Asynchronous Distributed Motion Planning with Safety Guarantees under Second-Order Dynamics

Devin K. Grady, Kostas E. Bekris and Lydia E. Kavraki

Abstract As robots become more versatile, they are increasingly found to operate together in the same environment where they must coordinate their motion in a distributed manner. Such operation does not present problems if the motion is quasi-static and collisions can be easily avoided. However, when the robots follow second-order dynamics, the problem becomes challenging even for a known environment. The setup in this work considers that each robot replans its own trajectory for the next replanning cycle. The planning process must guarantee the robot's safety by ensuring collision-free paths for the considered period and by not bringing the robot to states where collisions cannot be avoided in the future. This problem can be addressed through communication among the robots, but it becomes complicated when the replanning cycles of the different robots are not synchronized and the robots make planning decisions at different time instants. This paper shows how to guarantee the safe operation of multiple communicating second-order vehicles, whose replanning cycles do not coincide, through an asynchronous, distributed motion planning framework. The method is evaluated through simulations, where each robot is simulated on a different processor and communicates with its neighbors through message passing. The simulations confirm that the approach provides safety in scenarios with up to 48 robots with second-order dynamics in environments with obstacles, where collisions occur often without a safety framework.

1 Introduction

This paper considers multiple autonomous robots with non-trivial dynamics operating in a static environment. The robots try to reach their individual goals without col-

Devin K. Grady, Lydia E. Kavraki
Computer Science, Rice Univ., Houston, TX, e-mail: {devin.grady,kavraki}@rice.edu

Kostas E. Bekris
Computer Science and Engineering, Univ. of Nevada Reno, Reno, NV, e-mail: bekris@cse.unr.edu

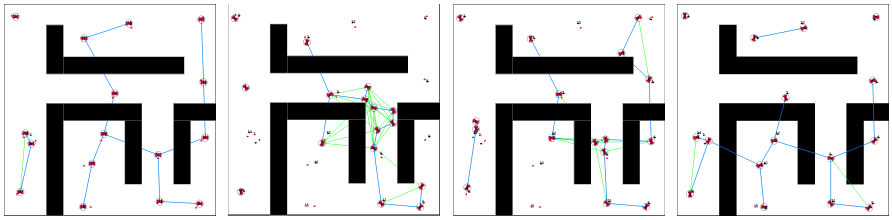


Fig. 1 A sample run in the office environment (left to right). Links show communicating robots.

lisions. Such scenarios are becoming increasingly interesting. For instance, consider the case of vehicles moving in a parking lot or going through a busy intersection, or unmanned aerial vehicles that carry out complex maneuvers. These examples involve second-order systems, which cannot stop instantaneously and must respect limits in the second-order derivatives of their state parameters. For such systems, collisions with other robots or obstacles cannot be easily avoided.

Real applications also require the solution of such problems in a decentralized manner. This work imposes a requirement for a decentralized solution and considers robots that replan their trajectories on the fly. Replanning allows robots to consider multiple alternative trajectories during each cycle and provides flexibility in changing environments. To coordinate the robots, this work utilizes communication. A planning algorithm makes use of information collected through communication to avoid collisions for the next cycle and ensure that robots reach states from where collisions can be avoided in the future. The duration of the cycle is the same for all robots, but the robots are not synchronized. Hence communication of plans can happen at any point and 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 setup.

Background Safety issues for dynamical systems were first studied many years ago. Collision-free states that inevitably lead to collisions have been referred as Obstacle Shadows [24], Regions of Inevitable Collision [20] or Inevitable Collision States (ICS) [13]. A study on ICS resulted in conservative approximations [13] and generic ICS checkers [21]. It also provided *3 criteria for motion safety*: a robot must (i) consider its dynamics, (ii) the environment's future behavior, and (iii) reason over infinite-time horizon [12]. This line of research, however, did not deal with coordinating robots as the current paper does.

Reactive methods, such as the Dynamic Window Approach [11] and Velocity Obstacles [10], can enable a robot to avoid collisions for unknown on dynamic environments. Many existing reactive planners, however, do not satisfy the criteria for motion safety [12, 21]. Path deformation techniques compute a flexible path, adapted on the fly to avoid moving obstacles [18, 27], but do not deal with ICS. Reciprocal Velocity Obstacles (RVOs) [4] involve multiple agents which simultaneously avoid one another without communication but do not deal yet with ICS. A related control-based approach [17] deals with second-order models of a planar unicycle but does not provide guarantees in the presence of obstacles.

In contrast to reactive approaches, this paper focuses on *planning* safe paths. Planning has a longer horizon so it does not get stuck in minima as easily and extends to high degrees-of-freedom systems. Reasoning about safety during planning focuses the search on the safe part of the state space. In this work planning is achieved using a sampling-based tree planner [20, 15, 2]. Alternatives include, among others, navigation functions [8] and lattice-based approaches [23].

Braking maneuvers have been shown sufficient in providing safety in static environments [26] and have been combined with sampling-based replanning [5, 2]. For dynamic environments, relaxations of ICS are typically considered, such as τ-safety [14]. This notion guarantees no collision for τ seconds in the future for each node of a sampling-based tree. A sampling-based planner was tested on air-cushioned robots moving in dynamic environments, where an escape maneuver was computed when the planner failed to find a solution [15]. Learning-based approximations of ICS can also be found [16], as well as approximations of state×time space obstacles [6]. Other works focus on the interaction between planning and sensing, and point out that planning must be limited within the robot's visibility region [1, 25]. The current paper extends the authors' earlier work [3], which integrated a sampling-based planner with ICS avoidance [2] to safely plan for multiple robots that formed a network and explored an unknown workspace. The previous work required a synchronous planning operation, which simplified coordination.

Planning for dynamic networks of robots has been approached by a combination of centralized and decoupled planning [7], without considering, however, the ICS challenge. Centralized planning does not scale and decoupled approaches, which may involve prioritization [9] or velocity tuning [22], are incomplete. The existing work follows a decoupled approach for performance purposes. In contrast to velocity tuning, it weakly constraints the robots' motion before considering interactions since it allows multiple alternative paths for each robot at each cycle. At the same time, it does not impose priorities but instead robots respect their neighbors in a way that emerges naturally from their asynchronous operation.

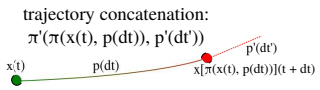
Contributions This work extends the range of problems that can be solved efficiently with guarantees for ICS avoidance. The paper presents a general framework for independent but communicating second-order robots to reach their destinations in an otherwise known environment. The framework is fully distributed and relies on asynchronous interaction among the robots, where the robots' replanning cycles are not synchronized, the robots have no knowledge about their clock differences and no access to a global clock. It 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, this work emphasizes the importance of communicating contingencies in multi-robot scenarios and studies the asynchronous case. A proof 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 48 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.

2 Problem Statement

Consider robots operating in the same known workspace with static obstacles. Each **robot** R^i exhibits drift and must satisfy non-holonomic constraints expressed by differential equations of the form: $\dot{x}^i = f^i(x^i, u^i)$, $g^i(x^i, \dot{x}^i) \leq 0$, where $x^i \in \mathcal{X}^i$ represents a **state**, u^i is a **control** and f^i, g^i are smooth. The subset of the **state space** $\mathcal{X}^i$ that does not cause a collision with static obstacles is denoted as $\mathcal{X}^i_f$. The robot model used in this paper can be found in Section 5 and involves acceleration controlled car-like systems, including versions with minimum positive velocity.

Each R^i is located at an initial state $x^i(0)$ and must compute plans that will bring it to its individual goal $x^i_g(t_{max})$ without collisions and within finite time t_{max}. Then:

- A **plan** is a sequence of controls $p(dt) = \{(u_1, dt_1), \ldots, (u_n, dt_n)\}$ $(dt = \sum_i dt_i)$.
- A plan $p(dt)$ executed at state $x(t)$ defines a **trajectory**: $\pi(x(t), p(dt))$, which is a sequence of states.
- A trajectory is **feasible** as long as it satisfies functions f^i and g^i for robot R^i.
- A plan $p(dt)$ is **valid** at state $x(t)$, if it defines a feasible trajectory $\pi(x(t), p(dt))$.
- A state along $\pi(x(t), p(dt))$ at time $t' \in [t : t + dt]$ is denoted as $x[\pi(x(t), p(dt))](t')$.
- A feasible trajectory $\pi(x(t), p(dt))$ is **collision-free** with respect to the static obstacles if: $\forall\ t' \in [t : t + dt] : \quad x[\pi(x(t), p(dt))](t') \ \in \ \mathcal{X}_f$.
- For a **trajectory concatenation** (figure below) $\pi'(\pi(x(t), p(dt)), p'(dt'))$, plan $p(dt)$ is executed at $x(t)$ and then $p'(dt')$ is executed at state: $x[\pi(x(t), p(dt))](t + dt)$.
- Two trajectories for R^i and R^j are **compatible:** $\pi^i(x^i(t^i), p(dt^i)) \asymp \pi^j(x^j(t^j), p^j(dt^j))$ as long as:
$$x[\pi^i](t) \asymp x[\pi^j](t) \quad \forall\ t \in [max(t^i, t^j) : min(t^i + dt^i, t^j + dt^j)]$$
where $x^i \asymp x^j$ means that R^i in state x^i does not collide with R^j at state x^j. The corresponding plans $p(dt^i)$, $p(dt^j)$ are also called compatible at states $x^i(t^i)$, $x^j(t^j)$.



The robots are equipped with an omnidirectional, range-limited communication tool, which is reliable and used for coordination and collision avoidance. The robots within range of R^i define the neighborhood N^i. A robot has information about other robots only if they communicate.

Given the above notation, the problem of **distributed motion planning with dynamics (**`DMPD`**)** can be defined as follows: Consider m robots with range-limited communication capabilities operating in the same workspace with obstacles. Each robot's motion is governed by second-order dynamics specified by f^i and g^i. Initially, robot R^i is located at state $x^i(0)$, where $x^i(0) \in \mathcal{X}^i_f$ and $\forall i, j : \ x^i(0) \asymp x^j(0)$. Each R^i must compute a valid plan $p^i(t_{max})$ so that:

- $x[\pi^i(x^i(0), p^i(t_{max}))](t_{max}) = x^i_g(t_{max})$ (i.e., the plans bring the robots to their individual goals within time t_{max}),
- $\forall\ i, \quad \forall t \ \in [0 : t_{max}] : \quad x[\pi^i(x^i(0), p^i(t_{max}))](t) \in \mathcal{X}_f$ (i.e., the resulting trajectories are collision-free with static obstacles)
- and $\forall\ i, j : \quad \pi^i(x^i(0), p^i(t_{max})) \asymp \pi^j(x^j(0), p^j(t_{max}))$ (i.e., the trajectories are pairwise compatible from the beginning and until all the robots reach their goals).

3 A Simple Framework without Safety Guarantees

This paper adopts a decentralized framework for scalability purposes. Each robot's operation is broken into intervals $([t_0^i : t_1^i], [t_1^i, t_2^i], \ldots, [t_n^i : t_{n+1}^i], \ldots)$, called cycles. During $[t_{n-1}^i : t_n^i]$, robot R^i considers different plans Π^i for cycle $[t_n^i : t_{n+1}^i]$, given the future initial state $x^i(t_n^i)$. Through coordination, R^i selects plan $p_*^i([t_n^i : t_{n+1}^i])$.

It is assumed that the duration of each cycle is constant and the same for all robots: $\forall i,\ \forall n : t_{n+1}^i - t_n^i = dt$. Nevertheless, the robots do not have a synchronous operation: the cycles among different robots do not coincide and t_0^i is typically different than t_0^j. Synchronicity is a restrictive assumption, as it requires all the robots to initiate their operation at exactly the same time although they may be located in different parts of the world and may not communicate their initial states.

Given this setup, Algorithm 3.1 outlines a straightforward approach for the single cycle operation of each robot that tries to find compatible plans. During $[t_{n-1}^i : t_n^i]$, R^i computes alternative partial plans Π^i for the consecutive planning cycle. In parallel, R^i listens for messages from robots in neighborhood N^i. The messages contain the selected trajectories for each robot. When time approaches $t_n^i - \epsilon$, R^i selects among all trajectories that are collision-free and compatible with the neighbors' messages, the one that brings the robot closer to its goal. If such a trajectory is indeed found at each iteration, then the DMPD problem is eventually solved by this algorithm.

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

$\Pi^i \leftarrow \emptyset$ and $\Pi^{N^i} \leftarrow \emptyset$
while $t < t_n^i - \epsilon$ **do**
  $\pi^i(x^i(t_n^i), p^i(t_n^i : t_{n+1}^i)) \leftarrow$ collision-free trajectory from a single-robot planner
  $\Pi^i \leftarrow \Pi^i \cup \pi^i(x^i(t_n^i), p^i(t_n^i : t_{n+1}^i))$
  if $R^j \in N^i$ is transmitting a trajectory π^j **then** $\Pi^{N^i} \leftarrow \Pi^{N^i} \cup \pi^j$
for all $\pi^i \in \Pi^i$ **do**
  for all $\pi^j \in \Pi^{N^i}$ **do**
    if $\pi^i \not\sim \pi^j$ (incompatible trajectories) **then** $\Pi^i \leftarrow \Pi^i - \pi^i$
$\pi_*^i \leftarrow$ trajectory in Π^i which brings R^i closer to the goal given a metric
Transmit π_*^i to all neighbors in N^i and execute π_*^i during next cycle

4 A Safe Solution to Distributed Motion Planning with Dynamics

A robot following the above approach might fail to find a trajectory π_*^i. This section describes a distributed algorithm that guarantees the existence of a collision-free, compatible trajectory for all robots at every cycle.

A. Safety Considerations - Inevitable Collision States: One reason for failure is when the single-robot planner fails to find collision-free paths. This is guaranteed to happen when $x^i(t_n^i)$ is an ICS. State $x(t)$ is ICS with regards to static obstacles if: $\forall\ p(\infty) :\ \exists\ dt \in [t, \infty)$ so that $x[\pi(x(t), p(\infty))] \notin \mathcal{X}_f$.

Computing whether a state is ICS is intractable, since it requires reasoning over an infinite horizon for all possible plans. It is sufficient, however, to consider conservative methods that identify states that are *not* ICS [13, 2]. The approximation reasons over a subset of predefined maneuvers $\Gamma(\infty)$, called here **contingency plans**. If R^i can avoid collisions in the future with static obstacles at $x^i(t_n)$ by guaranteeing that a contingency plan $\gamma^i(\infty) \in \Gamma^i(\infty)$ avoids collisions over an infinite horizon, then $x^i(t_n)$ is not ICS with regards to static obstacles. For cars, braking maneuvers are sufficient since it is possible to reason over an infinite time horizon whether these plans will collide with static obstacles. Circling maneuvers can be used for systems with minimum velocity limits, such as airplanes.

Multiple moving robots pose new challenges for ICS. Trajectories π^i and π^j may be compatible for the next cycle, but the corresponding robots may reach states that will inevitably lead them in a future collision. Thus, safety notions have to be extended into the multi-robot case. It is still necessary for computational reasons to be conservative and focus only on a set of contingency plans. For m robots $\{R^1, R^2, \ldots, R^m\}$ executing plans $\{p^1(dt^1), p^2(dt^2), \ldots, p^m(dt^m)\}$ at states $\{x^1(t), x^2(t), \ldots, x^m(t)\}$, state $x^i(t)$ is considered a **safe state** if:

$$\exists\, \gamma^i(\infty) \in \Gamma^i(\infty) \text{ so that } \forall\, t' \in [t,\infty): \quad x[\pi^i(x^i(t), \gamma^i(\infty))](t') \in \mathcal{X}_f \text{ and}$$
$$\forall\, j \in [1,m],\ j \neq i,\ \exists\, \gamma^j(\infty) \in \Gamma^j(\infty): \quad \pi^i(x^i(t), \gamma^i(\infty)) \asymp \pi^j(\,\pi^j(x^j(t), p^j(dt^j)),\ \gamma^j(\infty)).$$

In the above definition, dt^j is the remaining of robot R^j's cycle past time t. Note that a trajectory concatenation is used for R^j's trajectory. In this trajectory concatenation, $p^j(dt^j)$ is executed for time dt^j and then the contingency $\gamma^j(\infty)$ is applied. The reason is that as robots decide asynchronously, it may happen that at t, robot R^j has already committed to plan $p^j(dt^j)$. Extending the assumption in the problem statement about compatible starting states, the following discussion will assume that the initial states of all the robots are safe states. Then an algorithm for the DMPD problem must maintain the following invariant for each robot and planning cycle:

Safety Invariant: The selected trajectory $\pi^i_*(x^i(t^i_n), p^i(t^i_n : t^i_{n+1}))$:

a) Must be collision-free with obstacles.

b) Must be compatible with all other robots, during the cycle $(t^i_n : t^i_{n+1})$:
$\pi^i_*(x^i(t^i_n), p^i(t^i_n : t^i_{n+1})) \asymp \pi^j_*(x^j(t^i_n), p^j(t^i_n : t^i_{n+1})),\ \ \forall\, j \neq i.$

c) The resulting state $x[\pi^i_*](t^i_{n+1})$ is safe for all possible future plans $p^j(t^i_{n+1} : \infty)$ selected by other robots ($j \neq i$). In other words, the concatenation of π^i_* with $\gamma^i(\infty)$ must be compatible with the concatenations of other vehicles, i.e., $\forall\, j \neq i$:
$\pi^i(\pi^i_*(x^i(t^i_n), p^i(t^i_n : t^i_{n+1})), \gamma^i(\infty)) \asymp\ \pi^j(\pi^j_*(x^j(t^i_n), p^j(t^i_n : t^i_{n+1})), \gamma^j(\infty)).$

Point c) above means that R^i has a contingency plan at $x[\pi^i_*](t^i_{n+1})$, which can be safely followed for the other robots' choices given the 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.

Algorithm 4.1 Safe and Asynchronous Operation of R^i During Cycle $[t^i_{n-1} : t^i_n]$

1: $\Pi^i \leftarrow \emptyset$, $\Pi^{N^i}_{prev} \leftarrow \emptyset$, $\Pi^{N^i}_{new} \leftarrow \emptyset$
2: **for all** $R^j \in N^i$ **do**
3: $\quad \Pi^{N^i}_{prev} \leftarrow \Pi^{N^i}_{prev} \cup \pi^j(\,\pi^j(x^j(t^j_{n-1}), p^j(t^j_{n-1} : t^j_n)),\ \gamma(t^j_n : \infty)\,)$
(i.e., include all past trajectories and attached contingencies of neighbors)
4: **while** $t < t^i_n - \epsilon$ **do**
5: $\quad \pi^i(x^i(t^i_n), p^i(t^i_n : t^i_{n+1})) \leftarrow$ collision-free trajectory from a single-robot planner
6: $\quad \pi^i_\gamma \leftarrow \pi^i(\,\pi^i(x^i(t^i_n), p^i(t^i_n : t^i_{n+1})),\ \gamma(t^i_{n+1} : \infty))$ (i.e., contingency concatenation)
7: $\quad$ **if** $\forall\ t \in [t^i_{n+1} : \infty) :\ x[\pi^i_\gamma](t) \in \mathcal{X}_f$ **then**
8: $\quad\quad \Pi^i \leftarrow \Pi^i \cup \pi^i_\gamma$
9: $\quad\quad$ **for all** $\pi^j_\gamma \in \Pi^{N^i}_{prev}$ **do**
10: $\quad\quad\quad$ **if** $\pi^i_\gamma \not\sim \pi^j_\gamma$ **then**
11: $\quad\quad\quad\quad \Pi^i \leftarrow \Pi^i - \pi^i_\gamma$
12: $\quad$ **if** $R^j \in N^i$ is transmitting a trajectory and an attached contingency **then**
13: $\quad\quad \Pi^{N^i}_{new} \leftarrow \Pi^{N^i}_{new} \cup \pi^j_\gamma(\,\pi^j(x^j(t^j_n), p^j(t^j_n : t^j_{n+1})),\ \gamma(t^j_{n+1} : \infty)\,)$
14: **for all** $\pi^i_\gamma \in \Pi^i$ **do**
15: $\quad$ **for all** $\pi^j_\gamma \in \Pi^{N^i}_{new}$ **do**
16: $\quad\quad$ **if** $\pi^i_\gamma \not\sim \pi^j_\gamma$ **then**
17: $\quad\quad\quad \Pi^i \leftarrow \Pi^i - \pi^i_\gamma$
18: **if** Π^i empty or if a message was received during compatibility check **then**
19: $\quad \pi^i_* \leftarrow \pi^i(x^i(t^i_n), \gamma(t^i_n : \infty))$ (i.e., follow the available contingency for next cycle)
20: **else**
21: $\quad \pi^i_* \leftarrow$ trajectory in Π^i which brings R^i closer to the goal given a metric
22: Transmit π^i_* to all neighbors in N^i and execute π^i_* during next cycle

B. Safe and Asynchronous Distributed Solution: Algorithm 4.1, in contrast to Algorithm 3.1, maintains the safety invariant. The protocol follows the same high-level framework and still allows a variety of planning techniques to be used for producing trajectories. The differences with the original algorithm can be summarized as follows:

- The algorithm stores the messages received from neighbors during the previous cycle in the set $\Pi^{N^i}_{prev}$ *(lines 1-3)*. Note that the robots transmit the selected trajectory together with the corresponding contingency *(lines 12-13 and 22)*.
- A contingency plan $\gamma(t^i_{n+1} : \infty)$ is attached to every collision-free trajectory $\pi^i(x^i(t^i_n), p^i(t^i_n : t^i_{n+1}))$ and the trajectory concatenation π^i_γ is generated *(line 5-6)*. Note that potentially multiple different contingencies can be attached to the trajectory $\pi^i(x^i(t^i_n), p^i(t^i_n : t^i_{n+1}))$. Each resulting trajectory concatenation is treated individually by the algorithm.
- The trajectory π^i_γ is added to Π^i only if it is collision-free with static obstacles for an infinite time horizon *(lines 7-8)*, thus guaranteeing that $x[\pi^i](t^i_{n+1})$ is not `ICS`.
- π^i_γ is rejected, however, if it is not compatible with all the trajectories and contingencies of neighbors from the compatibility check *(lines 14-17)*. R^i checks not just trajectories for the next cycle but its trajectory concatenations with contingencies π^i_γ against its neighbors' trajectory concatenations π^j_γ.

- The final change *(lines 18-21)* addresses the possibility that Π^i is empty or when a message arrives while R^i executes its compatibility check. If any of the two is true, then R^i selects to follow the contingency $\gamma(t_n^i : \infty)$, which was used in the previous cycle to prove that $x(t_n^i)$ was safe. Otherwise, R^i selects among the set Π^i the trajectory that brings it closer to the goal according to a desired metric. previous cycle, stored in $\Pi_{prev}^{N^i}$ *(lines 9-11)*.
- The while loop *(lines 4-13)* is executed as long as time t is less than the end of the planning cycle (t_n^i) minus an ϵ time period. Time ϵ should be sufficient for the robot to complete the compatibility check *(lines 14-17)* and the selection process *(lines 18-22)*. 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. In a serial implementation ϵ has to be sufficiently large.

Overall, each robot selects a plan $p^i(t_n^i : t_{n+1}^i)$ and a contingency $\gamma^i(t_{n+1}^i : \infty)$ that respect the plans and contingencies of other robots that have been selected before time t_n^i. If no such plan is found or there is no time to check against newly incoming messages, then the contingency $\gamma^i(t_n^i : \infty)$ is selected.

Computational Complexity: The algorithm's complexity depends on the number of neighbors N^i, which in the worst case is the total number of robots N. In order to evaluate the cost of operations involving trajectories, it is important to consider a trajectory representation. A discrete sequence of states can be sampled along a trajectory, given a predefined resolution in time Q (i.e., the technique becomes resolution-safe in this case). Then, let S be the upper limit in the number of states used to represent each trajectory conceternation. P denotes the upper limit in the number of plans considered during each planning cycle for the current agent.

Given the above notation, the complexity of the algorithm's various operations is as follows: (a) *Lines 2-3 :* $S \times N$, (b) *Lines 7 - 8:* $P \times S$, (c) *Lines 9 -11:* $P \times N \times S^2$ (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 14-17:* Same as above, (e) *Lines 20-21:* P, assuming constant time for computing a cost-to-go metric for each state, (f) *Line 22:* $N \times S$.

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 improves efficiency but introduces the probability of collision due to resolution issues. 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 deacceleration also assist computationally in the case of braking maneuvers.

C. Guaranteeing Maintenance of the Safety Invariant: This section provides a proof that Algorithm 4.1 maintains the safety invariant given some simplyfying assymptions that will be waived later.

Theorem 1: Algorithm 4.1 guarantees the maintenance of the safety invariant in every planning cycle given it holds during the cycle ($t_0^i : t_1^i$) and that:

i) all robots can communicate one with another,

ii) plans are transmitted instantaneously between robots.

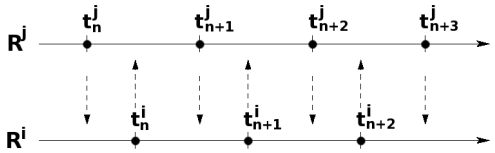


Fig. 2 The replanning cycles of two neighboring robots R^i and R^j. The times denote transitions between planning cycles for each robot. The vertical arrows denote the transmission of information, e.g., at t^i_n, R^i transmits $\pi^i(\,\pi^i(x^i(t^i_n)), p^i(t^i_n : t^i_{n+1})),\ \gamma(t^i_{n+1} : \infty)\,)$.

Proof: The proof is obtained by induction. The *base case* holds for R^i because of the Theorem's assumption that the Invariant holds during cycle $(t^i_0 : t^i_1)$. The *inductive step* will show that if the Invariant holds during the cycle $(t^i_n : t^i_{n+1})$ then it will also hold during the cycle $(t^i_{n+1} : t^i_{n+2})$ for Algorithm 4.1. Without loss of generality consider Figure 2 and focus on robot R^i. To prove the inductive step, it is necessary to show that each one of the three points of the Invariant will be satisfied during $(t^i_{n+1} : t^i_{n+2})$. For cycle $(t^i_{n+1} : t^i_{n+2})$ there are two cases: (1) A compatible trajectory $\pi^i_* = \pi^i_\gamma \in \Pi^i$ is selected, or (2) the current contingency is returned.

Case 1: A trajectory $\pi^i_\gamma \in \Pi^i$ is selected.

a) Trajectory π^i_γ has to be collision-free as part of Π^i.

b) Assuming instantaneous plan transmission and by time t^i_{n+1}, R^i has been sent and has available the choices of other robots for cycles that start before t^i_{n+1}. Since $\pi^i_\gamma \in \Pi^i$ is selected, none of these messages arrived during the compatibility check. This means that R^j's trajectory $\pi^j(\,\pi^j(x^j(t^j_{n+1}), p^j(t^j_{n+1} : t^j_{n+2})),\ \gamma(t^j_{n+2} : \infty)\,)$ is available to R^i during the compatibility check. Then the cycle $(t^i_{n+1} : t^i_{n+2})$ can be broken into two parts:

i) During part $(t^i_{n+1} : t^j_{n+2})$, the selected plan $p^i(t^i_{n+1} : t^i_{n+2})$ is compatible with $p^j(t^j_{n+1} : t^j_{n+2})$ because the second plan was known to R^i when selecting π^i_γ.

ii) For part $(t^j_{n+2} : t^i_{n+2})$ there are two cases for R^j at time t^j_{n+2}:

- R^j will either select a plan $p^j(t^j_{n+2} : t^j_{n+3})$ that is compatible with $p^i(t^i_{n+1} : t^i_{n+2})$,
- or it will resort to a contingency $\gamma^j(t^j_{n+2} : \infty)$, which, however, is already compatible with trajectory π^i_γ.

In both cases, R^j will follow a plan that is compatible with $p^i(t^i_{n+1} : t^i_{n+2})$.

Thus, the second point b) of the Invariant is also satisfied for robots R^i and R^j.

c) For the third point of the Invariant, the contingency $\gamma^i(t^i_{n+2} : \infty)$ has to be compatible with the future choices of the other robots. Focus again on the interaction between R^i and R^j. There are again two cases for R^j at time t^j_{n+2}:

i) R^j will select a plan $p^j(t^j_{n+2} : t^j_{n+3})$ and a corresponding contingency $\gamma^j(t^j_{n+3} : \infty)$. This plan and contingency respect by construction R^i's contingency $\gamma^i(t^i_{n+2} : \infty)$, since it was known to R^j at time t^j_{n+2}.

ii) Or R^j will resort to its contingency $\gamma^j(t^j_{n+2} : \infty)$, which, however, the contingency $\gamma^i(t^i_{n+2} : \infty)$ respected upon its selection.

In any case, whatever R^j chooses at time t^j_{n+2}, it is going to follow plans in the future that are compatible with $\gamma^i(t^i_{n+2} : \infty)$. Thus, point c) is also satisfied.

Case 2: A contingency $\gamma^i(t^i_{n+1} : \infty)$ was selected.

The *inductive hypothesis* implies that $x^i(t^i_{n+1})$ is a safe state. Thus:

a) $\gamma^i(t^i_{n+1} : t^i_{n+2})$ is collision-free with static obstacles

b) The current plans of all robots will be compatible with $\gamma^i(t^i_{n+1} : t^i_{n+2})$, which was known to them at time t^i_n. Furthermore, $\gamma^i(t^i_{n+1} : t^i_{n+2})$ already respects the contingencies of other robots that might be executed before t^i_{n+1}.

c) The state $x^i[\gamma^i(t^i_{n+1} : \infty)](t^i_{n+2})$ is trivially safe, because R^i can keep executing the same contingency for ever and this contingency will have to be respected by its neighbors, as it will always be known ahead of time.

In both cases, all three points of the Invariant are satisfied for R^i and the inductive step is proved. Thus, if the Invariant holds, the algorithm maintains its validity. □

D. Addressing the Assumptions:

Theorem 1 assumed that messages are transmitted instantaneously and that all the robots communicate one with another. The assumption that plans are transmitted instantaneously will not hold in real-world experiments with wireless communication. Similarly, it is more realistic to assume that robots can communicate only if their distance is below a certain threshold. In the latter case, the proposed approach can be invoked using only point to neighborhood communication and thus achieve higher scalability. The following theorem shows that the safety guarantees can be provided without these restrictive assumptions.

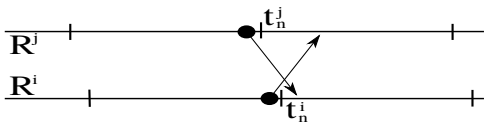


Fig. 3 If messages arrive after the start of a neighbor's future cycle, as with the message from R^j to R^i above, this is problematic.

Theorem 2: Algorithm 4.1 guarantees the maintenance of the safety invariant in every planning cycle given it holds during cycle $(t^i_0 : t^i_1)$ and that:

i) two robots with limited communication ranges can communicate before they enter into ICS given a predefined set of contingencies $\Gamma(\infty)$,

ii) robots utilize acknowledgments that signal the reception of a trajectory by a neighbor.

Sketch of Proof: Theorem 1 showed that the invariant holds as long as it was valid during the first cycle $(t^i_0 : t^i_1)$ and that two vehicles can communicate continuously since t^i_0. For two robots with limited communication range, denote as time t_{comm} the beginning of the first planning cycle of either robot after they are able to communicate. If at t_{comm}, both robots have available a contingency $\gamma(\infty) \in \Gamma(\infty)$, that can be used to prove the safety of their corresponding states, then all the requirements of Theorem 1 are satisfied for $t^i_0 = t_{comm}$. Thus the invariant will be maintained.

Regarding the issue of delayed messages, consider the case that R^j's cycle ends at time t_n^j, which is before the end of the neighboring R^i's cycle at time t_n^i. Figure 3 provides an example. If the transmission of the trajectory π_*^j to R^i is delayed, it might arrive after time t_n^i and R^i cannot detect that it did not take into account the choice of R^j during its compatibility check given Algorithm 4.1. Thus, R^i's choice might end up being incompatible with π_*^j. Notice that this problem becomes more frequent when Algorithm 4.1 is employed by robots that have synchronized cycles. If an acknowledgment message that signals the reception of a trajectory by a neighbor is used, however, R^i can acknowledge the message's reception, whether it arrives before or after t_n^i. If the acknowledgment arrives at R^j before t_n^j (as well as from all other neighbors), it knows that it is safe to execute π_*^j. If the acknowledgment is not received on time, R^j can revert to its contingency which is by construction respected by the future plan of R^i, whatever this is. Thus, the introduction of an acknowledgment resolves the issue of possible delays in the transmission of trajectories. □

5 Experimental Results

To validate the theoretical discussion, simulated experiments were conducted. Our first experiments revealed performance deficits, however, practical modifications in the implementation of the algorithm were made. These resulted in significant speed ups and quick convergence to a solution.

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

- Instead of checking all the candidate plans Π^i with the trajectories of the neighbors $\Pi_{new}^{N^i}$, only the best plan in Π^i according to a metric is checked. If this plan fails the check, then the previous contingency is selected.
- 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_{n+1}^i, a contingency $\gamma(t_{n+1}^i : \infty)$ is extended from $x(t_{n+1}^i)$. If the contingency is collision-free and compatible with the available trajectories of neighbors in $\Pi_{prev}^{N^i}$, $x(t_{n+1}^i)$ is assumed safe. Otherwise, it is unsafe and no future expansion of an edge is allowed past $x(t_{n+1}^i)$.
- The sampling-based expansion of the tree structure of trajectories is biased given a potential field in the workspace that promotes the expansion of the tree towards the goal [2]. The tree expansion is also biased away from other vehicles. Different algorithms can be considered for the actual planning process [20, 15, 8, 23].
- There is no need to differentiate in the implementation between $P_{prev}^{N^i}$ and $P_{new}^{N^i}$. Each robot maintains a buffer for messages from each neighbor. As new trajectories are transmitted, they replace the part of old trajectories that has already been executed by a neighbor along the buffer.

- The latency in the experimental setup was relatively low. Thus, the situation in Figure 3 did not arise. Thus, the acknowledgement step was not included for the experiments presented below, which reduced the number of peer-to-peer messages.

Modeled System: The experiments presented in this paper are using the model of a second-order car like vehicle [19] shown on the right side, where (x,y) are the car's reference point in Cartesian coordinates, θ is the car's orientation, w its velocity and ζ the steering angle. The controls are α, the acceleration, and ϕ the rate of change of the steering angle. There are limits both for state and control parameters ($\|w\| < w_{max}$, $\|\zeta\| < \zeta_{max}$, $\|\alpha\| < \alpha_{max}$, $\|\phi\| < \phi_{max}$). All robots have range-limited communication out to 30% of the total environment width, and brake to zero speed for contingency.

$$\begin{aligned} \dot{x} &= w \cdot cos\zeta \cdot cos\theta \\ \dot{y} &= w \cdot cos\zeta \cdot sin\theta \\ \dot{\theta} &= w \cdot sin\zeta \\ \dot{w} &= \alpha \\ \dot{\zeta} &= \varphi \end{aligned}$$

Environments Four simulated environments were used for the experiments:

1. An "empty" environment (Fig. 4 (left)),
2. an "office" environment (Fig. 1),
3. a " random" environment (Fig. 4 (right)), and
4. an "intersection" environment with two crossing corridors (Fig. 5).

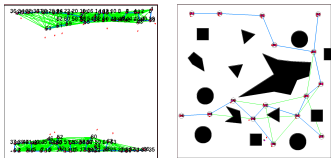


Fig. 4 Starting positions for the "empty" and "random" environments.

These environments are presented in approximate order of difficulty. The various experiments tested different numbers of vehicles: 2, 4, 8, 16, 32, 48. Because the 16 robots alone took up 6% of the entire workspace (ignoring obstacles), the size of the robots was reduced to half for the 32 robot case, and to a quarter of their size for the 48 robot case. If this was not done, then the robots would take up 12% and 18% of the workspace, respectively. Since much of the workspace is already occupied by obstacles, this reduction in size assists in reducing clutter effects that effect solution time. The empty environment was the easiest to solve. The office environment was chosen as a gauge for how hard a structured environment can be. The robots, in their original size, are about 1/5 of the size of the hallway. In the random environment, there were polygons of varying shapes and sizes. 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.

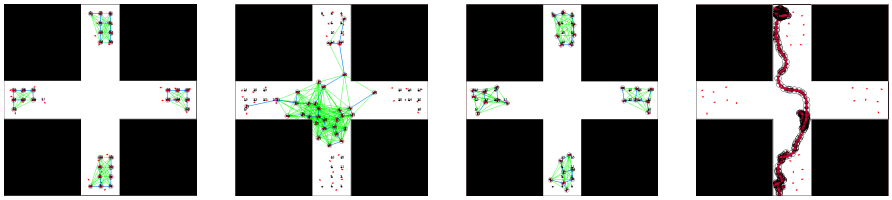
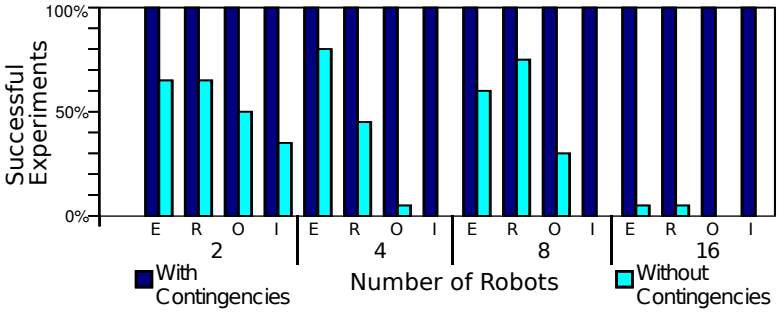


Fig. 5 Snapshots from a typical run with 32 robots; Final image is the full trajectory of robot 0.

When possible, starting/goal locations were identical across runs as more robots were added. Experiments for the same number of robots have the same start/goal

locations. All experiments were repeated at least 10 times. The algorithm was run in real time such that computation time is equal to execution time.

Evaluation of Safety To verify that the system implemented truly provides the guarantees presented in this paper, three different cases were considered for the algorithm: (i) an implementation without contingencies, (ii) with contingencies but for robots with synchronized cycles and (iii) with contingencies and robots that are *not* synchronized. For each type of experiment the following figure reports the percentage of successful experiments. 20 experiments were executed for each case, averaging across synchronous and asynchronous cases. The results presented clearly indicate that enabling contingencies results in a safe system in all cases.



Scalability and Efficiency Once the safety of the approach was confirmed, the focus turned on evaluating the effects of contingencies. A high-selection rate of contingencies is expected to decrease the performance of the robots, as these plans are not selected to make progress towards the goal. The following table presents the average duration of experiments in seconds and the average velocity achieved by the robots both for the case without contingencies and the case with contingencies (both for synchronized and asynchronous robots). The performance data without contingencies is from the cases where none of the robots entered `ICS`, which means they often correspond to fewer than 20 experiments, and in some cases there is no successful experiment without contingencies to compare against.

Effects of Contingencies		Number of Robots							
		2		4		8		16	
Scenes	Approach	Time	Vel.	Time	Vel.	Time	Vel.	Time	Vel.
Empty	Without Cont.	85.1	6.7	84.5	4.8	82.9	3.9	87.5	3.4
	With	82.6	6.9	90.9	4.7	88.8	3.7	335.8	1.4
Office	Without Cont.	97.0	8.3	98.1	6.6	X	X	X	X
	With	99.1	8.2	111.5	5.9	206.9	2.7	553.3	1.0
Random	Without Cont.	87.2	6.5	84.4	4.8	88.3	3.6	X	X
	With	88.0	6.5	103.1	4.4	92.4	3.6	604.8	1.3
Intersection	Without Cont.	101.0	8.0	100.0	8.0	X	X	X	X
	With	108.9	7.5	272.5	4.2	469.1	2.3	1415.4	1.0

The behavior of the robots is indeed more conservative when contingencies are employed and it takes longer to complete an experiment. Although the algorithm has no progress guarantees, the randomized nature of the probabilistically complete planning algorithms helped to offset this. The simulations always eventually

resulted in a solution for the tested problems even if the robots temporarily entered oscillatory motions. The local penalty for trajectories that brought an agent in close proximity to neighboring robots helped to reduce the occurrence of oscillations and resulted in significant improvements in performance.

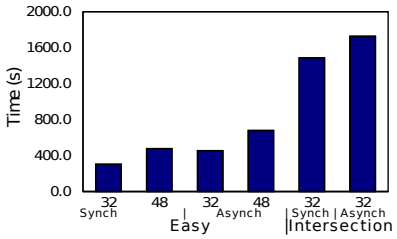
Synchronous vs. Asynchronous Another objective of the experimentation procedure was to evaluate the differences in the performance of the algorithm between the synchronous and the asynchronous case. In the synchronous case, all robots have a zero time offset but they are not aware of their synchronicity and they are not taking advantage of it as in previous work [3]. In the asynchronous case, the offsets are the same across 10 averaged runs. These offsets are randomly precomputed and range from 0 to a maximum of 3/4 of the planning cycle.

Sync. Vs. Async.		Number of Robots							
		2		4		8		16	
Scenes	Approach	Time	Vel.	Time	Vel.	Time	Vel.	Time	Vel.
Empty	Asynch.	81.5	7.0	85.5	4.8	87.3	3.8	400.0	1.4
	Synch.	83.8	6.8	96.3	4.5	90.3	3.6	271.5	1.4
Office	Asynch.	96.0	8.4	112.5	6.0	197.5	2.8	541.0	1.0
	Synch.	102.3	7.9	110.5	5.9	216.3	2.7	565.5	1.0
Random	Asynch.	85.5	6.7	90.8	4.5	85.8	3.8	729.6	1.4
	Synch.	90.5	6.3	115.5	4.2	99.0	3.5	480.0	1.3
Intersection	Asynch.	105.0	7.8	268.3	4.1	335.8	2.9	899.8	1.3
	Synch.	112.8	7.2	276.8	4.3	602.5	1.6	1931.0	0.8

When the robots' cycles are synchronized, then it will be often the case that robots are transmitting simultaneously, and potentially during the compatibility check of their neighbors. This in certain cases results in slightly longer durations for the completion of an experiment, as well as lower average velocities, but overall there is no consistent effect as in the random and empty scenes, there is a performance boost under synchronous operation, especially as the number of robots increases. In comparison to previous work [3] where synchronicity was specifically taken advantage of, it is clear that the quality of the paths selected are worse in the current asynchronous implementation. However, it is expected that further research in asynchronous coordination algorithms can reduce this performance gap.

Scaling Larger scale simulations for 32 and 48 robots were run to study the algorithm's scalability. For these cases, the approach without contingencies always fails. Note that as mentioned earlier, these robots are of reduced size to decrease the effects on completion time due to a cluttered environment.

Achieving safe, asynchronous operation for 48 second-order systems with the proposed setup is a challenge. The agent model is complex as are the safety guarantees address the ICS issue. The simulation environment mimics the constraints of real-world communication by running each agent on a separate processor and allowing only



message-passing communication (TCP sockets). An experiment with 48 robots requires 49 separate processors (1 processor is used as a simulation server).

Parameter Evaluation An important parameter for the proposed approach is the duration of the planning cycle. For shorter durations of cycles, there was a higher deviation between runs and it was not possible to execute the larger experiments with 32 and 48 robots for a cycle duration less than 2 seconds. This limitation is due to the single thread running the world simulation. It is expected that the limit in hardware implementation would be dependent on the communication latency. The average completion time shows a noticeable increase as the duration of a cycle increases. The experiments presented in the previous tables were executed for a cycle duration of 2.5 seconds.

Planning Cycle		Number of Robots							
		2		4		8		16	
Scene	Cycle	Time	Vel.	Time	Vel.	Time	Vel.	Time	Vel.
Empty	1.0s	53.3	10.8	52.5	7.8	59.2	5.8	96.9	3.5
	1.5s	59.3	9.7	63.8	6.4	60.0	5.3	197.1	2.0
	2.0s	71.4	8.0	74.0	5.8	75.6	4.2	116.8	2.7
	2.5s	79.5	7.2	82.8	5.2	86.5	3.7	134.0	2.2
	3.0s	98.4	5.8	98.4	4.4	99.9	3.2	135.0	2.0
	3.5s	167.7	3.8	193.6	2.5	125.5	1.7	482.7	0.7

6 Discussion

This paper presented a fully distributed algorithm that guarantees ICS safety for a number of second-order robots that move in the same environment. Simulations confirm that the framework indeed provides safety and is scalable and adaptable. Additional experiments not presented above were conducted for a system with positive minimum velocity, i.e., a system that cannot brake to zero velocity. Safety was achieved for this system using a different set of contingencies than braking maneuvers. In this case, the system was required to turn into the tightest circle possible without exceeding the specified limits on velocity and turning rate. Future work includes: (a) considering robots with different durations for planning cycles, (b) dealing with unreliable communication, (c) studying the effects of motion uncertainty to the protocol's performance, (d) distributed optimization for improving the quality of paths selected despite the asynchronous operation, (e) dealing with non-cooperating vehicles and (f) addressing tasks that go beyond moving from initial to final states. Experiments using physical systems with interesting dynamics would provide a real-world verification of the approach.

Acknowledgements Work by D. Grady and L. 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. Work by K. Bekris has been supported by NSF CNS 0932423. Any conclusions expressed here are of the authors and do not reflect the views of the sponsors. The authors would like to thank the anonymous reviewers for their helpful comments.

References

1. Alami, R., Simeon, T., Krishna, K.M.: On the influence of sensor capacities and environment dynamics onto collision-free motion plans. In: IEEE/RSJ IROS. Lausanne, CH (2002)
2. Bekris, K.E., Kavraki, L.E.: Greedy but safe replanning under kinodynamic constraints. In: IEEE ICRA. Rome, Italy (2007)
3. Bekris, K.E., Tsianos, K., Kavraki, L.E.: Safe and distributed kinodynamic replanning for vehicular networks. Mobile Networks and Applications **14**(3), 292–308 (2009)
4. Van den Berg, J., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA) (2008)
5. Bruce, J., Veloso, M.: Real-time multi-robot motion planning with safe dynamics. In: A. Schultz, L. Parker, F. Schneider (eds.) Intern. Workshop on Multi-Robot Systems (2003)
6. Chan, N., Kuffner, J.J., Zucker, M.: Improved motion planning speed and safety using regions of inevitable collision. In: 17th CISM-IFToMM RoManSy (2008)
7. Clark, C., Rock, S., Latombe, J.C.: Motion planning for multi-robot systems using dynamic robot networks. In: IEEE ICRA. Taipei, Taiwan (2003)
8. Dimarogonas, D.V., Kyriakopoulos, K.J.: Decentralized Navigation Functions for Multiple Robotic Agents. Intelligent and Robotic Systems **48**(3), 411–433 (2007)
9. Erdmann, M., Lozano-Perez, T.: On multiple moving objects. In: ICRA, pp. 1419–1424 ('86)
10. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. Int. Journal of Robotics Research **17**(7) (1998)
11. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robotics and Automation Magazine **4**(1) (1997)
12. Fraichard, T.: A short paper about motion safety. In: IEEE ICRA. Rome, Italy (2007)
13. Fraichard, T., Asama, H.: Inevitable collision states: A step towards safer robots? Advanced Robotics pp. 1001–1024 (2004)
14. Frazzoli, E., Dahleh, M., Feron, E.: Real-time motion planning for agile autonomous vehicles. AIAA Journal of Guidance, Control and Dynamics **25**(1), 116–129 (2002)
15. Hsu, D., Kindel, R., Latombe, J.C., Rock, S.: Randomized kinodynamic motion planning with moving obstacles. IJRR **21**(3), 233–255 (2002)
16. Kalisiak, M., Van de Panne, M.: Faster motion planning using learned local viability models. In: Proc. of the IEEE Int. Conf. on Robotics and Automation. Roma, Italy (2007)
17. Lalish, E., Morgansen, K.A.: Decentralized reactive collision avoidance for multivehicle systems. In: IEEE Conference on Decision and Control (2008)
18. Lamiraux, F., Bonnafous, D., Lefebvre, O.: Reactive path deformation for nonholonomic mobile robots. In: IEEE Transactions on Robotics, vol. 20, pp. 967–977 (2004)
19. Laumond, J.P. (ed.): Robot Motion Planning and Control. Lectures Notes in Control and Information Sciences 229. Springer (1998)
20. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. IJRR **20**(5), 378–400 (2001)
21. Martinez-Gomez, L., Fraichard, T.: Collision avoidance in dynamic environments: An ics-based solution and its comparative evaluation. In: IEEE ICRA. Kobe, Japan (2009)
22. Peng, J., Akella, S.: Coordinating multiple robots with kinodynamic constraints along specified paths. Int. Journal of Robotics Research **24**(4), 295–310 (2005)
23. Pivtoraiko, M., Knepper, R.A., Kelly, A.: Differentially constrained mobile robot motion planning in state lattices. In: Journal of Field Robotics, vol. 26, pp. 308–333 (2009)
24. Reif, J., Sharir, M.: Motion planning in the presence of moving obstacles. In: Proc. of the IEEE Int. Symp. on Foundations of Computer Science. Portland, OR (1985)
25. Vatcha, R., Xiao, J.: Perceived CT-Space for motion planning in unknown and unpredictable environments. In: Workshop on Algorithmic Foundations of Robotics. Mexico (2008)
26. Wikman, M.S., Branicky, M., Newman, W.S.: Reflexive collision avoidance: A generalized approach. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (1993)
27. Yang, Y., Brock, O.: Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation in dynamic environments. In: Robotics: Science and Systems II (2006)