some reason the robot A^i was forced to resort to the contingency $\gamma^i[t_n^i : \infty]$ (e.g., the set Π^i in the algorithm was empty or an acknowledgment was not received on time).

Case 1: π_n^i has been acknowledged. There are also two types of trajectories that A^j may be executing during δ_n^i:

Type 1.a) A trajectory π_m^j that its execution started before t_{n-1}^i and may still be executing during part of δ_n^i. The plan p_m^j may correspond either to an acknowledged trajectory or to a contingency trajectory:

- If p_m^j has been acknowledged by A^i, then this must have occurred before t_n^i. Thus, based on the algorithm, the selected trajectory π_n^i of A^i at time t_n^i must have the following property: $\pi\gamma_n^i \asymp \pi\gamma_m^j$.
- Alternatively, A^j may be executing a contingency $\gamma^j[t_{m-1}^j : \infty]$. Robot A^i, however, is aware of this contingency trajectory at time t_n^i. The contingency was transmitted either as the extension to a previously acknowledged trajectory before time t_n^i or A^i never acknowledged a selected trajectory by A^j, and the last robot had to resort to this contingency that was already respected by A^i. Thus, again: $\pi\gamma_n^i \asymp \gamma^j[t_{m-1}^j : \infty]$.

Type 1.b) A trajectory $\pi^j_{m'}$ that started after t^i_n. There can be multiple trajectories of similar nature if $|\delta^j_m| < |\delta^i_{n+1}|$. Again, the trajectory $\pi^j_{m'}$ can correspond either to an acknowledged trajectory or to a contingency one:

- Assume A^j plans and follows a trajectory $\pi^j_{m'}$. Since A^i is assumed to be executing π^i_n that is not a contingency, it has to be that A^j acknowledged and is aware of this trajectory before time t^i_n, i.e., before the selection of trajectory $\pi^j_{m'}$. This means that when A^j selects $\pi^j_{m'}$, it has to be true that: $\pi\gamma^i_n \asymp \pi\gamma^j_{m'}$.
- Assume A^j executes a contingency $\gamma^j[t^j_{m'-1} : \infty]$. If it is the first time that A^j reverts to a contingency, it has to be that the previous trajectory was acknowledged by A^i and this contingency was known by A^i and respected during the generation of $\pi\gamma^i_n$. If A^j was already in a contingency trajectory and there was at some point a trajectory acknowledged by A^i, this was also the case. If A^i never acknowledged any trajectories of A^j, then this contingency corresponds to the very first one that was used from state $x^j(t^j_0)$, which is assumed to be a safe state. In this case, it will also be known by A^i. In every case: $\pi\gamma^i_n \asymp \gamma^j[t^j_{m'-1} : \infty]$.

Case 2: A^i follows a contingency trajectory $\gamma^i[t^i_{n-1} : \infty]$. There are again the same types of plans that robot A^j may be executing during δ^i_n:

Type 2.a) A trajectory that its execution started before t^i_{n-1}. The trajectory could again correspond either to a selected, acknowledged trajectory π^j_m or a contingency $\gamma^j[t^j_{m-1} : \infty]$. Similar to the arguments as in the second subcase of 1.b, the contingency $\gamma^i[t^i_{n-1} : \infty]$ had to be known by A^j upon its execution. Thus, if π^j_m was selected, it was respecting the contingency $\gamma^i[t^i_{n-1} : \infty]$. Otherwise, if contingency $\gamma^j[t^j_{m-1} : \infty]$ is followed, either one of the two robots started executing a contingency first and from that point on the contingency was respected by the other robot, or the two contingencies are the same as the ones used to prove the safety of the original states $x^i(t^i_0)$ and $x^j(t^j_0)$.

Type 2.b) A trajectory $\pi^j_{m'}$ that its execution started after t^i_{n-1}. The same arguments as in 2.a hold true for all possible plans of the form $\pi^j_{m'}$, as at the moment they are selected A^j is aware of the contingency $\gamma^i[t^i_{n-1} : \infty]$.

Overall, regardless of whether a trajectory is planned and acknowledged or a contingency is selected, at each planning cycle and regardless the duration of each planning cycle, A^i can safely execute the algorithm's outcome. At the end of each planning cycle, the contingency is also guaranteed to be safe into the future. □

Theorem 1 assumed that all the robots communicate one with another. It is more realistic, however, to assume that robots can communicate only if their distance is below a certain threshold. In this case, the proposed approach can be invoked using only point to neighborhood communication and thus achieve higher scalability.

For two robots with limited communication range, denote without loss of generality time t^i_n the beginning of the first planning cycle of either robot after they are able to communicate. Assume at t^i_n robot A^i has a contingency $\gamma^i[t^i_n : \infty]$ available so that $\gamma^i[t^i_n : \infty] \asymp \pi^j[t^j_m : \infty]$, where $t^i_n \in \delta^j_m$. Then robots A^i and A^j are pairwise safe at states $x^i(t^i_n)$ and $x^j(t^j_m)$. This means that it is possible to identify $x^i(t^i_n)$ as state $x^i(t^i_0)$ from the previous theorem and $x^j(t^j_m)$ as state $x^j(t^j_m)$ from the previous theorem. Thus, as long as $\gamma^i[t^i_n : \infty] \asymp \pi^j[t^j_m : \infty]$ at time t^i_n, when two robots start to communicate, then it is still possible to use the arguments of the previous theorem and show safety. The next section shows how it is possible to guarantee the existence of such a contingency for the case of car-like vehicles using braking maneuvers by limiting the maximum velocity of the systems given their communication range.

4.4 Communication Constraints and Maximum Velocity Limits

For car-like vehicles with a limited communication range it is necessary to impose bounds on their maximum velocity in order to guarantee safety. Consider the worst case situation between two car-like vehicles, which are located just outside their communication ranges and head towards each other with maximum velocity. For braking contingencies, it is possible to limit the maximum velocity that the systems can achieve so as to guarantee that they always have enough time to decelerate at the point of first communication. Prior work (Grady et al., 2011) used the following equation:

$$v_{max} = -2\alpha + \sqrt{\alpha(\mathbb{E} - \mathbb{S} + 4\delta^2)},$$

where δ is the duration of a planning cycle, $\mathbb{E}$ is the communication range of the robots, $\mathbb{S}$ is the diameter of the robots, and α is the maximum (de)acceleration the robots can sustain.

The above bound assumed a common and global planning cycle. The maximum velocity in the case of different and dynamic planning cycles is more complex to compute. Analyzing worst case behavior in terms of stopping distance as prior work did is not directly applicable because planning cycles of other agents are not known.

Instead, it is possible to utilize the maximum planning cycle allowed over all robots. A simpler analysis is also sufficient, where it is possible to take only a purely local view. Instead of calculating the stopping distance for each robot, the analysis assumes that each robot has available half of the communication radius to stop in. In this way, each robot can locally reason about its local parameters and still be safe. The only parameter that is assumed known and identical globally is the communication radius and an upper bound on the planning cycle duration. The maximum velocity is thus subject to the following constraint:

$$v_{max} \cdot \delta_{max} + v_{max} \cdot \delta_{max} + d_{cont} = \frac{\mathbb{E} - \mathbb{S}}{2} \tag{4}$$

Each term on the left hand side is the following:

1. The distance that will be traveled in a planning cycle of maximum length at maximum speed while out of communication range $v_{max} \cdot \delta_{max}$.
2. Another full planning cycle at maximum speed once the robots have communicated. These allows them to have enough time to decide whether to enter into a contingency or compute compatible trajectories $v_{max} \cdot \delta_{max}$.
3. The distance the robot requires to enter contingency d_{cont}.

In the case of the car-like robots modeled in Section 5.1, $d_{cont} = \frac{{v_{max}}^2}{2\alpha}$, however robots with other dynamics, particularly robots that cannot come to a stop, must instead use a more complex form that takes into account turning radius, size, minimum and maximum velocity as well as acceleration limits. The other option is to simply provide a conservative bound given some maximum worst-case analysis of d_{cont}.

This distance must sum to at most half the communication range minus the minimum safety distance between robots, given as the right hand side of the equation. All distances and durations fall in the range $[0, \infty)$. Rearranging and solving for v_{max} the following is true:

$$v_{max} = \frac{\frac{\mathbb{E}-\mathbb{S}}{2} + d_{cont}}{2\delta_{max}}.$$

Substituting in the actual dynamics for the car-like system, however, introduces a v_{max} on the right hand side.

$$v_{max} = \frac{\frac{\mathbb{E}-\mathbb{S}}{2} + \frac{{v_{max}}^2}{2\alpha}}{2\delta_{max}}.$$

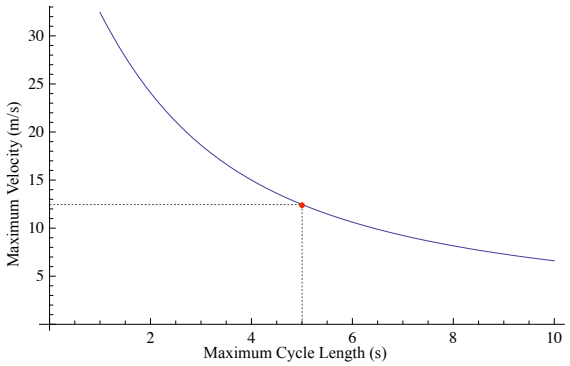
Solving again for v_{max} and assuming a positive square root as the negative solution is not physically meaningful, the following bound is obtained:

$$v_{max} = -2\alpha\delta_{max} + \sqrt{\alpha(\mathbb{E} - \mathbb{S} + 4\alpha\delta_{max}^2)}.$$

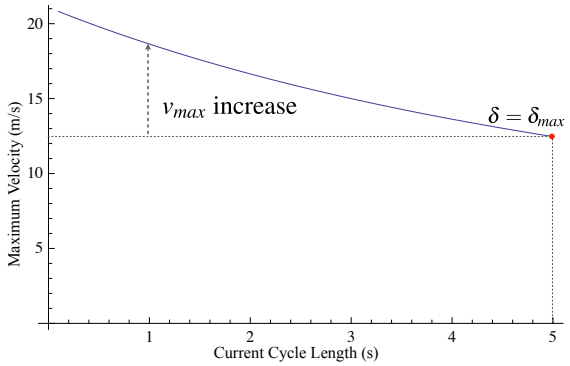
This equation is slightly more complex than the final form used in prior work, but it allows to guarantee a global velocity bound across all robots. Figure 5(a) plots the maximum velocity v_{max} against the maximum cycle length δ_{max}. It shows that the maximum allowed velocity decreases as the allowed range of planning cycles increases to larger δ_{max}.

However, a robot's planning cycle may be much less than the maximum. Even when a robot is not aware of its neighbors' planning cycle, it can still decrease the conservatism of this calculation and maintain safety. It may do this by locally calculating its own v_{max} that is greater than the global maximum allowed. The following equation shows the minor changes needed to locally increase v_{max} on a per planning cycle basis, based on using the actual value of the current planning cycle δ:

$$v_{max}\delta + v_{max}\delta_{max} + d_{cont} = \frac{\mathbb{E}}{2} - \mathbb{S}.$$



(a) Maximum velocity v_{max} as a function of maximum planning cycle duration in seconds.



(b) Maximum velocity v_{max} as a function of planning cycle duration in seconds, assuming a maximum duration of 5.0s.

Figure 5: Maximum velocity computation.

The equation is identical to the one above except it uses the current planning cycle for the first term. If δ is less than the bound δ_{max}, then this distance is less than the maximum cycle times the maximum velocity. Therefore, it is possible to make a less conservative approximation of the velocity bound for the system. It is still necessary, however, to assume a certain maximum speed for the entire cycle. The derivation is only slightly different than above and is not repeated, rather the end result is provided:

$$v_{max} = -\alpha\left(\delta + \delta_{max}\right) + \sqrt{\alpha\left(\mathbb{E} - \mathbb{S} + \alpha\left(\delta + \delta_{max}\right)^2\right)}. \tag{5}$$

Figure 5(b) plots v_{max} with a selected δ_{max} of 5.0s. For a varying δ, the figure shows that robots can move with a higher velocity if they have a shorter planning cycle.

For the current $\delta = \delta_{max}$, the value v_{max} is the same at the marked point on the graphs. This calculation assumes the worst case for future planning cycles, and could still be refined if the planning cycle duration update rule is taken into account. All experiments, unless specified, use this local per-robot computation for v_{max}.

4.5 A Framework for Pairwise Optimization of Paths

In Algorithm 4.1 each robot communicates only a single path to its neighbors. There is no way to detect the existence of a path that might be locally sub-optimal, yet allows a robots' neighbors to make more progress and is therefore better for the system as a whole. This work aims to show that it is feasible to implement a scheme for pairwise optimization using limited communication within the proposed safety framework of Algorithm 4.1. For example, robots can maintain estimates of a global payoff given the choices of their neighbors and try to iteratively improve their local approximation of this global quantity. However, this is complicated by the fact that robots operate on different planning cycles.

The approach considered here is to allow robots to communicate several options to their neighbors and let them provide feedback. In order to enable this, a pairwise optimization step has been introduced. This additional step consists of each robot proposing *several* plans to its neighbors and letting them vote on each plan. The robot then chooses the path which balances its own preference with the preferences from neighboring robots. This addition is completely orthogonal to the safety framework presented here, and as such, it does not affect the safety properties of the system in any way. The main difference is that additional time is required for this communication, i.e., the time parameter ϵ in Algorithm 4.1 needs to be larger. Therefore, the system has less time to plan. This is not problematic, however, for the approach in terms of safety and the variable ϵ can be considered as a tuning parameter in this case between planning time and coordination time. The smaller the ϵ value, the more time each robot has to generate candidate plans and less time to participate in pairwise optimization. The opposite is true for a larger ϵ value. However, a robot now receives feedback from its neighbors about several possible paths.

The feedback can be based on the goal of each individual robot. Each robot maintains a local cost map that guides the planner exploration and final plan selection. Instead of choosing the locally optimal plan right away, first a robot

polls its neighbors about its top K plans. This poll consists of only the Euclidean endpoints of the plans, indicating where the robot would like to travel in the workspace. This was chosen to balance information needed to be able to provide feedback, and keep communication as low as possible.

Upon receipt of a list of endpoints from a neighbor, a robot evaluates each of these endpoints to provide a vote. To do so, it compares these endpoints to the endpoint of its currently executing plan. The exact form of each vote is:

$$vote = max\left(\frac{(maxVoteDist - distance\,(evalPoint, currentPlanEndpoint))\,maxVote}{maxVoteDist}, 0\right).$$

This vote prefers that other robots are at least $maxVoteDist$ away from the voting robot. The vote values range from $maxVote$ at a distance of 0, linearly down to a vote of 0 at a distance of $maxVoteDist$. Each robot computes a local vote using the local cost map:

$$vote = -\frac{(costMap(inputPoint) - costMap(currentLocation))}{max\,(costMap(inputPoint), costMap(currentLocation))}.$$

So the local vote is between -1 and 1 and indicates how much progress that plan would make. All incoming votes are averaged, and then a robot casts this local vote as well and adds it to the result. This has the effect of making all neighbor votes combined be equal in magnitude to a robot's local vote.

Any and all messages in this pairwise payoff scheme may be dropped without affecting safety. A robot cannot tell the difference between another robot deciding not to respond to a poll, a dropped poll message, a dropped vote message, and a delayed vote or poll message beyond the time window that a robot accepts votes. In every case that vote simply does not get counted in the plan evaluation, and in the case that a robot gets no votes it chooses the locally optimal plan.

Experimental results showing the effect of enabling this pairwise payoff computation are shown in Section 5.4. All experiments have this pairwise payoff turned on by default unless specified otherwise.

4.6 Dynamic Cycles

This section presents an extension of the safety framework as presented in Section 4.1 to prove safety under both different and dynamic planning cycles. This extension showcases the generality of the approach with regard to planning strategy, and emphasizes the benefits of not requiring synchronization in the proposed approach. More significantly, dynamic planning cycles can be used to support a liveness argument, as discussed in Section 6.

The default operation of the vehicles presented in the experimental section is with differing but constant planning cycles. The cycles range from 2s to 5s, assigned as follows.

$$\text{Even numbered robots: } \delta^i = 0.1 \cdot (i \bmod 10) + 2,$$
$$\text{Odd numbered robots: } \delta^i = 0.1 \cdot (i \bmod 10) + 4.$$

The scheme provides both very large differences in planning cycle time between the two groups, as well as very small differences within a group. The robots also get random small offsets to their start time, as in previous work (Grady et al., 2011). To test dynamically changing cycle times, the robots are initialized at different points in time and they follow the scheme described above. Each round they also compute an update rule to either increase or decrease their planning cycle:

$$\delta(n+1) = \begin{cases} \min(\delta(n) + c_1\delta(n), \delta_{max}), & \text{if } ProgressMade \wedge {\sim}GoalReached \wedge \left({\sim}MissedAcks \vee \delta(n) < \frac{\delta_{max}+\delta_{min}}{2}\right), \\ \max(\delta(n) - c_2\delta(n), \delta_{min}), & \text{otherwise.} \end{cases}$$

Here, c_1 and c_2 are two constant scaling factors (set to 0.45 and 0.225, respectively, in the implementation). The update rule states that when progress is not being made, the planning cycle should increase in duration. However, if a robot has reached its goal, the planning cycle should not increase anymore so that the robot maintains its responsiveness and decreases the space×time obstacle that it will impose to its neighbors. The update rule also checks if acknowledgments were missed. If they were, then it is probably the primary cause for lack of progress. In this

case a robot does not increase its planning cycle beyond the average of maximum and minimum values. This again helps the robot's responsiveness to its neighbors. A robot will not be able to make progress if it continues to miss acknowledgments, and the longer planning cycles result in longer plans. The longer plans require more processing on the receiving end, therefore, if a robot is missing acknowledgment messages, it should not increase its planning cycle to the maximum. It is possible that an acknowledgment message was missed simply because a neighboring robot chose not to transmit one. This can happen if it is detected that two neighboring robots are transmitting incompatible plans at the same time. In this case, neither robot will acknowledge the other and both will enter contingency to maintain safety by design. In the future, a negative acknowledgment message is under consideration that will allow robots to differentiate between these two cases.

5 Experiments

To validate the theoretical discussion, simulations were conducted. A multi-robot simulator has been developed that runs in parallel, using one core per robot and one core to simulate the world. The experiments were designed to verify the safety of the motion coordination algorithm, measure the efficiency, measure bandwidth requirements, and determine the sensitivity of the performance to a few key parameters. Experiments were run for both car-like and airplane-like vehicles.

5.1 Modeled System

The experiments presented in this paper are using the model of a second-order car-like vehicle (Laumond, 1998):

$$\begin{aligned}
\dot{x} &= v\cos\zeta\cos\theta,\\
\dot{y} &= v\cos\zeta\sin\theta,\\
\dot{\theta} &= v\sin\zeta,\\
\dot{v} &= \alpha,\\
\dot{\zeta} &= \phi.
\end{aligned}$$

Here, (x, y) denote the car's position, θ is the car's orientation, v its velocity and ζ the steering angle. The controls are α, the acceleration, and ϕ, the rate of change of the steering angle. Both the state and control parameters are bounded:

$$|v| < v_{max}, \quad |\zeta| < \zeta_{max} = 0.03, \quad |\alpha| < \alpha_{max} = 7.5, \quad |\phi| < \phi_{max} = 0.025.$$

All robots have range-limited communication, out to 30% of the total environment width, and a braking maneuver to a full stop for contingency. The maximum velocity is given by Equation 5.

Environments Three simulated environments were used for the experiments:

1. An empty environment (Figure 6, left, shown with 32 robots),
2. an office environment consisting mainly of fairly wide corridors and open rooms (Figure 6, middle, shown with 16 robots), and
3. an intersection environment with two crossing corridors (Figure 6, right, shown with 32 robots).

These environments are presented in approximate order of difficulty. All environments have the same overall size: $1000m \times 1000m$. The various experiments tested different numbers of vehicles: 2, 4, 8, 16. The empty and intersection environments were run with 32 robots as well, because start and goal positions could be programmatically assigned. The less structured office environment had all start and goal positions hand-selected. The robots take up 6% of the workspace in the 16 robot experiment. In each environment the robots' goal positions are arbitrarily selected start positions of other robots. Because the obstacles already take up a significant portion of the workspace, for the 32 robot experiments the robot size was halved. This reduces the clutter effects that otherwise dominate the solution times.

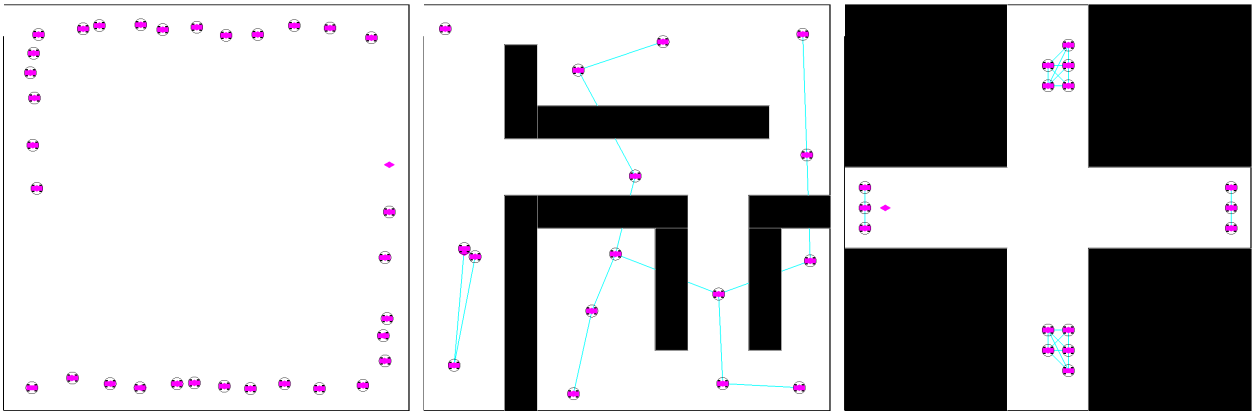


Figure 6: Starting positions for the environments simulated: empty, office, and intersection.

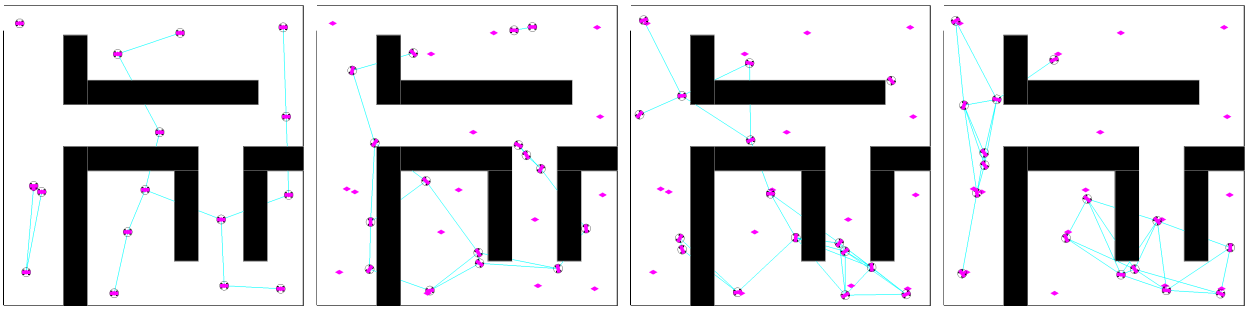


Figure 7: Snapshots from a typical run in the office environment.

The empty environment was the easiest to solve. The office environment was chosen as a gauge for how hard a structured environment can be. An example run is shown in Figure 7. The robots, in their original size $\mathbb{S}$, are about $\frac{1}{5}$ of the size of the hallway. The intersection case seemed to be the hardest to solve, since the robots not only have to navigate through a relatively narrow passage together with their neighbors, but they are all forced to traverse the center, almost simultaneously (see Figures 1 and 8).

Where possible, start and goal locations were kept the same across runs as more robots were added. Experiments for the same number of robots have the same start and goal locations. All experiments were repeated 10 times. In each test, the algorithm ran in real time such that simulation time is equal to execution time.

5.2 Implementation Specifics

This section describes some steps to make the implementation of Algorithm 4.1 more efficient computationally. In particular:

- At each step of the "while" loop in Algorithm 4.1 (lines 4–13), the implementation propagates an edge along a tree of trajectories using a sampling-based planner, instead of generating an entire trajectory. If the edge intersects t^i_{n+1}, the contingency $\gamma[t^i_{n+1} : \infty]$ is extended from $x^i(t^i_{n+1})$. If the contingency is collision-free and compatible with the messages of neighbors, state $x^i(t^i_{n+1})$ is proven safe. Otherwise, it is unsafe and no future expansion of an edge is allowed past $x^i(t^i_{n+1})$. This means that it is not necessary to compute contingencies for parts of the plan that originate at a different point in time other than t^i_{n+1}.

- The sampling-based expansion of the tree structure of trajectories is biased:

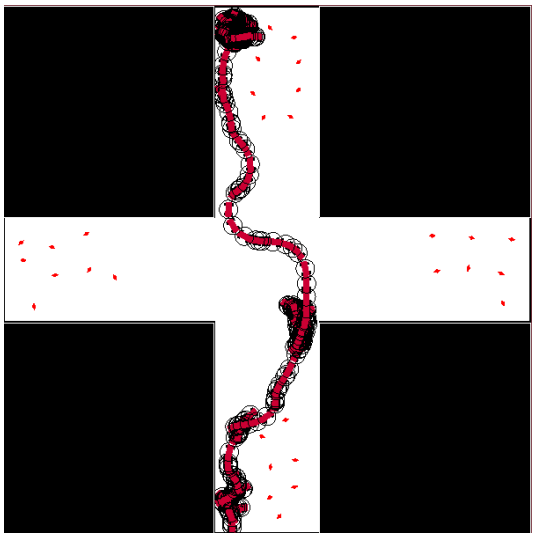


Figure 8: The full trajectory of robot 0. The small diamonds represent the final states of all robots.

  - A potential field in the workspace is used to promote the expansion of the tree towards the goal, while still maintaining probabilistic completeness (Bekris and Kavraki, 2007).
  - To increase the probability of compatible plans, the tree expansion is biased away from other vehicles. This has a significant effect in the algorithm's performance.

  Different sampling-based planners can be easily adapted to be employed in the above framework (LaValle and Kuffner, 2001; Hsu et al., 2002). It is also possible to consider the application of navigation or potential functions (Dimarogonas and Kyriakopoulos, 2007; Philippsen et al., 2008) and lattice-based methods (Pivtoraiko et al., 2009; Likhachev and Ferguson, 2009).

- Once a tree of sufficient size has been expanded, a discrete set of trajectories are extracted from the tree. Instead of checking the compatibility of all possible trajectories Π^i on the tree with the transmitted trajectories of the neighbors, plans on the tree are checked in a greedy best-first fashion until $K = 15$ trajectories are extracted. The endpoints of the trajectories are required to be distance $\mathbb{S}$ (i.e., the robots' size) apart from each other. If no valid trajectories are found by time $x^i(t_n^i)$, a contingency $\gamma^i[t_n^i : \infty]$ is selected for execution. Otherwise, the robot enters the pairwise payoff algorithm, considering the choices of at most a constant number of neighbors.

- In the actual implementation, there is no need to differentiate between $P_{prev}^{N^i}$ and $P_{new}^{N^i}$. Each robot maintains a buffer for messages for each neighbor. As new trajectories are transmitted, they replace the part of old trajectories that has already been executed on the buffer. Assuming no uncertainty, it is easy to identify which part of the past trajectory has been already executed. States are not time-stamped, as no global clock is assumed. This is because the first state along the newly received trajectory must exist on the old trajectory that was actually executed. Note that for each neighbor A^j, a robot A^i must always respect two possible trajectories, the current contingency $\gamma[t_m^j : \infty]$ and the trajectory $\pi\gamma_{m+1}^j$ acknowledged by A^i. The implementation emphasizes the constraint that the robots have no access to a global clock and they do not know when each state along a transmitted trajectory is going to be executed.

- The motion coordination algorithm of Section 4 specifies that robots transmit trajectories, i.e., sequences of states. If trajectories are transmitted between robots, then in order to execute the pairwise collision checking operation (lines 12 and 19 in Algorithm 4.1), it is necessary for a robot to know the geometry of its neighbors. Alternatively, and as discussed in Section 5.5, robots can transmit plans (i.e., sequences of controls) instead of trajectories. Plans are advantageous communication-wise because they typically require a smaller amount of information to

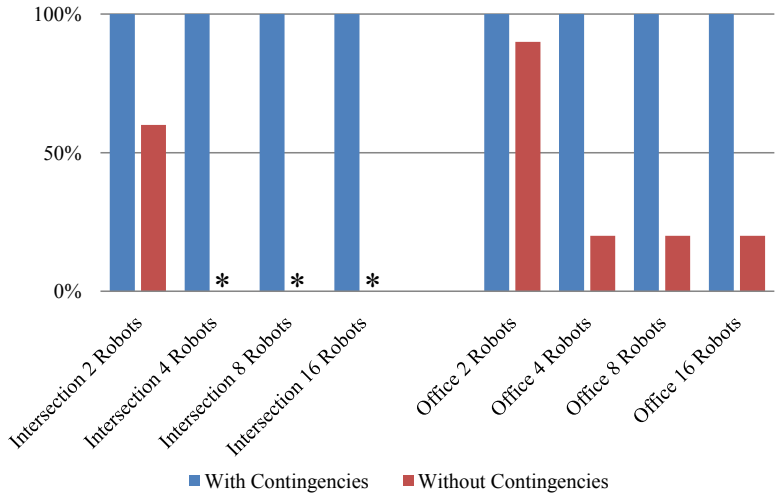


Figure 9: The benefits in safety of the proposed coordination algorithm 4.1 in the case of dynamic cycles that differ between each robot and using the pairwise payoff scheme. The "$*$" indicates that the intersection problem could never be completed with 4, 8, and 16 robots without considering contingencies.

be transmitted compared to trajectories. On the other hand, in order to execute the pairwise collision checking, it is necessary to resimulate the trajectory of a neighbor upon the receipt of its plan. This means that in the case that plans are being transmitted, robots need to also know the dynamics of their neighbors. In any case, the algorithm imposes no requirement that all the robots have the same geometry or dynamics. Heterogeneous teams of robots (geometry and dynamics wise) can use this approach to coordinate in a safe manner.

- It is important to note that even in the simulation environment employed for this work, it is difficult to perfectly synchronize the operation of all robots. This shows the importance of considering motion coordination algorithms without the need for synchronization.
- Unlike previous work (Grady et al., 2011) that assumed low latency communication, acknowledgment messages have been enabled in all results presented here. In the case of different and dynamic cycles it is not uncommon for acknowledgment messages to arrive too late but the approach still provides safety guarantees.
- If v_{max} increases, no part of the tree that was previously generated is invalidated. However, if v_{max} decreases then an unknown portion of the tree is invalidated. In the implementation, the whole tree is invalidated and planning restarts. This does cause a performance hit, but does not affect the algorithm in any other way. A tree checking strategy that only invalidates the required portions of the planning tree has been considered but not currently implemented.

5.3 Evaluation of Safety

To verify that the system implemented truly provides the safety guarantees presented in this paper, two different cases were considered for the algorithm: (i) an implementation without contingencies, (ii) with contingencies. Both implementations had different starting planning cycles, and we would dynamically change them on the fly. If progress was not made, the planning cycles were increased to provide a longer planning horizon. If progress was made, the planning cycle was decreased to take advantage of higher v_{max}. For each type of experiment Figure 9 reports the percentage of successful experiments out of 10 runs per data point. The results presented clearly indicate that enabling contingencies results in a safe system in all cases. Unsurprisingly, the harder intersection scene was solved less often than the empty scene in the experiments without contingencies.

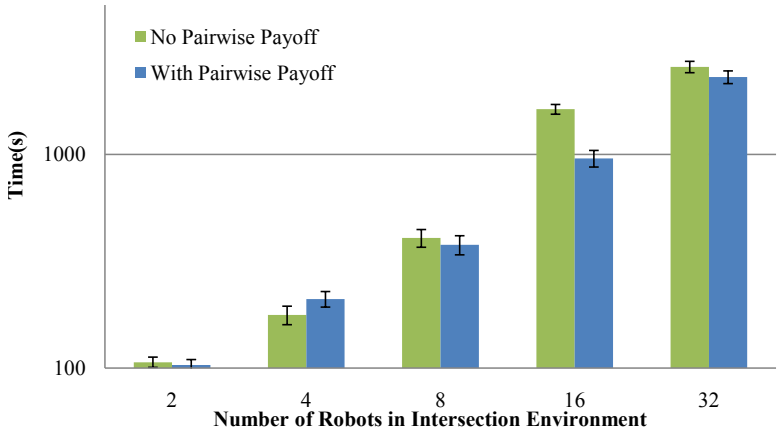


Figure 10: Pairwise payoff performance shown in the intersection environment where robots interact the most, 10 runs per data point. Time is plotted on a log scale.

5.4 Efficiency

Once the safety of the approach was confirmed, the focus turned on evaluating the effects of the pairwise payoff scheme on the efficiency. The efficiency is measured by the time it takes for all robots to reach their goal region. Figure 10 indicates that the pairwise payoff scheme helps most when there are more robots and therefore more inter-robot interactions. Robots are sending to their neighbors their top $K = 15$ plans in the pairwise payoff scheme. At the same time, they are picking trajectories that tend to avoid their neighbors planned paths. Communication costs are bounded by sending the poll message to a bounded set of neighbors. In the experiments presented here, 5 neighbors were polled at most. Although this is only an extremely crude approximation of global payoff, and the task at hand is not explicitly cooperative, it is shown here that the pairwise payoff scheme helps solve the most complex experiments faster.

5.5 Communication Costs

Another important measurement for this system is the communication required. Each robot keeps a log of the number of bytes that it is transmitting to each neighbor, and they are aggregated across the entire experiment to determine the average required bandwidth. As implemented in simulation, all robots perform point to point communication rather than utilize broadcast methods. This means that the total number of bytes transmitted is exactly equal to the total bytes received, although, in a hardware implementation broadcast messages would be expected to significantly reduce the transmitted bytes. There are four types of messages being sent in these experiments:

1. Plan selections,
2. Plan acknowledgments,
3. Multi-plan polls,
4. Multi-plan votes.

The last two message types are not required if pairwise payoff is not enabled. Because only the endpoints of the plans are transmitted for pairwise payoff, these messages do not make as large a difference in the required bandwidth as might otherwise be expected. The measured average bandwidth used per robot is presented in Figure 11.

Even with pairwise payoff enabled and 32 robots operating in the same environment, less than 1.1kB/s average bandwidth per robot is measured. It is expected that most systems will be limited instead by message latency times. The current real-time simulation framework cannot easily measure or enforce message latency because of the lack of synchronization between robots, however, this may be possible in future work and may provide insight into such possible limitations.

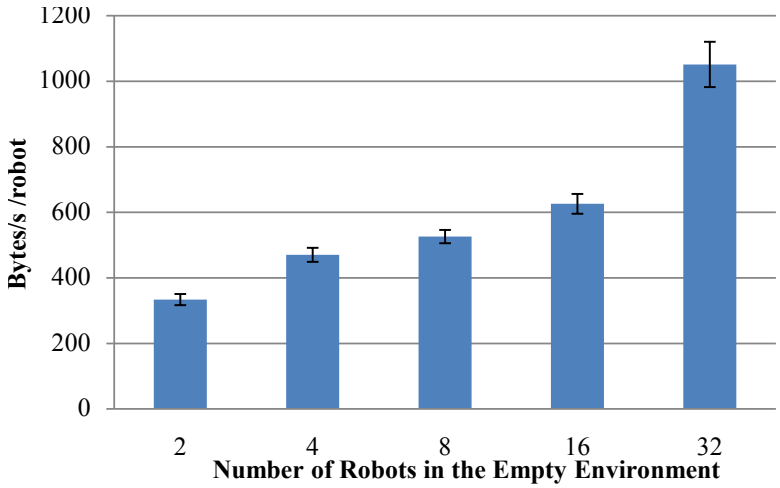


Figure 11: Average bandwidth required for robots to coordinate and find a solution to the decentralized motion planning problem with dynamics. 10 runs per data point.

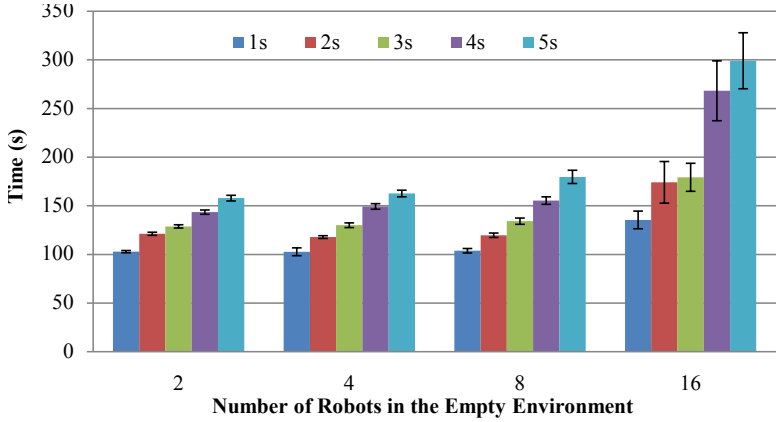


Figure 12: Planning cycle time has a significant effect on overall solution time in the empty environment. Each data point represents 10 runs.

5.6 Parameter Evaluation

This section investigates the effect that two important parameters have on the overall performance of the system. The first is the planning cycle of the robots, which determines the length of time that they have to come up with useful plans, as well as the size of the space×time obstacle their trajectories present to neighboring robots. The second is the communication range of each robot. This determines the number of neighboring robots that each robot communicates with, as well as its maximum velocity as computed in Section 4.4.

Planning Cycle An important parameter for the proposed approach is the duration of the planning cycle. The centralized simulation server was not able to handle cycle times of less than 1 second when more than 16 robots were simulated. Thus, such a duration is not shown here. This is strictly an implementation issue with the simulation and does not relate to the properties of the algorithm. It is expected that the limit in a hardware implementation would be dependent on the communication latency. Figure 12 shows that the average completion time noticeably increases as the duration of a cycle increases. The experiments presented in the previous figures were executed with differing cycle duration per robot, as described in Section 4.6. The faster solution time with shorter planning cycles is expected

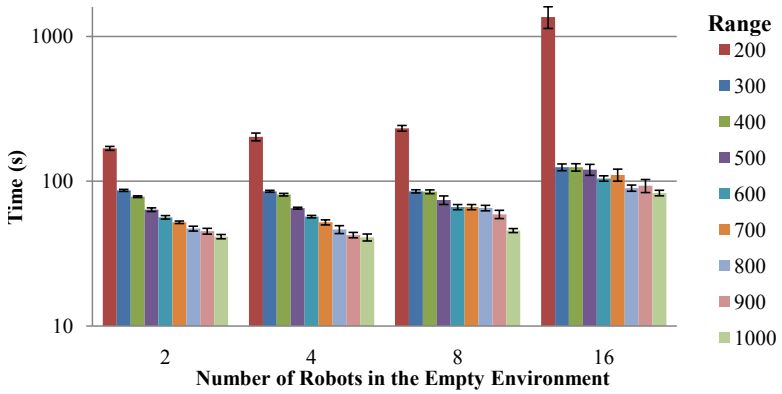


Figure 13: Increasing communication range (in meters) between robots significantly decreases overall solution times. Each data point represents 10 runs. The size of the environment is $1000m \times 1000m$.

behavior as the navigation problem in the empty environment is a simple one, and a lower planning cycle time increases v_{max} as well as decreases the space×time obstacles presented by the safety protocol. Both effects lend themselves to shorter solution times.

Communication Range An additional parameter considered in this work was the communication range. As shown in Figure 13, increasing the communication range allows for faster solution time. Again, a longer communication range allows for a larger v_{max}. Given the robot parameters presented, a communication range of 100m provided too low of a v_{max} to allow for a solution. For comparison, all environments measure $1000m \times 1000m$. A communication range of 200m caused the planner to approach v_{max} with almost every step and apply a negative acceleration in the very next step. This rapidly oscillating and low average velocity is responsible for the very slow completion time. All experiments other than this one were run with a communication range of 300m: the lowest tested range that provided a reasonable v_{max} for the entire range of planning cycles.

5.7 Different System Dynamics

To emphasize the generality of the proposed approach, an airplane-like robot model with a minimum velocity $v_{min} = 5$ was also used for experimentation. Otherwise, the system follows similar second-order car-like dynamics as those presented in Section 5.1. For this modified system, the braking maneuver is not possible as a contingency. Instead, the airplanes must enter a circular holding pattern. As long as the entire circular trajectory comprising the contingency is collision free, then a robot can continue in the holding pattern for all future times in the worst case. Thus, this is a valid contingency plan and does not violate the minimum velocity of the system.

A new contingency plan means that v_{max} needs to be computed with a different formula, based on a new d_{cont}:

$$d_{cont} = \frac{\pi}{\zeta_{max}} v_{min} + \frac{\zeta_{max}}{\phi_{max}} v_{max} + \frac{v_{max}^2 - {v_{min}}^2}{2\alpha}$$

This is a conservative approximation of d_{cont}. The first term $(\pi/\zeta_{max})v_{min}$ is an approximation of the forward distance traveled in a semicircle. Assuming $\zeta = \zeta_{max}$, it takes π/ζ_{max} time to traverse the semicircle. The distance traveled is then $(\pi/\zeta_{max})v_{min}$, assuming travel at minimum velocity. The second term $(\zeta_{max}/\phi_{max})\, v_{max}$ is the time that it takes to get to maximum turning, times the maximum forward speed. The third term is equivalent to the stopping distance, but is instead now a slowing distance to minimum velocity. These distances are not traveled in a straight line direction. Nevertheless, the above computation assumes so and provides a conservative approximation. Substituting this bound for d_{cont} in Equation 4 the following results are obtained:

$$v_{max}\delta + v_{max}\delta_{max} + \frac{\pi}{\zeta_{max}} v_{min} + \frac{\zeta_{max}}{\phi_{max}} v_{max} + \frac{v_{max}^2 - {v_{min}}^2}{2\alpha} = \frac{\mathbb{E} - \mathbb{S}}{2}.$$

Then it is possible to solve for v_{max}:

$$v_{max} = \frac{1}{\phi_{max}\zeta_{max}}\Big[-\alpha\zeta_{max}\left(\zeta_{max} + \phi_{max}\left(\delta_{max} + \delta\right)\right) + \sqrt{\zeta_{max}}\left(\phi_{max}^2\zeta_{max}v_{min}^2 + \alpha\phi_{max}^2\left(\mathbb{B}\zeta_{max} - \mathbb{S}\zeta_{max} - 2\pi v_{min}\right) + \alpha^2\zeta_{max}\left(\zeta_{max} + \phi_{max}\left(\delta_{max} + \delta\right)\right)^2\right)^{\frac{1}{2}}\Big].$$

Other than the change in v_{max}, the coordination algorithm can proceed as before without any changes. The planner does not require additional changes either, other than those required to respect the new v_{min}.

While the car-like robots are able to come to a complete stop within 16s, the airplanes need up to 70s to reach their circling holding mode. So the airplanes must reason about $\delta + 70$ seconds into the future. The collision checking time increases significantly with the length of a trajectory and decreases the responsiveness of robots to incoming messages. Each incoming trajectory must be completely collision checked before it is acknowledged, and the longer collision checking time means that many more acknowledgments are missed due to this additional latency.

Moreover, a simple analysis shows that the system, under the same parameters chosen for the car-like experiments, cannot be safe. In particular, for $\delta = 5$ the maximum velocity is $v_{max} = -1.5$, and $v_{max} = -2.1$ at $\delta = 1$, which clearly is unsatisfiable. Increasing the communication range $\mathbb{B}$ to 1000m, the maximum velocity gets to $v_{max} = 21.4$ for $\delta = 5$, or at $v_{max} = 27.8$ for $\delta = 1$. However, this communication range increase means that nearly all robots communicate with all other robots. This exacerbates the collision checking problem described above. In addition to all the computational difficulties incurred by the circular contingencies, start locations that are mutually safe become much harder to find in the case with a minimum velocity due to the comparatively large space×time obstacles that the contingencies cause. To avoid some of the difficulties that arise when differing cycles happen to line up very well and a robot is suddenly inundated with many requests in a short period of time, the cycle time was set to 2.5s for all robots, with varying offsets, as was done in prior work.

Despite the fact that the problem is quite more challenging when it involves systems with minimum positive velocity, it was still possible to run experiments and show safety. Using the parameters specified in the above paragraph for the plane robots, experiments in the office environment were executed successfully. In tests run with 2, 4, and 8 robots, which were repeated 10 times each, the system did not enter `ICS` at any time and was 100% safe. A graph for these experiments looks similar to that of Fig. 9, i.e., if the proposed scheme is employed, no collisions are observed, otherwise the systems are colliding. However, many of the experiments timed out and did not achieve their goals except for the simpler cases. The performance can potentially be improved by tuning planning parameters for the plane dynamics, and by a more sophisticated implementation of the path compatibility check (which is currently done by an all-against-all check of states along trajectories).

Systems with minimum velocity bounds, such as airplanes, raise an additional safety issue that does not relate to collision avoidance but to fuel. If such a system ends up in a holding pattern for a sufficient amount of time it might run out of fuel. It is possible to consider prioritized schemes that operate on top of the proposed safety framework, which give higher priority to systems with reduced resources, such as fuel, and assist them in making progress. In the general case, however, and for a decentralized approach, it is difficult to guarantee the solution of the motion coordination problem especially within prespecified time limits. This challenge relates to the liveness properties of a motion coordination protocol, which is an issue discussed in the last section of this paper.

6 Discussion

This paper presented a fully distributed algorithm for planning under second-order dynamics that does not make any synchronization assumptions about the operation of robots. The algorithm builds on top of a straightforward communication protocol, which, however, is not able to address issues arising due to inevitable collision states (`ICS`) with obstacles or among robots. This work shows that the proposed distributed algorithm guarantees `ICS` safety for second-order robots that move purposely in the same environment even if they are operating with dynamically changing planning cycles. The approach builds on top of conservative methods for guaranteeing safety that incorporate contingency trajectories (Fraichard and Asama, 2004; Petti and Fraichard, 2005; Bekris and Kavraki, 2007; Bekris et al., 2007, 2009; Grady et al., 2011). An analysis details the safety guarantees that the algorithm provides if the trajectories

are executed as planned and assuming that agents are in safe states once they start communicating. The paper discusses how to guarantee that robots are in safe states when they enter their neighbors' communication range by limiting their maximum velocity. The computation of the maximum velocity requires only local information and adapts to dynamic cycles. A possible way for optimizing the selection of trajectories is presented. This procedure demonstrates that once safety is guaranteed it is possible to consider robot interactions and exchange of additional information so as to improve efficiency. Simulations confirm that the framework indeed provides safety and is scalable and adaptable. In particular, the experiments suggest that the methodology can work with different types of contingency maneuvers.

Liveness issues While this paper focused on safety, an important question relates to liveness and completeness issues. The experimental results suggest that the approach can solve the problems tested for reasonable initial conditions. Nevertheless, the current framework does not guarantee a solution, as either a deadlock or a livelock may arise.

There are multiple aspects of the approach that affect its completeness. First, given that a replanning framework is used, a partial trajectory may not be sufficient to provide a solution. Recent work, however, has shown that for a single robot it is possible to provide probabilistic completeness guarantees within a replanning framework as long as the duration of a planning cycle can be adapted (Hauser, 2010). This was the motivating point for this work to show that safety can be guaranteed when robots adapt their planning cycles on the fly. Thus, a step has already been taken towards the direction of providing the capability to robots to extend their planning horizon when progress is not being made.

Another aspect of the approach that affects completeness is its decentralized nature. While highly desirable in real-world applications and necessary for scalability, the current decentralized solution does not reason in the composite state space of all the robots, which is necessary for completeness. One direction for resolving this issue is to consider a hybrid solution that integrates both decentralized and centralized planning, as in the work on dynamic robot networks (Clark et al., 2003), while also allowing for adaptive cycles (Hauser, 2010). Such a scheme would allow robots to operate in a decentralized manner when they can make progress. When progress is not being made, robots can increase their planning cycles and incorporate a longer planning horizon. If progress is still not being made, robots within a connected network component (i.e., a dynamic robot network) can switch to centralized planning until a solution that makes progress is found. Once the conflict is resolved, the robots can return to their decentralized alternative. Safety must still be guaranteed by the centralized planner if only a partial solution is returned. The contributions of the current paper are still relevant for this hybrid scheme as previous work on dynamic robot networks considered only first-order systems and did not deal with safety challenges. Overall, it is believed that the current method can be adapted to provide probabilistic completeness guarantees. The scalability of the described hybrid scheme that integrates a centralized planner into the current decentralized approach needs to be investigated.

Communication Cost and Alternative Methodologies This work takes a different approach compared to alternatives in the literature, such as control-based (Pallottino et al., 2007) and reactive approaches (Van den Berg et al., 2011), by incorporating communication between the robots. Alternatives often aim to solve motion coordination problems without any communication. On one hand, minimizing communication is indeed an important and desirable objective and provides increased decentralization and scalability. On the other hand, the current work provides the most general framework for guaranteeing safety for different system dynamics and for environments with obstacles. It is important to investigate ways to minimize the amount of communication for the proposed framework. One possible direction is to study the integration of the proposed planning framework with reactive and control-based alternatives. Towards this objective, it is interesting to consider the following: When are reactive methods sufficient for motion coordination and when should planning be considered for such problems? When is it necessary to exchange information between the robots regarding their intent and when information about their current state is sufficient?

Studying the Effects of Unreliable Communication The current framework has been devised so as to be able to deal in principle with latency and dropped messages and has been tested using message-passing communication. Even if acknowledgments are dropped the algorithm still provides safety: the lack of an acknowledgment will force the algorithm to select a contingency plan, which is guaranteed to be safe, as long as two agents have interacted once. Thus, the important requirement is that when two agents enter their respective communication ranges, they must be able to exchange messages.

Nevertheless, it is important to investigate the robustness of the approach given the limitations of wireless communication, specifically considering the bandwidth, latency, throughput and interference typical for wireless communication. Given the properties of wireless communication, latency is expected to be a more significant problem than bandwidth. In particular, the current experimental setup has considered reliable communication quality within a certain range that is about twice the breaking distance. In real environments this level of performance is rare and dropped communication will affect performance. This will result in an increased selection of contingency plans.

Beyond evaluating the effects of packet loss on the algorithm's performance, it is also interesting to investigate the relevance of "any-com" strategies (Otte and Correll, 2010). Such strategies aim to find an initial solution and then refine it as communication permits. In the "any-com" framework, a reliable communication environment allows better solutions to be found more quickly, but a more challenging one does not prohibit a solution from being found.

Other Directions for Future Work Beyond the above important issues, there are many interesting directions to consider for this line of research, such as the following:

Improving solution quality: The focus of this work has not been on the actual planning process for generating candidate trajectories or the pairwise optimization. It is possible to consider multiple alternatives for trajectory generation or pairwise trajectory optimization that will allow higher quality solutions within the proposed framework. Moreover, many aspects of the current approach can become less conservative. For instance, if there is some knowledge of when states in the trajectories are going to occur in time, the pairwise compatibility check (i.e., lines 10-13 and 17-20 in Algorithm 4.1) can become more efficient. In particular, if it is known that robot A^j will be at a specific state $x^j(t)$ at some point in time t during a window $t \in \Delta t$, then only states of other robots that occur during Δt need to be checked for compatibility with $x^j(t)$ (i.e., whether $x^i(t) \asymp x^i(t)$). Such information will reduce the state×time obstacle imposed by A^j to A^i.

Dealing with uncertainty: The current framework assumes that trajectories are executed perfectly. The replanning nature of the approach, however, does allow for adaptability in the case of errors in trajectory execution as the duration of replanning cycles decreases. It is interesting to investigate the robustness of the framework under uncertainty as well as an extension that considers probabilistic versions of safety and operates over distributions in the state space of robots.

Real-world experiments: This direction is related to the previous two. Given the focus of the approach on second-order systems, it is important to experiment with physical systems with interesting dynamics, such as `UAV`s. Such experiments can verify that `ICS` between multiple vehicles occur often in practice and that the proposed framework reduces the occurrence of collisions.

Addressing a variety of tasks: This work considers robots moving between independent initial and final states. Many other tasks, however, may require a different set of objectives and a level of cooperation between the robots while still requiring safety guarantees. For instance, a team of robots may need to move in an environment so as to explore it, or track an unknown target, or provide coverage. Many such applications may also require from the robots to maintain a communication network and an active link with a base station where information is forwarded to human operators.

Dealing with additional constraints: In relation to the previous direction, the safety framework may need to address additional constraints. For instance, if a communication network needs to be maintained, the robots have to treat the loss of connectivity similar to a collision. This is the approach followed in earlier work given the assumption that robots are synchronized (Bekris et al., 2009). Additionally, the robots may have to consider constraints, such as limited fuel. In this case, robots with lower fuel need to be given priorities within the safety framework so that they can proceed faster to their destination. Such constraints are directly related to the issue of liveness. It is in general difficult to provide guarantees for such problem instances.

Acknowledgements

Work by D. K. Grady, M. Moll and L. E. Kavraki on this paper has been supported in part by the US Army Research Laboratory and the US Army Research Office under grant number W911NF-09-1-0383 and by NSF IIS 0713623, NSF DUE 0920721, and NSF CCF 1018798. D. K. Grady has also been supported by an NSF Graduate Research Fellowship. Work by K. E. Bekris has been supported by NSF CNS 0932423. Experiments were run on equipment obtained by NSF EIA-0216467 and a partnership between Rice University, Sun Microsystems and Sigma Solutions Inc. Any conclusions expressed here are of the authors and do not reflect the views of the sponsors.

References

Alami, R., Simeon, T., and Krishna, K. M. (2002). On the influence of sensor capacities and environment dynamics onto collision-free motion plans. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 2395–2400.

Azarm, K. and Schmidt, G. (1997). Conflict-free Motion of Multiple Mobile Robots based on Decentralized Motion Planning and Negotiation. In *IEEE Intern. Conference on Robotics and Automation*, pages 3526–3533.

Bekris, K. E. and Kavraki, L. E. (2007). Greedy but safe replanning under kinodynamic constraints. In *IEEE Intl. Conf. on Robotics and Automation*, pages 704–710.

Bekris, K. E., Tsianos, K., and Kavraki, L. E. (2009). Safe and distributed kinodynamic replanning for vehicular networks. *Mobile Networks and Applications*, 14(3):292–308.

Bekris, K. E., Tsianos, K. I., and Kavraki, L. E. (2007). A decentralized planner that guarantees the safety of communicating vehicles with complex dynamics that replan online. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3784–3790.

Bennewitz, M., Burgard, W., and Thrun, S. (2002). Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots. *Robotics and Autonomous Systems*, 41(2):89–99.

Brock, O. and Khatib, O. (1999). High-speed navigation using the global dynamic window approach. In *IEEE Intl. Conf. on Robotics and Automation*, pages 341–346, Detroit, MI, USA.

Bruce, J. R. and Veloso, M. M. (2006). Safe multirobot navigation within dynamics constraints. *Proc. IEEE*, 94(7):1398–1411.

Chan, N., Kuffner, J. J., and Zucker, M. (2008). Improved motion planning speed and safety using regions of inevitable collision. In *17th Symposium on Robot Design, Dynamics, and Control (RoManSy'08)*, pages 103–114.

Clark, C., Rock, S., and Latombe, J.-C. (2003). Motion planning for multi-robot systems using dynamic robot networks. In *IEEE Intl. Conf. on Robotics and Automation*, pages 4222–4227, Taipei, Taiwan.

Dimarogonas, D. V. and Kyriakopoulos, K. J. (2007). Decentralized navigation functions for multiple robotic agents with limited sensing capabiliites. *Journal of Intelligent and Robotic Systems*, 48(3):411–433.

Du Toit, N. E. and Burdick, J. W. (2010). Robotic motion planning in dynamic, cluttered, uncertain environments. In *IEEE Intl. Conf. on Robotics and Automation*, pages 966–973.

Earl, M. and D'Andrea, R. D. (2005). Iterative MILP Methods for Vehicle Control Problems. *IEEE Trans. Robotics*, 21:1158–1167.

Erdmann, M. A. and Lozano-Pérez, T. (1987). On multiple moving objects. *Algorithmica*, 2(1):477–521.

Ferguson, D., Kalra, N., and Stentz, A. (2006). Replanning with rrts. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1243–1248, Orlando, FL.

Fiorini, P. and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *Intl. J. Robotics Research*, 17(7):760–772.

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33.

Fraichard, T. (2007). A short paper about motion safety. In *IEEE Intl. Conf. on Robotics and Automation*, pages 1140–1145, Rome, Italy.

Fraichard, T. and Asama, H. (2004). Inevitable collision states: A step towards safer robots? *Advanced Robotics*, 18(10):1001–1024.

Frazzoli, E., Dahleh, M., and Feron, E. (2002). Real-time motion planning for agile autonomous vehicles. *AIAA Journal of Guidance, Control and Dynamics*, 25(1):116–129.

Ghosh, R. and Tomlin, C. J. (2000). Maneuver design for multiple aircraft conflict resolution. In *Proc. American Control Conference*, Chicago, IL.

Grady, D. K., Bekris, K. E., and Kavraki, L. E. (2011). Asynchronous distributed motion planning with safety guarantees under second-order dynamics. In *Algorithmic Foundations of Robotics IX*, volume 68, pages 53–70.

Hauser, K. (2010). Adaptive time stepping in real-time motion planning. In *Workshop on the Algorithmic Foundations of Robotics*.

Hsu, D., Kindel, R., Latombe, J.-C., and Rock, S. (2002). Randomized kinodynamic motion planning with moving obstacles. *Intl. J. Robotics Research*, 21(3):233–255.

Kalisiak, M. and Van de Panne, M. (2007). Faster motion planning using learned local viability models. In *IEEE Intl. Conf. on Robotics and Automation*, pages 2700–2705, Rome, Italy.

Kant, K. and Zucker, S. (1986). Towards efficient trajectory planning: The path-velocity decomposition. *Intl. J. Robotics Research*, 5(3):72–89.

Lalish, E. and Morgansen, K. (2008). Decentralized reactive collision avoidance for multivehicle systems. In *IEEE Conf. on Decision and Control*, pages 1218–1224.

Lamiraux, F., Bonnafous, D., and Lefebvre, O. (2004). Reactive path deformation for nonholonomic mobile robots. *IEEE Trans. Robotics*, 20(6):967–977.

Large, F., Laugier, C., and Shiller, Z. (2005). Navigation among moving obstacles using the NLVO: Principles and applications to intelligent vehicles. *Autonomous Robots*, 19(2):159–171.

Laumond, J.-P., editor (1998). *Robot Motion Planning and Control*. Lectures Notes in Control and Information Sciences 229. Springer.

LaValle, S. M. and Hutchinson, S. (1998). Optimal motion planning for multiple robots having independent goals. *IEEE Trans. Robotics and Automation*, 14(6):912–925.

LaValle, S. M. and Kuffner, J. J. (2001). Randomized kinodynamic planning. *Intl. J. Robotics Research*, 20(5):378–400.

Li, W. and Cassandras, C. (2006). A cooperative receding horizon controller for multi-vehicle uncertain environments. *IEEE Trans. Automatic Control*, 51(2):242–257.

Li, Y., Gupta, K., and Payandeh, S. (2005). Motion planning of multiple agents in virtual environments using coordination graphs. In *IEEE Intl. Conf. on Robotics and Automation*, pages 378–383.

Likhachev, M. and Ferguson, D. (2009). Planning long dynamically feasible maneuvers for autonomous vehicles. *Intl. J. Robotics Research*, 28(8):933–945.

Loizu, S., Dimarogonas, D., and Kyriakopoulos, K. (2004). Decentralized feedback stabilization of multiple nonholonomic agents. In *IEEE Intl. Conf. on Robotics and Automation*, volume 3, pages 3012–3017.

Lumelsky, V. J. and Harinarayan, K. R. (1997). Decentralized motion planning for multiple mobile robots: The cocktail party model. *Autonomous Robots*, 4(1):121–135.

Martinez-Gomez, L. and Fraichard, T. (2009). Collision avoidance in dynamic environments: An ICS-based solution and its comparative evaluation. In *IEEE Intl. Conf. on Robotics and Automation*, pages 100–105, Kobe, Japan.

O'Donnell, P. and Lozano-Perez, T. (1989). Deadlock-free and collision-free coordination of two robot manipulators. In *IEEE Intl. Conf. on Robotics and Automation*, pages 484–489.

Otte, M. and Correll, N. (2010). Any-com multi-robot path-planning: Multi-robot coordination under communication constraints and dynamic team sizes. In *International Symposium on Experimental Robotics (ISRR)*, New Delhi, India.

Pallotino, L., Scordio, V. G., and Bicchi, A. (2004). Decentralized cooperative policy for conflict resolution among multiple autonomous mobile agents. In *IEEE Conf. on Decision and Control*, volume 5, pages 4758–4763.

Pallottino, L., Scordio, V., Bicchi, A., and Frazzoli, E. (2007). Decentralized cooperative policy for conflict resolution in multivehicle systems. *IEEE Trans. Robotics*, 23(6):1170–1183.

Peng, J. and Akella, S. (2005). Coordinating multiple robots with kinodynamic constraints along specified paths. *Intl. J. Robotics Research*, 24(4):295–310.

Petti, S. and Fraichard, T. (2005). Safe motion planning in dynamic environments. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Edmonton, AB, Canada.

Philippsen, R., Kolski, S., and Maček, K. (2008). Mobile robot planning in dynamic environments and on growable costmaps. In *Workshop on Planning with Cost Maps at the IEEE Intl. Conf. on Robotics and Automation*.

Pivtoraiko, M., Knepper, R. A., and Kelly, A. (2009). Differentially constrained mobile robot motion planning in state lattices. *J. Field Robotics*, 26(3):308–333.

Reif, J. and Sharir, M. (1985). Motion planning in the presence of moving obstacles. In *Proc. IEEE Intl. Symp. on Foundations of Computer Science*, Portland, OR.

Rimon, E. and Koditschek, D. E. (1992). Exact robot navigation using artificial potential functions. *IEEE Trans. Robotics and Automation*, 8(5):501–508.

Roussos, G., Dimarogonas, D. V., and Kyriakopoulos, K. J. (2010). 3D navigation and collision avoidance for non-holonomic aircraft-like vehicles. *Intl. J. Adaptive Control and Signal Processing*, 24(10):900–920.

Snape, J., van den Berg, J., Guy, S. J., and Manocha, D. (2011). The hybrid reciprocal velocity obstacle. *IEEE Transactions on Robotics*.

Tomlin, C. J., Mitchell, I., and Ghosh, R. (2001). Safety verification of conflict resolution maneuvers. *IEEE Trans. Intelligent Transportation Systems*, 2(2).

Tomlin, C. J., Pappas, G. J., and Sastry, S. (1998). Conflict resolution for air traffic management: A case study in multi-agent hybrid systems. *IEEE Trans. Automatic Control*, 43:509–521.

Van den Berg, J., Guy, S. J., Lin, M., and Manocha, D. (2009). Reciprocal *n*-body collision avoidance. In *International Symposium on Robotics Research (ISRR)*.

Van den Berg, J., Lin, M., and Manocha, D. (2008). Reciprocal velocity obstacles for real-time multi-agent navigation. In *IEEE Intl. Conf. on Robotics and Automation*, pages 1928–1935.

Van den Berg, J. and Overmars, M. (2005). Prioritized motion planning for multiple robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2217–2222.

Van den Berg, J., Snape, J., Guy, S. J., and Manocha, D. (2011). Reciprocal collision avoidance with acceleration-velocity obstacles. In *IEEE Intl. Conf. on Robotics and Automation*.

Vatcha, R. and Xiao, J. (2008). Perceived CT-Space for motion planning in unknown and unpredictable environments. In *Workshop on Algorithmic Foundations of Robotics*, Mexico.

Wikman, M. S., Branicky, M., and Newman, W. S. (1993). Reflexive collision avoidance: A generalized approach. In *IEEE Intl. Conf. on Robotics and Automation*, pages 31–36.

Yang, Y. and Brock, O. (2010). Elastic roadmaps—motion generation for autonomous mobile manipulation. *Autonomous Robots*, 28:113–130.