Robotica: (2021) volume 39, pp. 885–927. © The Author(s), 2020. Published by Cambridge University Press.
doi:10.1017/S0263574720000806

Safe Motion Planning Based on a New Encoding Technique for Tree Expansion Using Particle Swarm Optimization

Sara Bouraine* and Ouahiba Azouaoui

Center for the Development of Advanced Technologies (CDTA), Baba Hassen, Algiers, Algeria

(Accepted August 12, 2020. First published online: September 10, 2020)

SUMMARY

Robots are now among us and even though they compete with human beings in terms of performance and efficiency, they still fail to meet the challenge of performing a task optimally while providing strict motion safety guarantees. It is therefore necessary that the future generation of robots evolves in this direction. Generally, in robotics state-of-the-art approaches, the trajectory optimization and the motion safety issues have been addressed separately. An important contribution of this paper is to propose a motion planning method intended to simultaneously solve these two problems in a formal way. This motion planner is dubbed PASSPMP-PSO. It is based on a periodic process that interleaves planning and execution for a regular update of the environment's information. At each cycle, PASSPMP-PSO computes a safe near-optimal partial trajectory using a new tree encoding technique based on particle swarm optimization (PSO). The performances of the proposed approach are firstly highlighted in simulation environments in the presence of moving objects that travel at high speed with arbitrary trajectories, while dealing with sensors field-of-view limits and occlusions. The PASSPMP-PSO algorithm is tested for different tree expansions going from 13 to more than 200 nodes. The results show that for a population between 20 and 100 particles, the frequency of obtaining optimal trajectory is 100% with a rapid convergence of the algorithm to this solution. Furthermore, an experiment-based comparison demonstrates the performances of PASSPMP-PSO over two other motion planning methods (the PassPMP, a previous variant of PassPMP-PSO, and the input space sampling). Finally, PASSPMP-PSO algorithm is assessed through experimental tests performed on a real robotic platform using robot operating system in order to confirm simulation results and to prove its efficiency in real experiments.

KEYWORDS: Particle swarm optimization; Motion planning; Motion safety; Mobile robots; Dynamic environment.

1. Introduction

In recent years, robotic technology has largely evolved making robots present everywhere, at home, at work, in transportation, in recreational areas, etc. Different researches have been developed to design systems that integrate the human world, to serve humans (e.g., service robotics, transportation, space exploration, accompanying the elderly, leisure robots, companion robots, etc.). Great technological resources and years of research have been deployed aiming at reaching a level of intelligence approximating the human reasoning or animals' behaviour in accomplishing specific tasks. Hence, bio-inspired systems would be an interesting approach to solve many issues when working on robotic systems navigating in the real world.

In fact, a robotic system is subject in real-world situations, to the same constraints as a biological system. Two main constraints should be considered: the first one is related to the system's sensory

* Corresponding author. E-mail: s_bouraine@yahoo.fr

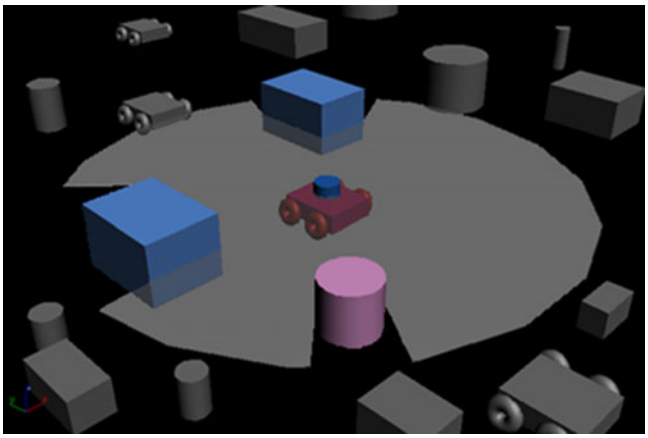

Fig. 1. Robot with a limited field-of-view in an unknown environment with fixed and moving objects. The black region is unobserved and may contain unexpected objects.

limitations (presence of unseen objects in the limits of the robot's field-of-view and occluded regions) and the second one is related to the dynamics of the environment (presence of moving objects with unknown future behaviour) (see Fig. 1).

The purpose of this paper is precisely to propose a motion planning method that handles such challenging constraints using an appropriate state × time model of the future, provides strict guarantees of motion safety with formal proofs and solves the trajectory optimization problem by proposing a technique that formally integrates safety to compute an optimal trajectory to the goal.

Given that motion safety has to do with staying away from states where a collision occurs (now or eventually), the first position taken in this work is to address the motion safety issue. In an ideal case, motion safety is the guarantee that no collision between the robot and its surroundings will ever occur whatever happens. Theoretically, this form of safety (called *absolute motion safety*) consists in finding a collision-free trajectory of infinite duration, which requires knowledge of the future up to infinity (*a priori* known objects behaviour). In situations such as Fig. 1, this form of safety is impossible to guarantee.[1] In a previous work,[2] this problem was solved by considering a weaker level of safety (better guarantee less than nothing) by guaranteeing that the robot will be at rest before an inevitable collision occurs whatever happens. A passive motion safety is therefore guaranteed where the robot takes its own responsibility with respect to the collision problem by braking and stopping before collision occurs. This choice was provably argued in ref. [2] the proposed solution is based on *the braking inevitable collision state concept* [braking inevitable collision states (ICS)] (for more details, see refs. [2, 3]). This form of safety is integrated in the motion planning process presented in this paper.

In order to solve a motion planning problem, two ways are possible: global approaches or local approaches. A global approach (e.g., refs. [4, 5]) computes a complete trajectory from an initial state to the goal. It is most often adapted for static environments and requires *a priori* knowledge of the environment. Whereas, a local approach uses the on-board sensors information of systems to calculate at each time step the control to apply and hence reasons only on the next step. Due to such reasoning, it lacks foresight to reach a goal. In the literature, these approaches are more known as obstacle avoidance approaches (e.g., refs. [6–8]). Given the constraints discussed above, none of the global or local methods offer a viable solution to our problem. Therefore, the reactive planning is an alternative solution to gain on reactivity and improve convergence towards the goal. As its name suggests, it is motion planning with a reactivity aspect that can be present in several ways. Some approaches compute a complete trajectory to the goal and when changes in the environment occur, this trajectory is re-planned while similar methods recompute only the obstructed part by an object. Other approaches adopt a completely different concept by reasoning over many time steps (i.e., for a given lookahead) and accordingly generate a partial trajectory. The whole set of partial trajectories drives the system to the goal. This is the approach adopted in this paper; in ref. [9], a safe partial motion planning (PMP) concept has been proposed. It allows planning safe partial trajectories given a regular update of the environment's evolution. However, the problem of trajectory optimization has not been addressed at all.

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

Among existing approaches (see Section 3), bio-inspired stochastic approaches present better performances to solve optimization problems, particularly the particle swarm optimization method (PSO)[10] which is famous for its efficiency and fast convergence. In this paper, the developed approach is dubbed PASSPMP-PSO. It is a modified version of the passively safe partial motion planner[9] based on the PSO method to solve the trajectory optimization problem. PASSPMP-PSO has to react during a limited period of time related to a given lookahead (*Aka* time horizon, i.e., depicting how far into the future the reasoning is used) during which motion safety must be guaranteed to find a near-optimal partial trajectory. To do so, a tree encoding technique is proposed, where partial trajectories are encoded as possible solutions (particles).

The paper is organized as follows: Section 2 states the motivation, contribution and novelty of this work. A review of the relevant literature is shown in Section 3. Section 4 presents the proposed approach. The motion planning scheme is explained in Section 4.1. The trajectory optimization-based approach is illustrated in Section 4.2, whereas the particle's encoding technique is itemized in Section 4.3 and the used objective function is given in Section 4.4. Validations in simulation and real-world experiments are given in Section 5. Finally, Section 6 includes a theoretical comparison of the proposed approach to other approaches in the field.

2. Motivation and Contribution

Motion safety and trajectory optimization are two closely related issues. In biology, the problem has been already addressed; fish schooling or bird flocking, for example, can both optimize individuals' motion and remain safe from a predator, thanks to the concept of "safety in numbers".[11] However, in robotics, most state-of-the-art approaches addressed the two issues separately, even if a natural way to evolve for a system is both accomplishing a mission in an optimal manner and remaining safe with regard to itself and its environment. As stated above, many trajectory optimization approaches are inspired from animals' swarm behaviour such as PSO, but they focus solely on the optimization problem. About the motion safety, it is a critical issue when navigating in the real world and among humans. Despite the rapid advance of robotic technology, developed robotic systems still fail to meet the challenge of motion safety. An eminent example is the self-driving vehicles, which can cause material and human damage. The first crash caused by a wrong decision (i.e., due to vehicle collision avoidance system) occurred in February 2016.[12] Such examples prove that safety is still not guaranteed. For a long time, motion safety notion has been ill-defined and it has been mainly addressed as a collision-free problem. However, it is now agreed that safety is more than a mere collision avoidance, but it is the ability of staying away from ICS, in any given situation or circumstance.[13] An ICS is a state for which, no matter what the future trajectory of the robot is, a collision eventually occurs. However, ICS are defined with an absolute motion safety perspective and therefore require to reason about the future evolution of the environment with an infinite lookahead. This is impossible to guarantee in real-world constraints. An alternative to the ICS concept in such challenging conditions is the braking ICS concept (braking ICS or ICS^b),[2,3] that is, a version of the ICS concept corresponding to passive motion safety. Braking ICS are defined as states such that, whatever the future braking trajectory followed by the robot is, a collision occurs before it is at rest. Passive motion safety is obtained by avoiding braking ICS at all times. Unlike existing trajectory optimization approaches, the proposed approach PASSPMP-PSO has the ability to generate near-optimal trajectories when staying away from braking ICS whatever happens, that is, near-optimal passively safe trajectories.

The main contributions and novelties of this paper are the following:

(1) We propose a new motion planner dubbed PASSPMP-PSO that integrates both safety guarantee and trajectory optimization in challenging scenarios as in Fig. 1. Among the few works rigorously addressing the safety issue (e.g., refs. [9,14]), none of them deals with trajectory optimization even if the two issues are strongly related (both are necessary for a good performance of a mission).
(2) PASSPMP-PSO computes a passively safe near-optimal partial trajectory, where motion safety and trajectory optimization are addressed in a formal way. PASSPMP-PSO reasons about the future evolution of the environment over a limited lookahead, which is formally defined so as to ensure passive safety guarantee. In addition, PASSPMP-PSO is based on a PMP concept to ensure regular update of the environment and therefore respect real-world constraints.

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

(3) PSO is a widely used method, thanks to its efficiency and fast convergence. However, to the authors' knowledge, this is the first time that motion safety (in the sense of avoiding ICS whatever happens) is integrated in PSO and also the first time that this issue is tackled in an optimization problem. The optimal solution is defined, thanks to an objective function based on three important constraints: passive motion safety guarantee, minimum trajectory time cost and minimum distance to the goal. Therefore, the approach used will be a variant of PSO.
(4) A new tree encoding technique is proposed, a modified priority-based encoding technique for trees. Previous approaches have addressed the encoding-based shortest path problem in the case of a network graph, while, in this paper, it is applied for rapidly exploring random trees (RRTs) or to find the optimal trajectory in a tree.
(5) PASSPMP-PSO deals with real-world constraints, moving objects with unknown future behaviours, both seen and unseen (i.e., objects present in the limits of the robot's field-of-view and occluded ones) with a conservative model of the future. All these constraints are addressed at once, unlike previous approaches that deal with only some constraints or none. PASSPMP-PSO also provides feasible trajectories and it is very suitable for solving high dimensional problems in the state × time space, thanks to its tree-based expansion strategy.
(6) PASSPMP-PSO has been tested in simulation to demonstrate its performances under challenging constraints (e.g., crowded environments with objects travelling at high speed, etc.) and in real experiment to validate the algorithm in real-world situations on an experimental robotic system. It should be noted that most existing approaches dealing with the motion safety issue (e.g., refs. [9, 14]) are validated only in simulation. Finally, PASSPMP-PSO performances have been compared, for the same test conditions, to a previous work presented in ref. [9] and to input space sampling (ISS), a commonly used reactive planning method known to be among the best when dealing with unknown environment.

3. Related Works

As stated above, a still open issue is the paradigm motion safety versus trajectory optimization, *should the robot select the safest trajectory or the most optimal*? So far, these two issues have been addressed separately. Furthermore, most motion planning schemes do not deal with the safety issue.

When it comes to motion safety, most approaches focus on providing a safety guarantee without worrying about reaching a goal in an optimal way[9,14] or not.[15,16] In the latter case, a simple cost function is generally used, for example, minimal distance to the goal. It is important to emphasize that there is a rich literature in collision avoidance approaches;[6,7,17–20] however, all of them provide only collision-free guarantee where motion safety is not considered at all. Motion safety requires guaranteeing that collision will never occur. When a mobile robot is led to navigate in the real world, the dynamics of moving objects should be considered because even if a collision does not occur at the present time, there is no guarantee that it will not occur in the future. Consequently, it is necessary to design a navigation scheme for which motion safety is guaranteed. Among the few works addressing safety problem, many do not consider moving objects at all (only fixed objects are considered).[21–24] Other works cope with this problem by solving a multi-robots collision avoidance problem (both robot and moving objects are considered as robots).[25–28] Collision is avoided by coordinating the movement of the whole robots. However, this solution is ineffective in the presence of uncontrollable moving objects (which is the case in the real world). There are few approaches that take into account the future behaviour of moving objects and use either a probabilistic model[1,31,32] or a deterministic one.[7,14,15,29,30] Generally, with deterministic models, one considers objects with a constant velocity or *a priori* known trajectory. Both approaches are interesting, but there is no guarantee of safety. Another real-world constraint that should be considered is the limited field-of-view of the robot and occlusions. When using the embedded sensors of the robot to perceive the environment, only a partial view of its surroundings is possible, which implies that some objects are perceived while others are not. From the safety perspective, these unseen objects represent a risk in case they are not handled in time. There are few works that take into consideration these constraints: the problem of regions hidden by obstacles (occlusions) is tackled in refs. [33, 34] and the problem of limited field-of-view is addressed in ref. [35] for a static environment and in ref. [14] for dynamic environment, but it is assumed that objects should always be maintained within the robot's field-of-view.

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

Therefore, an important contribution of the proposed approach PASSPMP-PSO is to formally guarantee motion safety while considering at once limited field-of-view, occlusions and unknown future behaviour of both seen and unseen objects. To do so, a conservative model of the future is used to handle all possible future trajectories of each object during a given lookahead.

There is a rich literature to solve the motion planning issue. However, from the paradigm global approaches versus local approaches, reactive planning is the most suitable solution when dealing with partially observable environments. Mainly, two types of methods can be distinguished: the first type is a variant of the global approaches that has been modified so as to deal with environment changes. There are simple methods such as trajectory deformation (TD)[36,37] and discrete planning methods (e.g., A*3D, D*, etc.),[38–40] for which the trajectory is either deformed or recomputed in case it is obstructed by a new obstacle. Even if these methods are reactive to environment changes, most often the future behaviour of moving objects is not considered or it is assumed *a priori* known or short-term predicted. Moreover, *a priori* knowledge of the environment is necessary as an initial plan is required. Similarly, methods like RRT[33] have been modified to address dynamic environments.[41–45] Each variant offers solutions and has drawbacks. For example, in Refs. [42] and [43] a linear velocity of objects is considered but the computing time is very expensive, in ref. [41] (anytime RRT) objects trajectories are not considered but suboptimal solutions are guaranteed while another variant (RRT*)[46] is asymptotically optimal for static environments only. However, for the whole cases, objects trajectories are not considered or assumed *a priori* known. For the second type of methods, instead of planning a complete motion to the goal, planning is carried out for a certain period of time. Therefore, sampling methods like ISS[47–50] and state space sampling (SSS) techniques[47,51] are best suitable for unknown environment, where the robot uses its embedded sensors for perception. With a similar concept, tentacles-based approach[52,53] uses different curvatures discretizing the basic driving options of a vehicle. These approaches are local planners and require a global planner to guide the robot towards the goal. Approaches like global dynamic window[54,55] integrate a reactive approach to determine the control to apply and a global planner to solve the motion planning problem. However, all these methods do not consider objects trajectories and are inefficient in complex environments. More interesting methods plan over an appropriate time horizon where the information about the environment is regularly updated, for example, the PMP method[56] and the model predictive control (MPC) method.[57–60] A serious drawback with MPC is the lack of a guarantee that the environment's model is still valid over the planning period, where no constraint is considered with respect to this time duration. Besides, PMP considers a decision time constraint for planning, and the model of the future is assumed valid during planning and execution. Nevertheless, only collision-free trajectories based on *a priori* known model of the future can be provided with no formal guarantees. In this case, motion safety is not guaranteed. Even so, the concept of planning partial trajectories is very interesting, because it allows world model update (necessary when only a partial view of the environment is available) and it is very suitable for real-time applications as the planning and execution processes are interleaved. For this reason, one purpose of this paper is to adopt a planning strategy based on a PMP concept, but for which the problems of safety and trajectory optimization are solved.

Regarding the second part of the paradigm motion safety versus trajectory optimization, in motion planning, an important issue that has been tackled for a long time is the trajectory optimization problem. Among the above approaches, A* and Dijkstra[61,62] can offer a potential solution, but they are limited to low dimensional problems. Recent works on trajectory optimization include stochastic trajectory optimization for motion planning (STOMP) based on a gradient-free stochastic optimization[63] and covariant Hamiltonian optimization for motion planning (CHOMP) which is a gradient-based method.[64,65] These methods optimize individual trajectory steps. To deal with obstacles, a pre-computed distance for collision checking in trajectory optimization was integrated. These methods are computationally expensive and can be subject to local minima. Bio-inspired stochastic optimization approaches such as genetic algorithms (GA),[66] ant colony optimization (ACO),[67] neural networks (NN)[68,69] and PSO[10] present better performances to find a global optimum. Indeed, for the last few years, a great deal of interest has been brought to bio-inspired-based approaches to solve motion planning problems, such as in refs. [70–74]. Most often, the kinodynamic constraints of the system and the future evolution of the environment are not considered. Among these methods, PSO is the most performant to solve different optimization problems. This method is interesting due to its simplicity, its low computing cost and fast convergence compared to the other optimization methods. However, as the other optimization-based approaches, the safety issue is not tackled at all.

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

The most relevant works mentioned above, dealing with the motion planning issue, are summarized in Table I. They are evaluated with respect to different constraints: robot's dynamics, planning using sensors, unknown and dynamic environment, reasoning about the future, considering unseen objects (field-of-view limits and occlusions), safety guarantee and optimization.

The main contribution of this paper is to propose a motion planning approach, called PASSPMP-PSO that solves both motion safety and trajectory optimization issues. It is based on a new tree encoding technique based on PSO to find an optimal safe trajectory that drives the robot from an initial state to a goal state. Furthermore, unlike previous works, PASSPMP-PSO considers at once limited field-of-views, occlusions and unknown future behaviour of objects (for both seen and unseen objects). PASSPMP-PSO is therefore able to meet all the addressed constraints in Table I, while the other methods fail.

4. The Proposed Approach: PASSPMP-PSO

The main purpose of this paper is to generate a trajectory from an initial state to a goal state for a mobile robot navigating in dynamic and partially observable environment, where both motion safety and trajectory optimization issues are tackled. This work is therefore about solving a motion planning problem by finding an optimal and safe trajectory. The proposed approach PASSPMP-PSO has been built upon the following main points:

- In its general form, the motion planner scheme is based on the process of interleaving planning and execution, adopted in ref. [9]. During this process, a partial trajectory is generated at each planning cycle. The main advantage of this planning scheme is to react to the environment dynamics given an appropriate time horizon (formally set) and to manage the real-time constraint. Section 4.1 provides a description of this process.
- The trajectory optimization problem is solved, thanks to a tree encoding technique based on the PSO method. For more clarity, the basic principal of PSO is first introduced in Section 4.2. In PSO, the population (the swarm) is represented by a set of particles that are moving in the search space, representing possible solutions to the optimization problem. Each particle is assigned position and velocity vectors. From a formal point of view, PSO is mainly based on two equations (Eqs. (2) and (3)) to update these vectors that are continuously recomputed through optimization iterations, given two key parameters: the personal best position and the global best position. These parameters should be computed, thanks to a specific objective function, where at the end of the optimization process, the optimal solution corresponds to the global best particle found so far.
- After introducing PSO, the tree encoding technique is presented in detail in Section 4.3. A new approach is proposed, it is a modified priority-based encoding technique for trees. The main objective is to find the optimal safe partial trajectory in an RRT (expanded during a planning cycle). The primary issue lies in how the particles are encoded. In this step, the particle is represented by a vector of priorities corresponding to a position vector, which is associated with the nodes of the expanded tree. The second issue is how to construct a trajectory from this priorities' vector. The whole construction process is explained and supported by Algorithm 1. This trajectory is constructed based on a minimum cost value (given Eq. (4)) depending on both priorities of nodes and cost of trajectory's primitives (state trajectories delimited by two nodes). However, one condition that must be checked is the validity of the constructed trajectory, which is explicitly stated in Definition 1.
- At the end of the previous step, several trajectories are constructed representing possible solutions to the motion planning problem. The optimal solution (best particle) is selected, thanks to the objective function provided in Eq. (5) that satisfies three main constraints: (1) passive motion safety: it is a level of safety which is defined with respect to the braking ICS concept.[2] A passively safe state is a braking ICS-free state. Indeed, braking ICS states must be avoided by the robot in order to guarantee a passive safety. Definitions 2 and 3 define, respectively, what a passively safe state and a braking ICS state is, and they are respectively formulated by Eqs. (6) and (7). Furthermore, to determine the passive safety of a whole trajectory, Property 1 is necessary. After establishing this property and definitions, the weighting factor associated with safety constraint can be expressed by Eq. (8). Noting that this parameter can be improved by including sensory

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

Table I. Review of motion planning methods. (–) the constraint is verified or can be verified in certain conditions.

Methods	Kinematic constraints	Dynamic constraints	Sensory information	Unknown environment	Unseen objects	Dynamic environment	Future reasoning	Safety	Optimization
TD[36]	Yes	Yes	–	–	No	Yes	–	No	–
PMP[56]	Yes	Yes	Yes	Yes	Yes[34]	Yes	–	–	–
Tentacles[53]	Yes	No	Yes	Yes	No	Yes	No	No	No
ISS[49]	Yes	Yes	Yes	Yes	No	Yes	No	No	No
SSS[51]	Yes	Yes	Yes	Yes	No	Yes	No	No	No
AD*[35]	Yes	Yes	–	–	No	Yes	–	No	–
MPC[59]	Yes	Yes	Yes	Yes	No	Yes	Yes	No	–
RRT[41]	Yes	Yes	Yes	Yes	No	Yes	– [43,45]	No	Yes[46] (in static environment)
STOMP[63]	Yes	No	–	No	No	No	No	No	Local minima
CHOMP[64]	Yes	No	–	No	No	No	No	No	Local minima
GA-based planning[70]	No	No	No	No	No	No	No	No	Yes
ACO-based planning[71]	No	No	No	No	No	No	No	No	Yes
NN-based planning[72]	No	No	No	No	No	No	No	No	Yes
PSO-based planning[73]	No	No	No	No	No	No	No	No	Yes (fast convergence)

uncertainty (given Eqs. (9)–(17)). (2) A minimum distance to the goal formulated by Eq. (18). (3) A minimum time cost of the trajectory formulated by Eq. (19).

- The passive safety guarantee of PASSPMP-PSO is formally proved in Section 4.5. At first, PASSPMP-PSO time cycle must be set while respecting a decision time constraint (as the robot has a limited time to return a decision otherwise a collision occurs), given Eq. (20). This decision time is defined, thanks to Definition 4. Secondly, Property 2 with its corresponding proof demonstrates that at each planning cycle, a passively safe trajectory (ICS^b-free) is found. Finally, the model of the future used (conservative model) must still be valid over a limited interval of time to guarantee safety, given Property 3 (with the corresponding proof). This time interval depends on the time horizon defined in Eq. (21).
- The optimality and stability of PASSPMP-PSO are addressed in Section 4.6.
- The overall process of the PASSPMP-PSO algorithm is detailed in Section 4.7.

The reader is referred to Table II for the list of variables and parameters used in this paper.

4.1. Motion planning scheme

To deal with the environment dynamics and robot sensory limitations, PASSPMP-PSO is based on a PMP concept. A detailed study about the planning strategy has been provided in ref. [9], where the developed approach is called PASSPMP. It is a cyclic process whose basic principle is to compute a partial trajectory during a limited period of time δ_{cycle}, given an updated model of the future $\mathcal{W}$ (a space $\times$ time representation). The whole set of computed trajectories over planning cycles drives the robot $\mathcal{A}$ to its goal (see Fig. 2). $\mathcal{A}$'s dynamics is described by the following differential equations of the form:

$$\dot{s} = f(s, u) \tag{1}$$

where $s \in \mathcal{S}$ is the state of $\mathcal{A}$, $\dot{s}$ its time derivative and $u \in \mathcal{U}$ a control. $\mathcal{S}$ and $\mathcal{U}$ are, respectively, the state space and the control space of $\mathcal{A}$. The state of $\mathcal{A}$ at time t_k is denoted s_k.

The state $\times$ time space of $\mathcal{A}$ is explored based on a variant of the *rapidly exploring random tree approach*, a diffusion technique that extends a sub-tree during each cycle.[9] The tree is a set of nodes and primitives (state trajectories delimited by two nodes) that is incrementally built through different levels of extension (depths). Each node is extended by a fixed set of controls (i.e., an exhaustive search). Passively safe partial trajectories are generated while considering a limited lookahead (*aka* time horizon) T_h. The time horizon is how far reasoning is done in the future. This parameter is critical, the motion safety guarantee is not just about keeping the robot away from collision states, but also keeping it away from the states that would drive it to a collision at a given point in the future. That is why reasoning about the future is required. For safety purpose, the robot must have enough time to avoid the collision. In general, it is assumed that a prior knowledge about the future behaviour of the environment is available. In this case, to guarantee that the robot will never reach an ICS, risks of collision are checked through an infinite time horizon. However, when only a partial knowledge of the environment is available (the robot has a limited field-of-view), the previous assumption is no longer valid. Therefore, safety must be guaranteed over a finite time horizon. This amounts to ensure that no collision will occur during this time limit. The choice of this time parameter is defined in Section 4.5.

This same safety level is adopted in PASSPMP-PSO. Nevertheless, in PASSPMP, the optimization problem has not been addressed. PASSPMP-PSO tackles both the two issues and uses an encoding technique to construct partial trajectories to deal with the huge search spaces.

4.2. Trajectory optimization using PSO

In this paper, the trajectory optimization problem is solved using the PSO approach. Among relevant trajectory optimization-based methods,[63–65] PSO has the best performances to find a global optimum with less computing cost. It is an efficient stochastic population-based optimization approach inspired by the social behaviour of bird flocking or fish schooling, proposed by Kennedy and Eberhart.[10] A detailed review on PSO is presented in ref. [75]. Even if the flock behaviour sometimes seems anarchic when birds suddenly change their direction, scatter or regroup, this behaviour is very intelligent. Indeed, when looking closely, a huge number of birds move synchronously based on

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

Table II. Variables and parameters.

Notation	Description
$\mathcal{A}$	The robotic system
s	State of $\mathcal{A}$
$\dot{s}$	Time derivative of s
u	A control
$\mathcal{S}$	State space of $\mathcal{A}$
$\mathcal{U}$	Control space of $\mathcal{A}$
s_k	State of $\mathcal{A}$ at time t_k
(x, y)	Cartesian coordinates of $\mathcal{A}$
θ	Orientation of $\mathcal{A}$
v	Linear velocity of $\mathcal{A}$
ξ	Steering angle of $\mathcal{A}$
u_α	Linear acceleration
u_ξ	Steering angle velocity
L	Wheelbase of $\mathcal{A}$
$\mathcal{A}(s)$	Closed subset of the workspace occupied by $\mathcal{A}$ when it is in the state s
$\tilde{u}_b$	A braking trajectory
$\tilde{\mathcal{U}}_b^s$	Set of all possible braking trajectories for the state s
t_b	Duration of the braking trajectory $\tilde{u}_b$
$\tilde{s}(s, \tilde{u}_b, t)$	State reached at time t starting from s while following $\tilde{u}_b$
$\mathcal{W}$	Model of future (the dynamics of the environment's objects)
δ_{cycle}	Planning cycle time
T_h	Time horizon
Π_k	Partial trajectory executed during a cycle k
p	A particle in the swarm
N_p	Swarm size
$X(p)$	Position vector of p
$V(p)$	Velocity vector of p
f	Objective function of the optimization algorithm
τ	Optimization process iteration
Pbest	The personal best position (among all traversed positions by the particle)
Gbest	The global best (among all positions traversed by the whole swarm thus far)
w_c	Cognitive parameter, affecting attractive forces towards a personal best
w_s	Social parameter, affecting attractive forces towards a global best
$rand_i$	Random variables uniformly distributed in the range [0, 1]
N_{tree}	Number of nodes in the tree
η_{root}	Root of the tree
d_{tree}	Tree depth
$\Pi_{particle}$	Constructed trajectory
η	A tree node
$\delta\Pi$	A trajectory primitive
w_{ps}	Safety weighting factor
w_d, w_t	Distance and time weighting factors
$Cost_d$	Distance cost function of a trajectory
$Cost_t$	Time cost function of a trajectory

a social interaction and some individual skills with the final objective to reach food. This complex flock behaviour can be simply formalized as follows: *to reach the goal (food), every bird has to learn from its own experience and the experience of its neighbours thanks to its memorization and communication abilities with others, and so it continuously updates its position*.

This motion to the quest for food looks like a search of a solution for an optimization problem, where every bird is a possible solution to the problem. In PSO, a member of the population

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

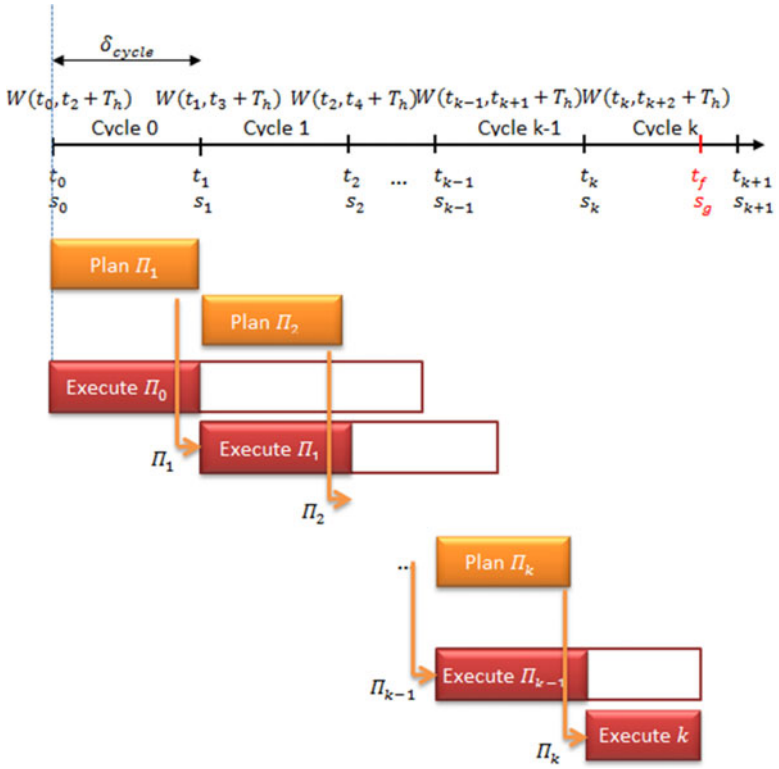

Fig. 2. Planning/execution process timeline. During a cycle k, while the partial trajectory Π_k (computed in cycle $k-1$) is executed, the planner computes a passively safe partial trajectory Π_{k+1} to be executed in cycle $k+1$. The world model $\mathcal{W}$ is periodically updated and remains valid over the time interval $[t_k, t_{k+2} + T_h]$, sufficient to support the planning and execution of Π_{k+1} besides to the passive safety of s_{k+2}.

is dubbed *particle* representing a possible solution to the optimization problem in the search space. Each particle p in the swarm is assigned a position $X(p)$ and a velocity $V(p)$, for $p = 1, ..., N_p$; N_p the swarm size. These particles are moving in the search space and continuously update their positions and velocities given two key information. The first one is the personal best position (the particle's own best position), that is, the best position among all particle's traversed positions, denoted *Pbest*. The second one is the global best position which is the best position among the whole swarm's traversed positions so far, denoted *Gbest* (i.e., the optimum *Pbest* among all particles). It represents the global optimum. The best positions can be determined, thanks to a specific objective function (fitness) f. This function must answer three issues: passive motion safety, minimum trajectory's time cost and minimum distance to the goal. By minimizing this function, the corresponding particle will verify both the safety guarantee condition and the trajectory optimality among the set of all possible solutions.

The formal definitions to update the velocity and hence position of a given particle p for one optimization step τ are given in what follows:

$$V_{\tau+1}(p) = V_\tau(p) + w_c|rand_1|(Pbest_\tau(p) - X_\tau(p)) + w_s|rand_2|(Gbest_\tau - X_\tau(p)) \quad (2)$$

$$X_{\tau+1}(p) = X_\tau(p) + V_{\tau+1}(p) \quad (3)$$

$V_{\tau+1}(p)$ and $X_{\tau+1}(p)$ are, respectively, the velocity and the position of a particle p at iteration $\tau + 1$, where $V_{\tau+1}(p)$ must be upper bounded; $V_{\tau+1}(p) \in [-V_{pmax}, +V_{pmax}]$. $rand_1$ and $rand_2$ are random variables uniformly distributed in the range [0, 1]. w_c and w_s are acceleration coefficients, also known, respectively, as cognitive and social parameters affecting attractive forces either towards a personal best *Pbest* or a global best *Gbest*. Indeed, the swarm displacement can be easily distinguished from Eq. (2) where three main behaviours can be highlighted: (1) the particle is urged to move continuously in its current direction (the first term; $V_\tau(p)$), (2) the particle movement is subjected to a cognitive constraint that attracts the particle to return to its own best position

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

Fig. 3. The particle p, an N_{tree} dimensional vector, with $X(p)$ the particle's position vector.

(i.e., $w_c|rand_1|(Pbest_\tau(p) - X_\tau(p))$) and (3) a social constraint that attracts it towards the best position found so far among the swarm (i.e., $w_s|rand_2|(Gbest_\tau - X_\tau(p))$). All the above parameters vary with PSO iterations and particles.

4.3. Encoding particles for trajectory optimization problem

The previous section introduced the standard PSO method. In PSO, particles represent candidate solutions to the optimization problem, a tricky step of this algorithm is the particles encoding. In PASSPMP-PSO, an optimal safe partial trajectory should be found during each planning cycle. To do so, the adopted solution is based on encoding partial trajectories into particles.

There are two main families of encoding techniques proposed in GA-based shortest path problems. From a given graph[1], they encode a path into a chromosome, but the way of encoding is different leading to two different approaches: *direct* and *indirect encoding* techniques. In the *direct approach*, the path is directly encoded in the chromosome, that is, node IDs are directly represented in the chromosome.[76–78] However, in the *indirect approach*, some guiding information about the path, called priorities, is encoded.[79,80] Given these works (applied mainly in GA), the encoding has been subsequently applied in PSO to solve a shortest path problem (based on graph theory).[81–84] Among the two types of encoding (direct and indirect), only the indirect encoding is suitable for PSO as this latter is based on arithmetic operation for updating the particles' position and velocity (Eqs. (2) and (3)). Direct encoding is more appropriate for discrete optimization. In refs. [81–84], an indirect encoding has been used to find the shortest path in a graph.

However, all the above works have addressed the encoding-based shortest path problem in the case of a network graph. To the authors' knowledge, this is the first time it is applied for RRTs or to find the optimal trajectory in a tree.

To solve the trajectory optimization problem in PASSPMP-PSO, partial trajectories from the expanded tree (denoted TREE, with $\text{TREE} = \{\eta_1, ..., \eta_{N_{tree}}\}$ and $\eta_1 = \eta_{root}$ is the root node) are implicitly encoded into particles. To do so, a *modified priority-based encoding technique* (indirect approach) for trees is proposed; the particle p is represented by a set of priorities associated with the nodes of the expanded tree (during a planning cycle), where the size of the particle corresponds to the number of nodes in the tree, N_{tree}. This vector of priorities corresponds to a position vector $X(p)$ (each position is associated with a tree node) (see Fig. 3). Once this vector is defined, a valid trajectory from the node η_{root} (root of the tree) to a final node (belonging to the last tree depth denoted d_{tree_f}) should be constructed. A valid trajectory in the tree can be defined as follows:

Definition 1 (Valid trajectory). Given an expanded tree, a valid trajectory is a set of nodes, such as all nodes belong to different tree depths and every two successive nodes share the same trajectory primitive.

Let us define $\Pi_{particle}$ the constructed trajectory, η a tree node and $\delta\Pi$ a trajectory primitive.

The whole construction process is illustrated in Algorithm 1. The trajectory is incrementally constructed; at the first algorithm iteration, the trajectory is initialized with η_{root} associated with the ID 1 (from the first tree depth). This node is considered as a parent node. In the next iteration, following Definition 1, the next trajectory node is selected from the next tree depth and this node shares the same primitive with the parent node (with ID: *ParentID*). Therefore, the node to select is a child node (with ID: *ChildID*) from a set of child nodes (set of IDs: *ChildID_SET*) associated with this parent node. The selection is based on a minimum cost value which depends on both the priority of the node and the cost of the primitive relating this node to the parent node $\delta\Pi(ParentID, ChildID)$. This cost can be computed by the following equation:

$$Cost_{ChildID} = X(p)[ChildID]Cost(\delta\Pi(ParentID, ChildID)) \tag{4}$$

[1]A graph consists of vertices (nodes) and edges connecting each two vertices in the graph. A path in this graph is a sequence of edges from a start node to a destination node, where each two successive edges share the same node.

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

Algorithm 1 Trajectory construction in the particle.

Input: particle p; N_{tree}-dimensional vector $X(p)$, TREE (expanded tree), root node= η_{root} (at depth $d_{tree} = 0$), d_{tree_f} (the last depth).
Output: *Traj_IDs* (constructed trajectory IDs for a particle p), $\Pi_{particle}$ (state trajectory corresponding to *Traj_IDs*).

$it_{construct} = 0$;
$ParentID = ID(\eta_{root}) = 1$; //ID of the parent node
$Traj\_IDs(it_{construct}) = \{ParentID\}$;
while $d_{tree} < d_{tree_f}$ **do**
  // for all descendants (children) of the node with ID *ParentID*
  Forall $ChildID \in ChildID\_SET(ParentID)$ **do**
  //compute child cost
  $Cost_{ChildID} = X(p)[ChildID]Cost(\delta\Pi(ParentID, ChildID))$
  // $\delta\Pi(ParentID, ChildID)$ is a trajectory primitive delimited by the two nodes with IDs *ParentID* and *ChildID*
  end
  $ChildID_{selected} = \underset{p=1,\ldots,N_p}{\operatorname{argmin}} Cost_{ChildID}$; //child with minimum cost
  $Traj\_IDs(it_{construct} + 1) = Traj\_IDs(it_{construct}) \cup ChildID_{selected}$;
  $ParentID = ChildID_{selected}$;
  $it_{construct} + +$;
endwhile
return $\Pi_{particle}$ //state trajectory corresponding to *Traj_IDs* (with each state belonging to TREE).

This process is repeated through all tree depths. At the end, the IDs of the constructed trajectory (*Traj_IDs*) for a particle p are returned. The trajectory $\Pi_{particle}(p)$ with the IDs corresponding to *Traj_IDs* represents a possible solution to the optimization problem.

4.4. Objective function

From the previous step (Section 4.3), during a planning cycle, several trajectories are constructed from the expanded tree, representing possible solutions ($\Pi_{particle}(p), p = 1, \ldots, N_p$) to the motion planning problem. In our case, the optimal solution should be selected with respect to three main constraints: passive motion safety, a minimum distance to the goal and a minimum time cost of the trajectory. Consequently, the particles are optimized by minimizing the following objective function f:

$$f(p) = w_{ps}(w_d Cost_d(\Pi_{particle}(p)) + w_t Cost_t(\Pi_{particle}(p))) \tag{5}$$

where w_j are weighting factors and $Cost_j$ are cost functions, both related to the above three constraints.

4.4.1. Safety constraint. Concerning the first constraint, despite the importance to find the shortest trajectory, the first objective of PASSPMP-PSO remains to guarantee passive motion safety, that is, the optimal solution has to verify this criterion. It is based on *the braking inevitable collision state concept* (braking ICS).[2] Let us recall some necessary definitions for PASSPMP-PSO. Definitions 2 and 3 determine, respectively, what a passively safe state and a braking ICS state are:

Definition 2 (Passive motion safety). Given a model of the future $\mathcal{W}$, a passively safe or p-safe state for a system $\mathcal{A}$ is a state s such that there exists one braking trajectory starting at s which is collision-free until $\mathcal{A}$ has stopped.

This definition can be also formulated by :

$$s \text{ is p-safe iff } \exists \tilde{u}_b \in \tilde{\mathcal{U}}_b^s, \forall t \in [0, t_b[, \mathcal{A}(\tilde{s}(s, \tilde{u}_b, t)) \cap \mathcal{W}(t) = \emptyset \tag{6}$$

where $\tilde{u}_b$ a braking trajectory, $\tilde{\mathcal{U}}_b^s$ the set of all possible braking trajectories for the state s and t_b the duration of the braking trajectory $\tilde{u}_b$. $\mathcal{A}(\tilde{s}(s, \tilde{u}_b, t))$ designates the closed subset of the workspace

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

occupied by $\mathcal{A}$ when it is in the state $\tilde{s}$ reached at time t starting from s while following $\tilde{u}_b$. The dynamics of the environment's objects (model of future $\mathcal{W}$) is considered to determine a collision-free braking trajectory ($\mathcal{A}(\tilde{s}(s, \tilde{u}_b, t)) \cap \mathcal{W}(t) = \emptyset$, in Eq. (6)).

Definition 3 (Braking ICS). A braking ICS (ICS^b) is a state for which whatever the future braking trajectory of $\mathcal{A}$, it is impossible to stop before collision.

Formally, a braking ICS (ICS^b) is defined by the following equation:

$$\begin{aligned}\text{ICS}^b(\mathcal{W}) = \{s \in \mathcal{S} | \forall \tilde{u}_b \in \tilde{\mathcal{U}}_b^s, \exists t \in [0, t_b[, \\ \mathcal{A}(\tilde{s}(s, \tilde{u}_b, t)) \cap \mathcal{W}(t) \neq \emptyset\}\end{aligned} \tag{7}$$

From Definitions 2 and 3, a state that is a braking ICS is a no p-safe state (and inversely). As a result, this type of states must be avoided by the robot to guarantee a passive safety (see ref. [2] for more details).

Furthermore, the following property is necessary to determine the p-safety of a given trajectory:[9]

Property 1 (Passively safe trajectory). Given a model of the future and a state trajectory Π with an initial state s_0 and a final state s_f, if (1) Π is collision-free and (2) s_f is ICS^b-free, then every state of Π is ICS^b-free. Therefore, Π is ICS^b-free; by duality, Π is p-safe.

Proof. Let us assume that: (1) s_2 (belonging to the state trajectory between s_0 and s_f) is p-safe, *that is,* it exists at least one braking trajectory starting at s_2 which is collision-free until $\mathcal{A}$ has stopped. (2) s_1 is not p-safe, *that is,* whatever existing braking trajectories, a collision occurs (no collision-free braking trajectory).

However, as the state trajectory between s_0 and s_f is collision-free and there exists a braking trajectory for the state s_2, starting from s_1, $\mathcal{A}$ can brake without a collision occurring. Therefore, s_1 is p-safe, a contradiction with (2).

In order to address the passive safety constraint, in Eq. (5), an enabling factor w_{ps} representing the safety factor is used. It can be expressed as follows:

$$w_{ps} = \begin{cases} \infty & \text{if } \Pi_{particle}(p) \text{ is not p-safe (i.e., } \text{ICS}^b) \\ 1 & \text{otherwise} \end{cases} \tag{8}$$

The braking ICS-ness of $\Pi_{particle}(p)$ can be verified, thanks to Property 1 and the ICS^b-CHECK algorithm (which checks whether a given state is ICS^b or not).[2] When $\Pi_{particle}(p)$ is not p-safe (ICS^b), w_{ps} and the function f are set to infinity. However, if $\Pi_{particle}(p)$ is p-safe, w_{ps} is set to 1, to restrict the objective function to the two other constraints, that is, $f = w_d Cost_d(\Pi_{particle}(p)) + w_t Cost_t(\Pi_{particle}(p))$.

In cases other than braking ICS, w_{ps} can be improved by including sensory uncertainty. w_{ps} can be determined by evaluating the probability of p-safety of $\Pi_{particle}$ (the probability that a collision will not occur in the future). As the p-safety of the particle is verified by ICS^b-CHECK,[2] a probabilistic version of this algorithm should be used in such a case. In refs. [85, 86], a probabilistic ICS concept has been proposed. However, the main issue with such a concept is that no strict guarantees of safety are available. Instead, the risks of collision are minimized.[87]

A safe state is a state that is not ICS. Therefore, by duality, a probabilistic ICS-CHECK has to determine the probability that a state s is an ICS. This algorithm should have as input, the state to check (s) and a probabilistic model of the future $\mathcal{W}_p$. Indeed, instead of a conservative model (braking ICS case), a probabilistic model of the future, which can be built by different existing methods,[88–91] should be used. A probabilistic measure is assigned to each object's future trajectory.

As the function f in Eq. (5) should be minimized to solve the optimization problem, a small w_{ps} corresponds to a high safety probability of $\Pi_{particle}$. By duality, it corresponds to a low ICS-ness probability of $\Pi_{particle}$. Therefore,

$$w_{ps} = P_{ICS}(\Pi_{particle}) \tag{9}$$

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

Let $\Pi_{particle} = \{s_0, ..., s_f\}$, w_{ps} can be expressed according to the probability of ICS-ness of each state trajectory s_i (each state corresponds to a given time step and is independent of other time steps);

$$w_{ps} = \prod_{i=0}^{f} P_{ICS}(s_i) \tag{10}$$

$P_{ICS}(s)$ is determined by a probabilistic ICS-CHECK, where the calculation steps are the same as in ICS^b-CHECK.[2] However, instead of determining a binary answer for ICS-ness, a probability of ICS-ness is computed. In ICS^b-CHECK, ICS^b is determined given a set of controls $\tilde{u}_b \in \tilde{\mathcal{U}}_b^S$ and a world model $\mathcal{W}$. s is ICS^b if at least a braking trajectory $\tilde{u}_b$ is collision-free (see ref. [2] for more details about the computing process). With a same reasoning, P_{ICS} is the minimum of collision probabilities for a set of controls $\tilde{u} \in \tilde{\mathcal{U}}^S$, such that:

$$P_{ICS}(s) = \min_{\tilde{u} \in \tilde{\mathcal{U}}^S} \left(P_{ICS}^{(\tilde{u} \in \tilde{\mathcal{U}}^S)}(s) \right) \tag{11}$$

where $P_{ICS}^{(\tilde{u} \in \tilde{\mathcal{U}}^S)}(s)$ can be computed from the probability of collision between $\mathcal{A}(s)$ and $\mathcal{W}_p$ for $\tilde{u}$ (when $\mathcal{A}$ follows $\tilde{u}$) at each time t of this trajectory, that is, $P_{ICS}^{(\tilde{u},t)}(s)$. It is assumed that the probability at each time step is independent of the other time steps. Note that from partial planning perspective, the model of the future should be considered over the time interval $[t_k, t_{k+2} + T]$ (as illustrated in the planning process of Fig. 2), with t_k the initial time step of the planning cycle k and T an appropriate time horizon. In this case:

$$P_{ICS}^{(\tilde{u} \in \tilde{\mathcal{U}}^S)}(s) = \prod_{t=t_k}^{t_{k+2}+T} P_{ICS}^{(\tilde{u},t)}(s) \tag{12}$$

To determine the probability of a collision between $\mathcal{A}(s)$ and $\mathcal{W}_p$ at time t when following $\tilde{u}$ (i.e., $P_{ICS}^{(\tilde{u},t)}(s)$), we must define the subset of points X_t for which $\mathcal{A}$ will be in collision with $\mathcal{W}_p$ at time t when following $\tilde{u}$.

$$X_t = \{q_w / q_w = (x, y) \in Tran_{\tilde{u}}^{-1}(t)[\mathcal{W}_p(t) \ominus \mathcal{A}(s_t)]\} \tag{13}$$

where $\ominus$ denotes Minkowski difference and q_w is the workspace coordinates of $\mathcal{A}(s)$. $Tran_{\tilde{u}}^{-1}$ is the unique geometric transformation describing the motion of $\mathcal{A}$ from s to s_t when following $\tilde{u}$ [with s (state to be checked) the initial state of $\tilde{u}$ and s_t the collision state].

$$P_{ICS}^{(\tilde{u},t)}(s) = \max_{q_w \in X_t} (P(q_w)) \tag{14}$$

This is the probability that a subset of points X_t is occupied by $\mathcal{W}_p$. $\mathcal{W}_p$ is modelled by an independent stochastic process for each object $B_j, j = 1, ..., n_b$ of the environment. $f_p^{B_j}(x, t)$ and $f_p^{B_j}(y, t)$ denote the probability density functions of occupancy of an object B_j in the 2D workspace at time t for a position coordinates x and y. Therefore, the probability of collision of q_w; $P^{B_j}(q_w)$ for B_j can be obtained by integration:

$$P^{B_j}(q_w) = \int_{X_t} f_p^{B_j}(x, t) f_p^{B_j}(y, t) dx dy \tag{15}$$

$f_p^{B_j}$ can be modelled by a normal probability density distribution, where sensors' uncertainty can be added as Gaussian noise.

$$f_p^{B_j}(x, t) = N(\mu, \sigma^2) \tag{16}$$

μ is the mean value of the pdf of x at time t and σ is the corresponding covariance. Finally, the probability of a collision of q_w for all environment objects is

$$P(q_w) = \min_{j=1,...,n_b} P^{B_j}(q_w). \tag{17}$$

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

4.4.2. Distance constraint. The second constraint that should be considered is the minimum distance to the goal. It is expressed by the term $w_d Cost_d(\Pi_{particle}(p))$ in Eq. (5). w_d is a weighting factor that biases the function f towards the distance cost $Cost_d$. This cost represents the Euclidean distance between the goal state s_g and the final state of the trajectory $\Pi_{particle}(p)$; s_f (i.e., the state with the ID corresponding to the last ID in *Traj_IDs*). It has the form:

$$Cost_d(\Pi_{particle}(p)) = ||s_f - s_g|| \tag{18}$$

4.4.3. Time constraint. Besides the minimum distance, when reasoning in the state × time space, the time parameter is very important. Indeed, the third constraint is to consider a minimum time cost of the solution, thanks to the term $w_t Cost_t(\Pi_{particle}(p))$ in the function f (Eq. (5)). w_t is a weighting factor related to the time parameter which biases f towards the time cost $Cost_t$, where $Cost_t$ is the time cost of the trajectory $\Pi_{particle}(p)$ such that:

$$Cost_t(\Pi_{particle}(p)) = \sum_{j=1}^{|Traj\_IDs|-1} \Delta T_{\delta\Pi_j} \tag{19}$$

with $\Delta T_{\delta\Pi_j}$ the time cost of $\delta\Pi_j$, the trajectory primitive linking two successive nodes corresponding to two successive IDs in *Traj_IDs*. The two weighting factors w_s and w_t balance the prominence of distance and time parameters to find the optimal solution.

4.5. Passive safety guarantee

The passive safety of PASSPMP-PSO has been defined and formulated in the previous sections, but it should be proved. Therefore, to guarantee the safety of PASSPMP-PSO, the first addressed issue is the algorithm time cycle δ_{cycle} that should be set carefully because the robot has a limited time to return a decision, otherwise a collision may occur. Consequently, a decision time constraints δ_d must be respected. It depends on the current state of the robot and the surrounding environment. Generally, its value should be selected so that the robot does not take more time than the collision time (i.e., time taken by the robot to collide with an object) to decide what to do and to give the robot enough time to avoid collision. δ_d can be formally defined as follows:

Definition 4 (Decision time). Given the current trajectory of $\mathcal{A}$, δ_d is the time that separates the current time instant of $\mathcal{A}$ from the instant corresponding to the first ICS^b state on this trajectory.

To fulfil this requirement, the time cycle must never exceed the decision time, that is,

$$\delta_{cycle} < \delta_d \tag{20}$$

This condition is necessary for the choice of PASSPMP-PSO time cycle. However, this is insufficient to ensure that the robot always remains passively safe. Indeed, it must be proven that at each planning cycle, a passively safe trajectory (ICS^b-free) is found. To do so, the following property is required:

Property 2 (P-Safety guarantee). If a state s_0 is p-safe, then there exists at least one p-safe state trajectory that drives $\mathcal{A}$ through states that are also p-safe.

Proof. Given Definition 2, if s_0 is p-safe, then there exists at least a braking trajectory $\tilde{u}_b$ (from the state s_0 to s_j) that is collision-free. At the state s_j, $\mathcal{A}$ is at rest. Therefore, s_j is p-safe. By applying Property 1, as $\tilde{u}_b$ is collision-free and s_j is p-safe, then $\tilde{u}_b$ is p-safe and every state of this trajectory is also p-safe. Let $\tilde{u}_b$ a particular case of possible trajectories, it has been proven that there exists a p-safe trajectory.

Given Property 1, Property 2 guarantees that at each planning cycle, a p-safe partial trajectory is available.

Another important element regarding safety issue is the model of the future. To consider all possible future motions of a given moving object with an unknown future behaviour (seen or unseen), the model used herein is conservative: given an upper bound of the velocity of objects, every point of the limit and outside the field-of-view is modelled as a disc that grows as time passes, that is, a cone in space × time (see Fig. 4).[2] Regarding this model, it is necessary to answer the question how far into

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

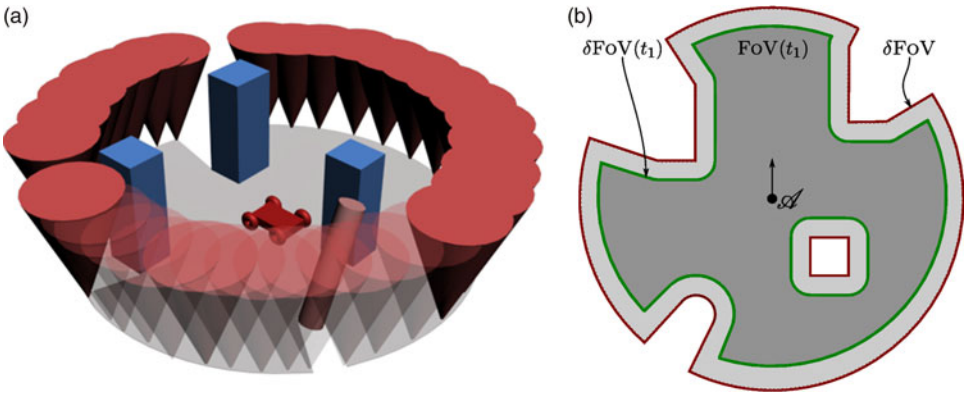

Fig. 4. Model of the future: (a) space × time conservative model (partially represented for visualization purposes) for a scenario with three fixed objects and one moving object. Every point of the limit of the field-of-view and occlusions (unseen objects) is modelled as a disc that grows as time passes (i.e., a cone in space × time). The fixed object remains constant over time (i.e., a vertical line in space × time). The moving objects are modelled according to their future behaviour (i.e., a curve in space × time) if it is available and reliable, otherwise it is treated as an unseen object and modelled as a growing disc. (b) How FoV shrinks as time passes (for a time t_1 greater than the sensing time), with δFoV the limit of the field-of-view and FoV the subset of $\mathcal{WS}$ perceived by $\mathcal{A}$.

the future reasoning is done? This model must still valid over a limited interval of time to guarantee safety, hence, the following property:

Property 3 (**Future model validity**). The model of the future has to remain valid over the time interval $[t_k, t_{k+2} + T_h]$, for every cycle k, with $t_{k+2} = t_k + 2 * \delta_{cycle}$.

Proof. PassPMP-PSO is based on a PMP concept (see Section 4.1) where the future model is updated at each cycle. Let us consider the planning cycle 0. Based on the world model $\mathcal{W}(t_0)$, PassPMP-PSO has a time interval $[t_0, t_1]$ to compute the trajectory Π_1 that will start at time t_1 (corresponding to the state s_1, i.e., the beginning of the next cycle $[t_1, t_2]$). This behaviour is iteratively repeated until $\mathcal{A}$ reaches its goal. From a safety point of view, it is guaranteed that s_1 is p-safe since it is obtained from Π_0 (computed during the previous cycle) which is a p-safe trajectory. Thanks to Property 2, the existence of a p-safe trajectory Π_1 (starting at s_1 and driving $\mathcal{A}$ to the state s_2) is guaranteed. Normally, to ensure that PassPMP-PSO can compute a p-safe trajectory Π_2 at the next cycle ($[t_1, t_2]$), s_2 must be p-safe with respect to $\mathcal{W}(t_0)$. However, s_2 corresponds to the time instant t_2 when $\mathcal{W}(t_0)$ begins at t_0. Assuming that, at time t_2 (when the updated model of the future $\mathcal{W}(t_2)$ is available), s_2 turns out to be a braking ICS with respect to $\mathcal{W}(t_2)$, it means that a collision will occur. That is why, $\mathcal{W}(t_0)$ must be considered until $t_2 + T_h$.

From a braking ICS perspective and given Definition 2, for an arbitrary subset. $\tilde{\mathcal{U}}_b^S$ (from a state s) of the whole set of possible braking trajectories $\tilde{u}_b$ of finite duration t_b, a finite time horizon T_h exists:

$$T_h = \max_{\tilde{u}_b \in \tilde{\mathcal{U}}_b^S} \{t_b\} \tag{21}$$

For details and proof, readers are referred to ref. [2].

4.6. Stability and optimality

Regarding stability of the control system, in a partially unknown environment, stability in the sense of reaching the goal is not possible to guarantee as long as the system has only a partial knowledge of its environment. For example, unknown objects (i.e., not perceived by $\mathcal{A}$) could prevent the system to reach its goal. This problem has already been tackled in ref. [92], where it has been established that in such conditions, stability is impossible to ensure. The primary concern of PassPMP-PSO is to guarantee the safety of the robot's mission rather than completion of the mission, which may be infeasible. Typically, motion safety has to be guaranteed before tackling the stability issue, and so it is a problem of interest in itself.

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

For the same reasons, the optimality issue is challenging in such conditions. Generally, to solve an optimization problem, the solution should be defined by constraints to respect and an objective function to maximize or minimize. The solution is feasible (admissible) if it respects the imposed constraints. Equivalently, the optimality problem of motion planning requests finding a feasible trajectory with a minimum cost.

First of all, the feasibility of PASSPMP-PSO's resulting trajectory can be explicitly proven. Conceptually, PASSPMP-PSO explores the search space by extending a tree in the state × time space of $\mathcal{A}$. The generated trajectories obey the feasibility constraint given the dynamic model of the system (see Eq. (1)). Furthermore, PASSPMP-PSO is based on a passive safety guarantee where generated trajectories are p-safe and every trajectory state is p-safe, that is, not braking ICS (given Properties 1 and 2).

From the definition of a braking ICS (ICS^b) in Eq. (7), $\mathcal{A}(\tilde{s}(s, \tilde{u}_b, t))$ designates the closed subset of the workspace occupied by $\mathcal{A}$ when it is in the state $\tilde{s}$ reached at time t starting from s while following $\tilde{u}_b$ by integrating the differential equation of $\mathcal{A}$'s dynamics (Eq. (1)). States p-safety is then defined while respecting kinodynamic constraints of the system. Therefore, given all these points, PASSPMP-PSO is feasible.

Now that the feasibility is proven, the optimization problem can be addressed. Given the same constraints mentioned above, PASSPMP-PSO computes partial trajectories to the goal so as to deal with real-world situations. In this case, the optimization problem is subdivided into sub-problems where the optimal solution is defined for each sub-problem. Thus, given the planning concept and the harsh constraints imposed by the environment and the robot, a near-optimal partial trajectory is computed at each PASSPMP-PSO planning cycle. It is defined as follows:

Definition 5 (**Near-optimality**)**.** Given a PMP problem defined with an initial s_k, a goal state s_g, a planning time cycle δ_d and a cost function $f : \Sigma \rightarrow \mathbb{R}_{\geq 0}$, a trajectory Π^* is near-optimal if Π^* is feasible and $\Pi^* = \text{argmin} f(\Pi_n), \forall \Pi_n \in \Sigma$.

With Π_n a feasible partial trajectory and Σ the set of all feasible partial trajectories generated during the planning cycle. Based on the objective function in Eq. (5), the near-optimal solution corresponds to the global best particle, that is, $\Pi^* = Gbest$. *Gbest* is the global minimizer of the objective function. It is the best particle among all particles (trajectories), that is, $f(Gbest) = \{f(\Pi_{particle}(p)), p = 1, ..., N_p\}$. Therefore, Π^* is a near-optimal solution during δ_c time cycle.

4.7. PASSPMP-PSO *algorithm*

Let us first summarize the addressed parts about the design concept of PASSPMP-PSO. Conceptually, the algorithm has been designed primitively, with a natural reasoning behaviour approach. For example, if a human is in a dynamic environment and has to reach a target that is outside his field-of-view, he will naturally plan an optimal partial trajectory in the limit of his knowledge and will modify his trajectory continuously given the new update from the environment. Similarly, given the model of the environment, PASSPMP-PSO computes an optimal partial trajectory in the limit of its knowledge (related to T_h, the time horizon). This trajectory can be completely or partially executed when a new one is available. However, to handle real-time constraints, planning and execution are interleaved, that is, when executing a trajectory, a new one is planned to be executed in the next cycle. The search space is explored using a tree-based method. It is based on a state × time space exploration, where each node in the tree is extended given a set of feasible controls, starting with the root node, and the process is repeated for new extended nodes until the whole tree is constructed. From the set of tree nodes, trajectories are encoded into particles, thanks to the encoding technique presented in Section 4.3. The optimal trajectory corresponds to the global best *Gbest*. It is determined using the objective function of Eq. (5) which is based on the time/distance costs and the safety guarantee condition. Indeed, to guarantee safety, the natural behaviour when encountering a danger (imminent collision) is to brake and stop. That is, in cases where collision is inevitable, the system has to brake and stop to remain safe (given Definition 2). Therefore, given the PMP process of PASSPMP-PSO (more details are in ref. [9]), at each planning cycle, a near-optimal passively safe partial trajectory is computed. The set of partial trajectories computed during the whole process drives the robot to its goal s_g.

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

Algorithm 2 PASSPMP-PSO process for planning cycle k.

Input: goal s_g, model of the future $\mathcal{W}(t_k, t_{k+2} + T_h)$, cycle duration δ_{cycle}, partial trajectory Π_k to execute, number of particles $N_p - 1$.
Output: Π_{k+1} optimal p-safe partial trajectory to execute in next cycle $k + 1$.

1: TREE=GROW_TREE($\mathcal{W}(t_k, t_{k+2} + T_h)$, s_{k+1}, δ_{cycle}); //state × time space exploration
2: Π_{k+1}= TRAJ_OPTIMIZATION (TREE,s_g, $N_p - 1$); //compute an optimal p-safe partial trajectory to apply in cycle $k + 1$
3:
4: **Procedure** GROW_TREE($\mathcal{W}(t_k, t_{k+2} + T_h)$, s_{k+1}, δ_{cycle})
5: Tree initialization: $\eta_{root} = s_{k+1}$ (initial state of $k + 1$ cycle, provided by Π_k), TREE $= \{\eta_{root}\}$, $N_1 = \{\eta_{root}\}$ (set of nodes η_i belonging to actual depth, $i = 1, ..., |N_1|$), $d_{tree} = 0$ (current tree depth);
6: set $t_{exploration} < \delta_{cycle}$;
7: **while** $t < t_{exploration}$ **do**
8: $N_2 = \emptyset$;
9: **for** i=1 to $|N_1|$ **do**
10: **Forall** $u_j \in U_{ctrl}$ **do** //each node η_i is expanded given a set of control U_{ctrl}
11: $(\eta_{new}, \delta\Pi_{new}) \leftarrow$ SAFE_EXPANSION(η_i, u_j,$\mathcal{W}$,TREE);
12: //save ID of η_{new} as child node ID in children IDs set of the parent η_i
13: *ChildID_SET*(η_i).push_back(ID(η_{new}));
14: N_2.push_back(η_{new});
15: **end**
16: **if** $\eta_{new} = \emptyset$ **then**
17: $(\eta_{new}, \delta\Pi_{new}) \leftarrow$ GENERATE_BRAKING_TRAJ(η_i);
18: *ChildID_SET*(η_i).push_back(ID(η_{new}); //save ID of η_{new} as a child node ID of the parent node η_i
19: **end if**
20: **end for**
21: $N_1 \leftarrow N_2$;
22: $d_{tree} = d_{tree} + 1$;
23: **end while**
24: **return** TREE
25: **EndProcedure**
26:
27: **Procedure** SAFE_EXPANSION(η_i,u_j,$\mathcal{W}$, TREE)
28: // generate a trajectory primitive
29: (η_{new}, $\delta\Pi_{new}$) $\leftarrow$ GENERATE_PRIM(η_i, u_j); //$\delta\Pi_{new}$ is delimited by the two nodes η_i and η_{new}
30: //passive safety check based on ICS^b-CHECK algorithm[2]
31: **if** BRAKING_ICS_CHECK(η_{new},$\mathcal{W}$) = False **then**
32: //η_{new} is p-safe
33: **if** $\delta\Pi_{new}$ is collision-free **then**
34: TREE = TREE $\cup$ η_{new};
35: TREE = TREE $\cup$ $\delta\Pi_{new}$;
36: **return** TREE
37: **end if**
38: **end if**
39: **EndProcedure**

Algorithm 2 illustrates the overall process of PASSPMP-PSO for one planning cycle (cycle k) and Algorithms 1, 3 and 4 represent sub-algorithms corresponding to the functions of Algorithm 2. During this cycle, the partial trajectory Π_{k+1} (the output of Algorithm 2) is planned towards the goal s_g based on the conservative model of the future $\mathcal{W}$ dealing with unseen objects (occlusions and perception limits) and their unknown future behaviour. This model is valid during $2\delta_{cycle} + T_h$ (given Property 3), that is, it remains valid so as to guarantee p-safety of Π_{k+1} until s_{k+2} (its final state).

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

Algorithm 3 Trajectory optimization.

Input: expanded tree TREE, goal s_g, number of particles $N_p - 1$.
Output: best particle Π_{k+1}.

1: //Particles initialization (optimization iteration $\tau = 0$)
2: **for** each particle p=0 to $N_p - 1$ **do**
3: $(X_\tau(p), V_\tau(p)) \leftarrow$ INITIALIZE_SWARM(N_{tree}); //Algorithm 4
4: $Pbest_\tau(p) = X_\tau(p)$;
5: $\Pi_{particle}(p) \leftarrow$ CONSTRUCT_TRAJ($X_\tau(p)$); //state trajectory constructed using Algorithm 1
6: compute $f_\tau(p)$; // the objective function according to Eq. (5)
7: **end for**
8: $Gbest_{index} = \underset{p=1,\ldots,N_p}{\text{argmin}} f_\tau(p)$; //index of the particle corresponding to the global best
9: $Gbest_\tau = X_\tau(Gbest_{index})$;
10: **while** $\tau < iteration_{max}$ **do**
11: **for** each particle p=0 to $N_p - 1$ **do**
12: $V_{\tau+1}(p) = V_\tau(p) + w_c|rand_1|(Pbest_\tau(p) - X_\tau(p)) + w_s|rand_2|(Gbest_\tau - X_\tau(p))$;
13: **if** $|V_{\tau+1}(p)| > V_{pmax}$ **then**
14: $V_{\tau+1}(p) = V_{pmax}$;
15: **end if**
16: $X_{\tau+1}(p) = X_\tau(p) + V_{\tau+1}(p)$;
17: //construct trajectory
18: $\Pi_{particle}(p) \leftarrow$ CONSTRUCT_TRAJ($X_{\tau+1}(p)$); //state trajectory constructed according to Algorithm 1
19: //compute the objective function
20: compute $f_{\tau+1}(p)$; //according to Eq. (5)
21: **if** $f_{\tau+1}(p) < f_\tau(p)$ **then**
22: $Pbest_{\tau+1}(p) = X_{\tau+1}(p)$; //update personal best
23: **end if**
24: **end for**
25: //update global best
26: **if** $\min_{p=1,\ldots,N_p} f_{\tau+1}(p) < \min_{p=1,\ldots,N_p} f_\tau(p)$ **then**
27: $Gbest_{index} = \underset{p=1,\ldots,N_p}{\text{argmin}} f_{\tau+1}(p)$;
28: $Gbest_{\tau+1} = X_{\tau+1}(Gbest_{index})$;
29: **end if**
30: **end while**
31: $\Pi_{k+1} = \Pi_{particle}(Gbest_{index})$;
32: **return** Π_{k+1}

Algorithm 4 Particles initialization.

Input: N_{tree}.
Output: X, V; position and velocity vectors.

1: **for** n=0 to N_{tree} **do**
2: $X(p)[n]$=RANDOM_POSITION();
3: $V(p)[n]$=RANDOM_VELOCITY();
4: **end for**
5: **return** X
6: **return** V

In parallel to planning, Π_k the trajectory computed in the previous cycle is executed. The goal s_g, the world model $\mathcal{W}$ and the time cycle δ_{cycle} are all inputs of Algorithm 2 to plan the trajectory Π_{k+1}.

Associated with Algorithm 2, the schematic diagram of Fig. 5 simply introduces a general overview of the different parts of the planning process and how they interact. PASSPMP-PSO algorithm contains mainly two steps: (1) the state × time space exploration (GROW_TREE function) and

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

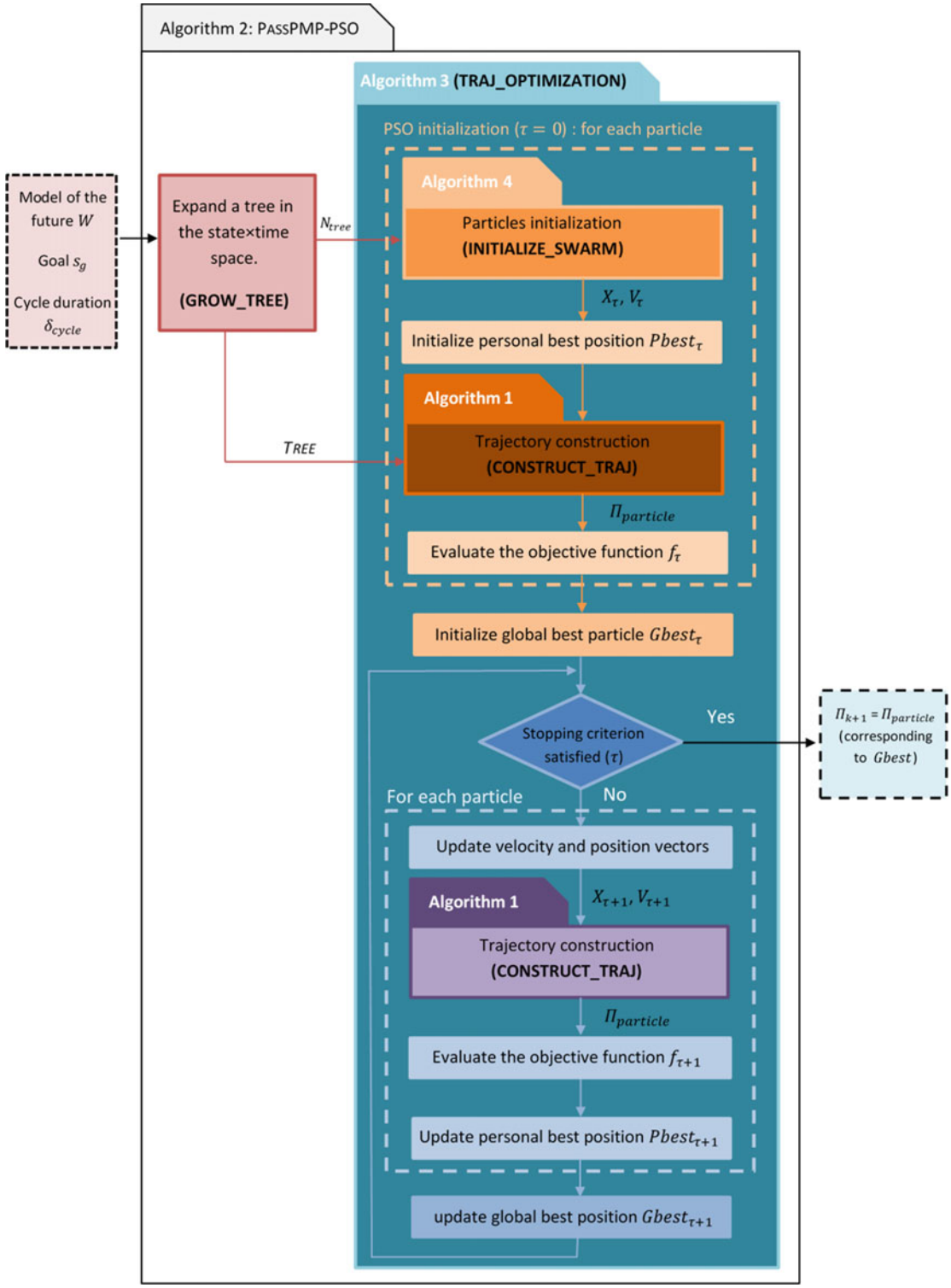

Fig. 5. The schematic diagram of PASSPMP-PSO process of Algorithm 2.

(2) computing an optimal p-safe partial trajectory Π_{k+1} (TRAJ_OPTIMIZATION function, given by Algorithm 3). The latter step requires Algorithm 4 for the particles initialization and Algorithm 1 for the trajectories construction (as illustrated in Fig. 5).

The first step of Algorithm 2 is therefore performed by expanding a tree in the state × time space of $\mathcal{A}$ (GROW_TREE function, in Algorithm 2). For each node η_i (parent node), all child nodes (states) are tested for p-safety and then p-safe nodes are retained as descendants of this node. Their

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

corresponding IDs are, respectively, saved in the *ChildID_SET* to be used in the next step. For a given node η_i, there exist situations where no p-safe primitive exists. A collision-free braking trajectory is therefore generated to guarantee passive safety (given Definition 2). In this case, η_i has a unique child whose ID is also saved.

The second step of the PASSPMP-PSO algorithm is to compute a near-optimal p-safe partial trajectory Π_{k+1} to apply in cycle $k+1$ (using the TRAJ_OPTIMIZATION function, in Algorithm 2 and detailed in Algorithm 3). This is accomplished, thanks to an iterative optimization process based on PSO. During the first iteration ($\tau = 0$), the swarm is randomly initialized by the INITIALIZE_SWARM function, in Algorithm 4, where each particle is represented by an N_{tree} dimensional position vector $X_\tau(p)$ (see Fig. 3). Based on the X_τ vector, for each particle p, the personal best vector $Pbest_\tau$ is initialized and a state trajectory is constructed using Algorithm 1. The global best $Gbest_\tau$ is then computed based on the objective function f_τ of each particle. The velocity and position vectors are then updated given $Pbest_\tau$ and $Gbest_\tau$. As for the first process iteration, a state trajectory $\Pi_{particle}$ is constructed (for each particle) based on the new position vector $X_{\tau+1}$ according to Algorithm 1. The corresponding objective function $f_{\tau+1}$ is computed according to Eq. (5). Consequently, the personal best $Pbest_{\tau+1}$ and the global best $Gbest_{\tau+1}$ are computed. This process is repeated for a maximum number of iterations ($iteration_{max}$). At the end of this process, the constructed trajectory $\Pi_{particle}$ corresponding to the global best (the best solution found so far) is saved as Π_{k+1}.

5. Results

In this paper, a new motion planner called PASSPMP-PSO has been proposed. Unlike existing state-of-the-art approaches, it solves both motion safety and trajectory optimization issues, thanks to a new tree encoding technique based on PSO. The particularity of this method is to extend feasible trajectories (instead of paths for the other encoding techniques) in the state $\times$ time space, that is, to consider the kinodynamic constraints of the system and the time parameter. In addition, the solution is defined, thanks to an objective function which, unlike the other methods, depends on a safety criterion and minimum trajectory costs (*wrt* time and distance). Safety is formally guaranteed, it is based on a passive motion safety concept. To better deal with real-world constraints, PASSPMP-PSO interleaves planning and execution processes. At each cycle, a near-optimal and passively safe partial trajectory is generated. It is guaranteed that at each cycle, a p-safe trajectory is always found. PASSPMP-PSO algorithm has been implemented in simulation and in real-world experiments, to illustrate its performances, respectively, in challenging scenarios and real situations. Furthermore, it has been evaluated *wrt* two other motion planning methods.

5.1. Simulation results

5.1.1. Test conditions. To validate PASSPMP-PSO and demonstrate its performances, it has been implemented and tested in simulation on scenarios similar to that of Fig. 1. The simulation environment is a 2D workspace of 180×180 m, featuring fixed and moving objects with arbitrary trajectories[2] and an upper bounded velocity $v_{b_{max}} = 20$ m/s. The used robotic system $\mathcal{A}$ is a car-like vehicle, governed by the following dynamics:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ v\tan\xi/L \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u_\alpha + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_\xi \tag{22}$$

where $\mathcal{A}$'s state is a 5-tuple $s = (x, y, \theta, v, \xi)$, (x, y) the Cartesian coordinates, θ the orientation, v the linear velocity and ξ the steering angle. (u_α, u_ξ) is the control couple, with u_α the linear acceleration and u_ξ the steering angle velocity. L denotes $\mathcal{A}$'s wheelbase.

The system constraints are : $|v| \leq v_{\max}$, $|\xi| \leq \xi_{\max}$, $|u_\alpha| \leq u_{\alpha_{\max}}$ and $|u_\xi| \leq u_{\xi_{\max}}$, with, $v_{\max} = 20$ m/s, $\xi_{\max} = 0.314$ rad, $u_{\alpha_{\max}} = 7$ m/s^2 and $u_{\xi_{\max}} = 0.314$ rad/s.

[2]Note that objects trajectories are generated using B-spline method.

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

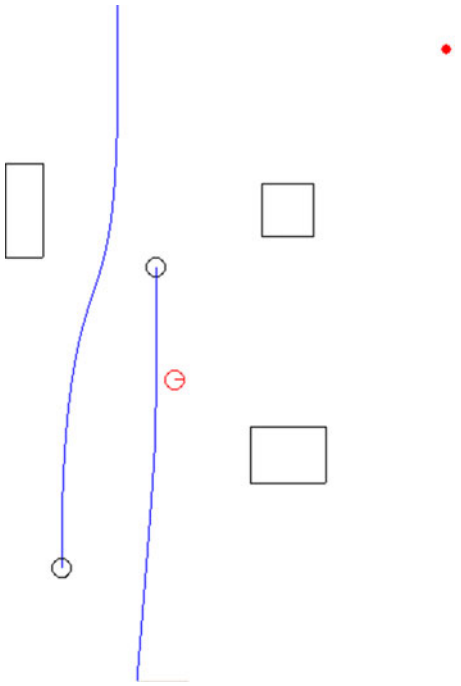

Fig. 6. Test scenario: the environment features three fixed objects (polygons) and two moving objects (black discs) (with their future trajectories). The robot A is the disc at the centre and the red point is the goal state.

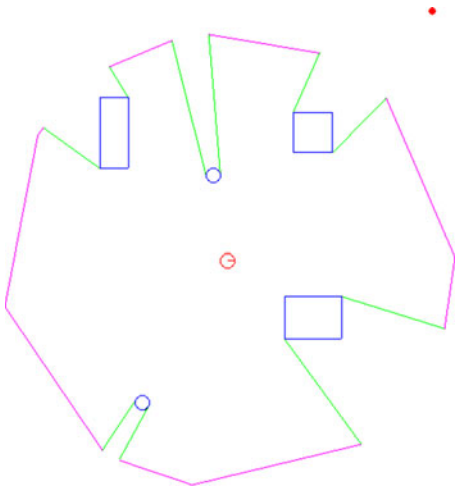

Fig. 7. Field-of-view for the scenario of Fig. 6. The circular arc corresponding to the maximum range of the range finder has been replaced by straight segments; this conservative simplification could easily be lifted. The limits of the field-of-view are represented by the magenta segments and occlusions are represented by the green segments.

$\mathcal{A}$ is equipped with an omnidirectional laser range finder placed in the centre of $\mathcal{A}$ (i.e., $\mathcal{A}$ has a 360° field-of-view), with a maximum range of 80 m. A conservative model of the future is considered in PASSPMP-PSO dealing with field-of-view limits, occlusions and the unknown future behaviour of moving objects.

The implementation of different developed algorithms has been done in C++ on a laptop computer Intel Core i7 1.6GHz CPU, 4GB RAM, ATI Mobility Radeon HD 4500 GPU.

5.1.2. PASSPMP-PSO *at work.* PASSPMP-PSO has been tested in two different scenarios: a simple environment and a crowded environment. To understand how PASSPMP-PSO works, it is illustrated at first for a simple scenario featuring two moving objects and three fixed objects. The 2D workspace and $\mathcal{A}$'s field-of-view are, respectively, represented in Figs. 6 and 7. The size of the swarm

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

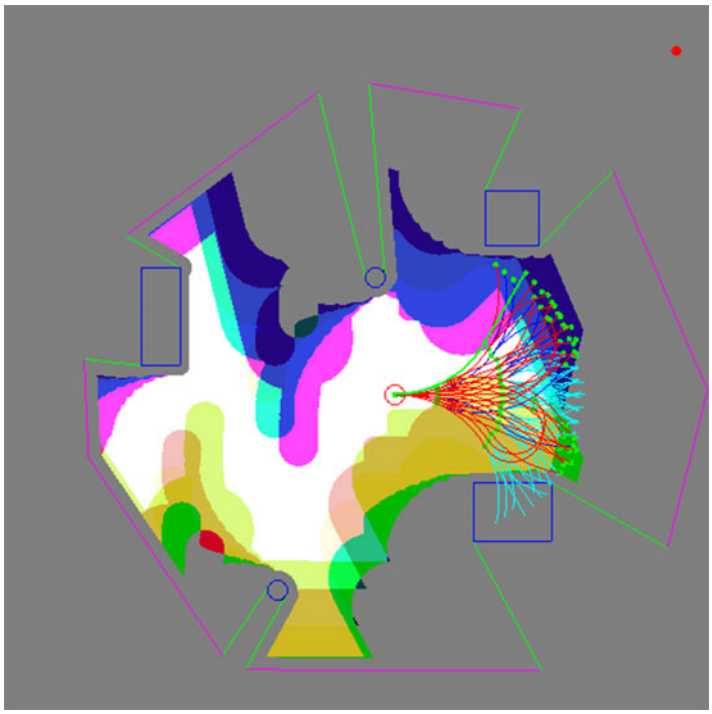

Fig. 8. PASSPMP-PSO for a given planning cycle. The robot is the red disc at the centre. The environment corresponds to the scenario of Fig. 6. The red point is the goal state. The grey regions illustrate the braking ICS states (computed by ICS^b-CHECK^2). The green points are all p-safe tree nodes. Particles are represented by the red trajectories (corresponding to the final iteration of the optimization process). The optimal solution is the green trajectory. For more clarity, the trajectory primitives represented in cyan are not p-safe and the blues are other possible trajectories (not encoded as particles).

is 100 particles. The acceleration coefficients w_c and w_s in Eq. (2) are set to 2 [$w_c = w_s = 2$][10,93] and the weighting factors w_t and w_d in the objective function of Eq. (5) are, respectively, set to 0.8 and 0.2 [$w_t = 0.8$ and $w_d = 0.2$].

Figure 8 illustrates PASSPMP-PSO behaviour during one planning cycle. After the expansion of the tree in the search space, partial trajectories are constructed from the set of tree nodes based on the encoding algorithm (Algorithm 1). Finally, each particle represents a partial trajectory (trajectories in red). Given the objective function, the optimization problem is solved based on safety and trajectory cost (*wrt* time and distance) constraints. To verify the passive safety, ICS^b-CHECK^2 is used. Based on the current model of the future, the grey areas in Fig. 8 represent ICS^b states while the remaining areas of the space are p-safe. Note that for friendly printing, ICS^b regions are represented in grey instead of black colour representing the intersection of all ICS^b sets as stated in ICS^b-CHECK algorithm.[2] A first conclusion is that all particles are p-safe as they are out of the ICS^b regions. By adding the cost constraints, the optimal solution can finally be determined. It is the particle represented in green (the global best *Gbest*). Therefore, unlike existing trajectory optimization approaches, the resulting solution in our case fulfils both safety and optimality conditions.

Figure 9 illustrates the optimization process through different iterations. At each iteration, the global best (the green trajectory) is determined from a set of possible partial trajectories (particles). The particles are different over iterations, as at each iteration, a trajectory is constructed given an updated particle's priority vector.

Figure 10 illustrates the planning process for different PASSPMP-PSO cycles, with $\delta_{cycle} = 2.4$ s. At each cycle, the world model is updated and accordingly ICS^b-free regions are recomputed. Based on these regions, a p-safe optimal partial trajectory is computed to be executed in the next cycle [i.e., during a cycle k and given the world model $\mathcal{W}(t_k)$ (corresponding to the state s_k), the trajectory Π_{k+1} is computed (starting at the state s_{k+1})]. Planning and execution are interleaved; when a trajectory is executed, the trajectory to be executed in the next cycle is computed and the process is repeated until the robot reaches its goal. Through its different cycles, PASSPMP-PSO drives the robot to its goal

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

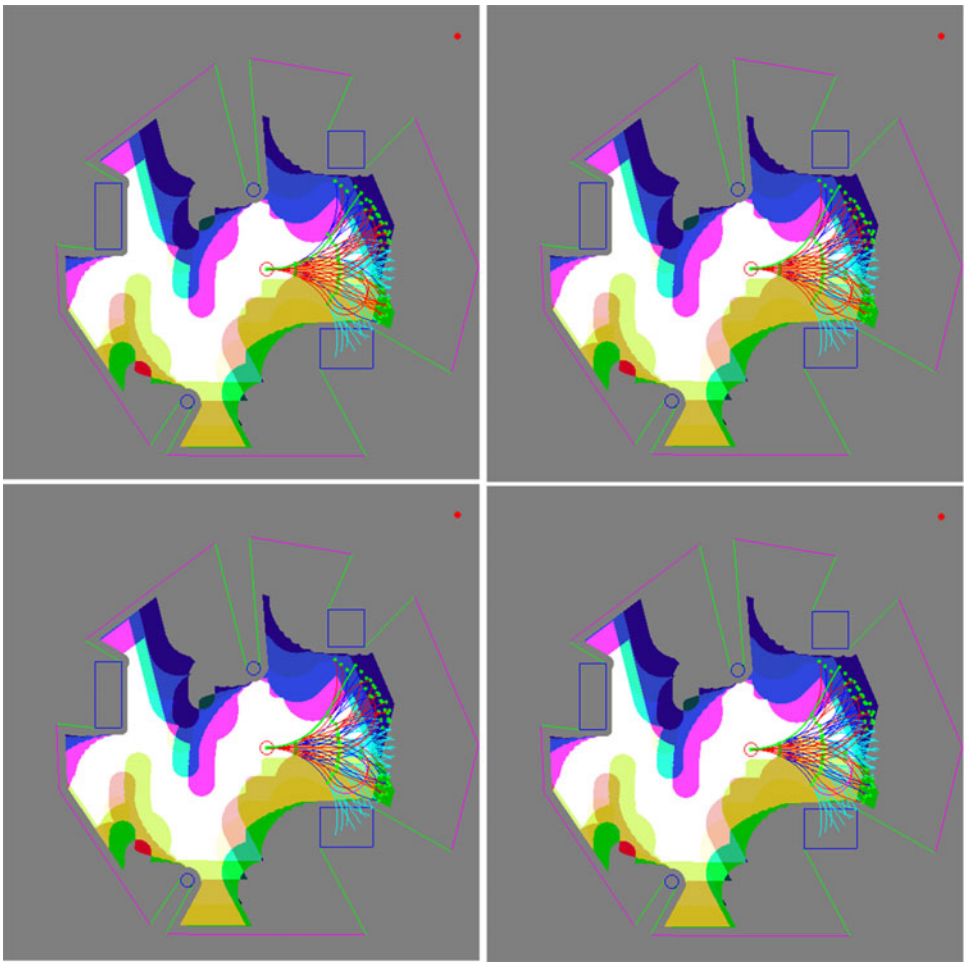

Fig. 9. Snapshots of PASSPMP-PSO for different iterations of the optimization process. The optimal solution is represented in the last snapshot (on the bottom right), that is, the global best found so far (the solution is the green trajectory).

by generating optimal partial trajectories while guaranteeing passive safety. When most planning approaches miss reactivity to environment changes, especially with regard to its future evolution, PASSPMP-PSO with its PMP process, guarantees a regular update of the environment future model within a valid lookahead. From a passive safety perspective, this last parameter depends on the maximum braking time (the ratio between the linear speed of the robot and its linear acceleration), given Eq. (21). Furthermore, a special feature of this model lies in considering the future evolution of both perceived objects and unexpected objects (representing imminent collision risk from occluded regions and field-of-view limits). Therefore, the robot avoids these moving objects or brakes to stop when avoidance is impossible. To remain p-safe, the robot is always moving out of the grey area (i.e., braking ICS states) in Fig. 10.

To evaluate PASSPMP-PSO performances, it has been tested in a more complex scenario (see Fig. 11), a crowded environment with 22 objects moving arbitrarily and at high speed. Figure 12 illustrates the PASSPMP-PSO process interleaving planning and execution, many snapshots corresponding to different cycles are represented, with $\delta_{cycle} = 1.4\,\text{s}$. At each cycle, a partial trajectory is computed while another is executed. The executed trajectory is the thick mark behind the robot (computed in the previous cycle). The planned trajectory is the trajectory ahead of the robot (to be executed in next cycle). The whole computed partial trajectories drive the robot to its goal.

5.1.3. PASSPMP-PSO *performances.* The purpose of this section is to evaluate the performances of PASSPMP-PSO *wrt* some important algorithm parameters, namely the w_t and w_d weighting factors, the objects velocity, the time cycle and the swarm size. The experiments have been carried out in the

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

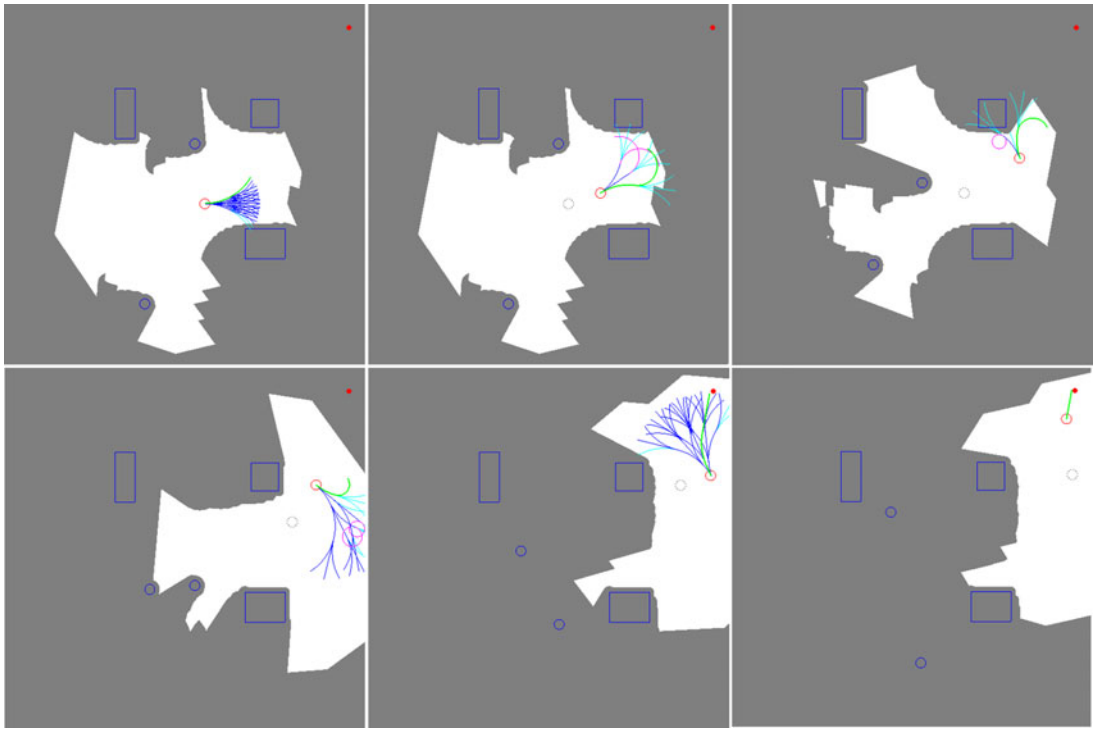

Fig. 10. PASSPMP-PSO planning cycles. During the current cycle (corresponding to a given snapshot) and given the current world model/ICS^b region, the next trajectory is planned (for more clarity, the robot state corresponding to the first state of the current cycle is represented by the grey dashed disc, depicting also the first state of the previously planned trajectory). The optimal partial trajectory is green. Trajectory primitives in blue are p-safe while those in cyan are ICS^b. The magenta primitives are collision-free braking trajectories generated to guarantee passive safety (a case where no p-safe primitive exists for a given node). The grey regions of the space are the ICS^b states to be avoided by the robot. Note that the first snapshot corresponds to the first plan computed while the robot is at rest (as no plan is *a priori* available at the beginning of the process).

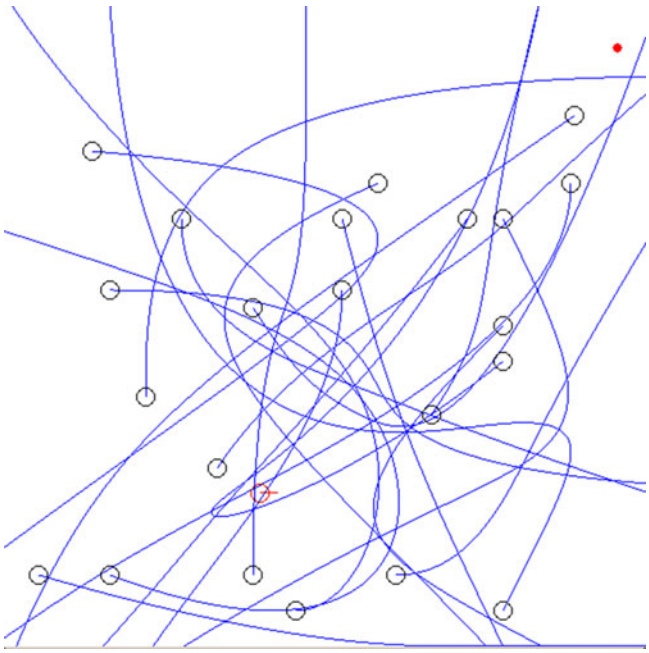

Fig. 11. A crowded environment with 22 objects moving arbitrarily.

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

Fig. 12. PASSPMP-PSO at work in a crowded environment: sequences of $\mathcal{A}$'s displacement at different instants of the process. The red mark left behind the robot is the executed trajectory. The trajectory ahead the robot is the planned trajectory (to be executed). The marks in cyan behind the objects represent their displacement and the arrows depict the directions of their trajectories, where each object is assigned an arrow colour (for a better visualization of their position with time).

scenario of Fig. 12 and for the same experimental conditions as those defined in Section 5.1.1 ($\mathcal{A}$'s kinodynamics constraints, $\mathcal{A}$'s field-of-view, environment constraints, etc.).

Weighting factors. One contribution of PASSPMP-PSO is to find an optimal solution, which is based on the objective function of Eq. (5). The two parameters w_d and w_t of this function (distance and time weighting factors) can significantly affect PASSPMP-PSO's results. To analyse the performance of the approach according to different combinations of these two weighting factors, from 100 runs of the algorithm, some illustrative results are depicted in Tables III and IV. Let us first recall that PASSPMP-PSO is based on a tree exhaustive expansion, that is, the tree is expanded according to a set of controls defined by a constant maximum linear acceleration $u_\alpha = u_{\alpha_{max}}$ and a constant steering angle $|u_\xi| <= u_{\xi max}$, namely the control couples: $(u_\alpha, -u_\xi)$, $(u_\alpha, 0)$, (u_α, u_ξ). This tree is expanded in a passively safe manner (given Algorithm 2), where, in case of no p-safe node is available, a braking trajectory is generated. Therefore, each generated partial trajectory (particle) can be

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

Table III. Performance of PASSPMP-PSO according to different combinations of the two weighting factors w_d and w_t for a given goal state (tests carried out in the scenario of Fig. 11). f, $cost_d$ and $cost_t$ correspond to the best particle. The grey shaded values represent the results corresponding to the best combination (w_d, w_t).

w_d	0	0.1	0.2	0.3	0.5	1
w_t	1	0.9	0.8	0.7	0.5	0
$cost_d(m)$	39.9	29.4	29.4	24.9	24.9	24.9
$cost_t(s)$	4.8	4.8	4.8	7.2	7.2	7.2
f	4.8	9.7	9.7	12.5	12.5	24.9

Table IV. Performance of PASSPMP-PSO according to different combinations of the two weighting factors w_d and w_t for different goal states (tests carried out in the scenario of Fig. 11). For simplicity, only the best (w_d, w_t) combination is represented for each goal.

w_d	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
w_t	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2
$cost_d(m)$	131.1	29.4	12.5	11.9	30.9	12.2	32.4	11.2
$cost_t(s)$	5.3	4.8	5.3	5.3	5.3	4.8	5.3	5.3
f	17.9	9.7	7.4	7.9	18.1	9.2	24.2	10
$Diff_d^+(m)$	17	4.5	3.4	2.2	1.6	1.1	0.7	0.4
$Diff_t^-(s)$	1.9	2.4	1.9	1.9	1.9	2.4	1.9	1.9

the concatenation of a set of primitives (accelerating, decelerating or with constant velocity) and a braking trajectory (in case the robot has to stop to remain p-safe).

Table III depicts the performance of PASSPMP-PSO according to w_d and w_t for a given goal state, where the results for each (w_d, w_t) combination correspond to the best particle. Given Algorithm 3, the best solution is selected from a set of particles (partial trajectories) verifying p-safety criterion, which can be of different durations (cumulated time $cost_t$ in Eq. (19)) and of different Euclidean distances (distance separating the final state of the partial trajectory and the goal state; $cost_d$ in Eq. (18)). Some trajectories can be optimal *wrt* time, that is, the corresponding cumulative time $cost_t$ is the minimum among possible trajectories, but in return, the distance $cost_d$ is very large. This is the case where the function f is minimized only *wrt* time (i.e., $w_d = 0$, then $f = cost_t$) (case of the first column of Table III). Other trajectories are optimal *wrt* distance ($cost_d$ is the minimum), while the duration of the trajectory is large (with regard to the durations of the other trajectories). In this case, the function f is minimized only *wrt* distance (i.e., $w_t = 0$, then $f = cost_d$) (case of the last column of Table III). Unlike these two cases, our purpose is to minimize the function f *wrt* both time and distance parameters. Therefore, the partial trajectory should be selected by finding a compromise between these two parameters. It is clear from Table III that, for values of w_t less than 0.8 (respectively $w_d > 0.2$), the trajectory is selected by minimizing the distance (i.e., $cost_d$ is minimum). However, when $w_t \geq 0.8$ (respectively $w_d \leq 0.2$), both time and distance parameters are minimized. $cost_d$ corresponding to the selected trajectory is greater than the minimum $cost_d$ with a difference of 4.5 m, but the duration of the trajectory gains 2.4 s less. Moreover, the distance travelled during this time (2.4 s) is generally greater than the lost distance (4.5 m). If, for example, the robot is moving with 5 m/s speed, the distance travelled during this time (2.4 s) is 12 m, which is much greater than 4.5 m.

The same tests as those of Table III have been carried out but for different goal states. The corresponding results are depicted in Table IV; for each goal state, only the results corresponding to the best combination (w_d, w_t) are represented (the column in grey in Table III is represented for each case). For each combination (w_d, w_t), the objective function f, the distance to the goal $cost_d$ and the trajectory's time cost $cost_t$ are computed (as for Table III). We also compute the distance difference between the selected trajectory and the trajectory with the minimum distance $cost_d$, denoted $Diff_d^+$, as well as the time difference between the two trajectories, denoted $Diff_t^-$. The main remark deduced from Table IV is that the larger the w_d factor (or the smaller w_t), the smaller the distance difference $Diff_d^+$. This makes sense because the larger the w_d, the more the distance parameter is favoured. Moreover, when comparing the obtained results for low and high values of w_t, it may seem that it is better to use low values of w_t (relatively to $Diff_d^+$ and $Diff_t^-$ values). However, if, for example, a time

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

Table V. Effect of the objects maximum speed on the resulting p-safe states set.

$v_{b_{max}}$ (m/s)	5	10	20	30	40	50
P-safe states (%)	35.7	34.2	31.8	21	16	11.8

Table VI. Evaluation of the duration of the generated trajectory and tree expansion for different values of PASSPMP-PSO's time cycle (for scenario of Fig. 11).

$\delta_c(s)$	0.3	1.5	2.5	3.5	4.5	5.5
n_{nodes}	1	13	40	135	176	237
$\delta_e(s)$	0	1.76	2.64	3.52	7.12	8

factor $w_t = 0.2$ is used (resp. $w_d = 0.8$), in the case of the test of Table III, only the distance will be minimized and not both parameters (time and distance). That is why, it is more convenient to use high values of w_t. When $w_t = 0.9$, $Diff_d^+$ is relatively large, but in the case where $w_t = 0.8$, the value of $Diff_d^+$ is acceptable relative to time earned ($Diff_t^-$). However, an important note is that in the case where $w_t = 0.9$, the selected partial trajectory never ends with a braking trajectory (except of course in the case where all trajectories end with a braking trajectory). Therefore, if in a particular case, it is not desirable to favour braking trajectories, it is sufficient to set the w_t factor to this value. These results justify our choice for parameters $w_t = 0.8$ and $w_d = 0.2$.

Influence of velocity. Given the conservative nature of the model of the future, this model allows to take into account all the possible future movements of a given object with an arbitrary dynamics while considering only its maximum speed. The maximum speed of moving objects is therefore an important parameter that directly affects the computation of braking ICS states (respectively, p-safe states set). To do so, the ICS^b set is computed for different values of the maximum speed of moving obstacles. The results obtained are represented in Table V, where the percentage of p-safe states (around the robot) is given according to the different values of obstacles' maximum speed $v_{b_{max}}$ (noting that the whole region outside the field-of-view limits is unsafe). From the table, it is clear that the higher the speed, the more the p-safe region shrinks (the number of p-safe states decreases). Since each point of the field-of-view limits and occlusions is considered as a potential object with an unknown trajectory, the future behaviour of each of these points is modelled by a disc that increases proportionally with time, with a growth ratio corresponding to the object's maximum speed (a cone in the space × time). Therefore, when the maximum speed of objects varies, the growth ratio of discs varies too. Accordingly, when the speed $v_{b_{max}}$ increases, the space occupied by the discs also increases (the free space decreases). It can be concluded that the set of p-safe states differs according to the nature of objects being present in the environment. For example, if moving obstacles are pedestrians with 5 m/s maximum speed (a pedestrian running), the p-safe states region is relatively large (the case of the first column in Table V), while, for a vehicle with 50 m/s maximum speed, the p-safe region decreases considerably (the case of the last value in Table V). This value is set according to the application and the robotic system constraints.

Influence of the time cycle. PASSPMP-PSO generates a p-safe partial trajectory towards the goal at each δ_{cycle} time cycle. In order to understand the algorithm's behaviour, the duration δ_e of this trajectory as well as the number of nodes generated during the cycle are estimated for different values of δ_{cycle}. Table VI shows that the duration of the trajectory increases *wrt* δ_{cycle}. The longer the time cycle, the longer the planner has time to explore the search space. As a result, the generated trajectory is larger and closer to the goal. However, it is worth noting that when the cycle's duration is very small, it may be insufficient to calculate a p-safe trajectory (the case of the first δ_c value in Table VI). In this case, the duration of the safety check process (computation of ICS^b) is greater than the duration of the cycle. Recalling that the safety check process is based on the ICS^b-CHECK algorithm.[2] Therefore, PASSPMP-PSO performance depends on the ICS^b-CHECK time complexity. It is also illustrated in Table VI that the number of nodes n_{nodes} increases with δ_{cycle}, which is obvious from the moment that the tree grows over time and its expansion is limited by the cycle duration (see Algorithm 2, GROW_TREE function). The number of nodes n_{nodes} grows exponentially, but for the last two values of the table, it is not the case, where the number of nodes generated should

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

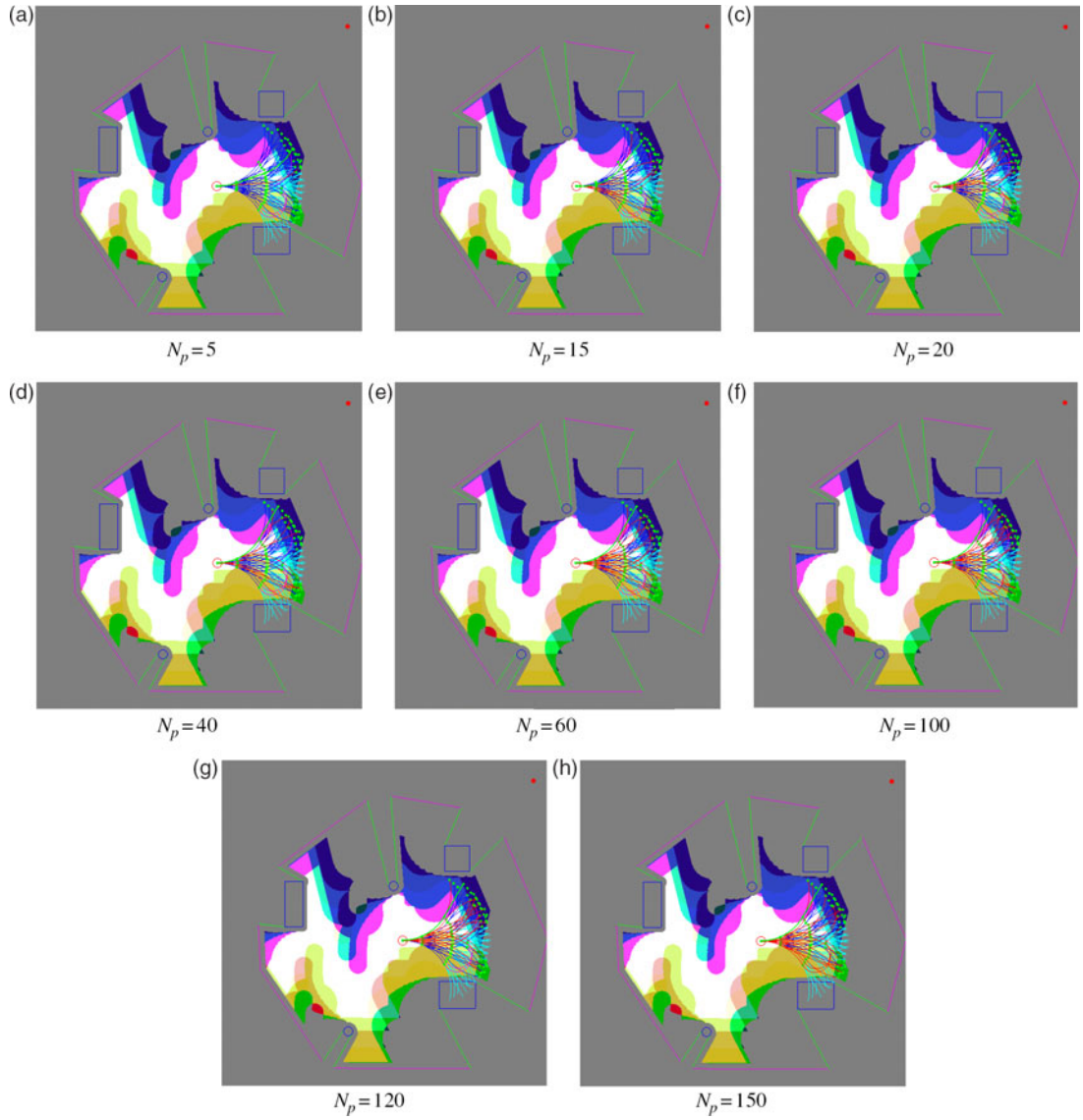

Fig. 13. Evaluation of PASSPMP-PSO for different swarm sizes N_p. Particles (decoded trajectories) are the red trajectories and the optimal solution is the green trajectory.

have been larger. This is due to unsafe regions (ICS^b states); the longer the tree extends, the more trajectory primitives generated will come close to the ICS^b states (mainly corresponding to the limit of the field-of-view). As a result, many nodes cannot be expanded (i.e., it is not possible to generate p-safe primitives from a given node). In this case, a collision-free braking trajectory is generated to guarantee passive safety (see line # 17 of Algorithm 2). The time cycle must then be chosen neither too small so as to take into account the complexity of ICS^b-CHECK nor too big because of the restrictions of the braking ICS region. Moreover, from a PMP perspective, it will also allow the model of the future to be more frequently updated.

Swarm size. An important parameter in the PASSPMP-PSO's optimization process is the number of swarm particles. PASSPMP-PSO is therefore evaluated for different swarm sizes (N_p). After several tests, some results are presented in Fig. 13. For a number of particles less than 20, the resulting solution (trajectory depicted in green) is not the optimal solution, as the number of decoded trajectories (depicted in red) is small. Noting that for $N_p = 15$, among many runs, the optimal solution can be found but it is random (due to the random initialization of the optimization process). From 20 particles and more, PASSPMP-PSO manages to find the optimal solution. The statistical result regarding the frequency for obtaining optimal trajectory over 100 runs is 100%. However, this is not

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

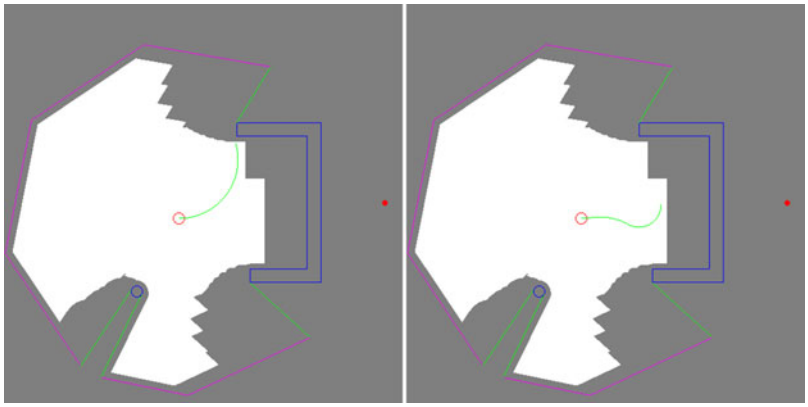

Fig. 14. The behaviour of PASSPMP-PSO (on the left) and the PASSPMP (on the right) in the presence of a U-shaped object.

.

the case for most encoding-based shortest path approaches (e.g., refs. [79, 82]) for which the frequency of obtaining optimal path is less than 100% for the same range of particles values. Besides, we observe for PASSPMP-PSO that beyond 100 particles, decoded trajectories are different when the size changes but their number does not increase. It can be concluded that PASSPMP-PSO performs well for $N_p \in [20, 100]$, and especially N_p must not be less than 20 particles, because it is not sufficient to find the optimal solution.

5.1.4. PASSPMP-PSO *versus* PASSPMP. In this section, PASSPMP-PSO is compared to PASSPMP[9] with respect to the problem of local minima. The two algorithms have been tested in the same conditions and constraints. Figure 14 illustrates their behaviour in a scenario containing a U-shaped fixed object. It is clear that both approaches verify passive motion safety as the robot avoids ICS^b states (grey regions). However, the planned trajectory is different for each case. In PASSPMP, the robot is trapped in a local minimum, because the planned trajectory ends with a braking trajectory and then drives the robot to a stagnation state. As the PASSPMP ultimate objective is to guarantee p-safety, the robot brakes when approaching the ICS^b regions. Furthermore, even if the robot can leave the concave region, its trajectory will not be an optimal solution. In another case, PASSPMP-PSO planned trajectory is an optimal solution and prevents the robot from falling into a local minimum. It can be concluded that PASSPMP-PSO has more performances than PASSPMP in finding a global optimum.

5.1.5. PASSPMP-PSO *versus ISS approach.* This section evaluates the performance of PASSPMP-PSO *wrt* another planning approach, ISS.[49] Among planning approaches, this approach is the most suitable for unknown environments where the robot only uses its on-board sensors for perception. Furthermore, it is very adapted for real-time applications and it is highly applied in road networks and rough terrains. This method is used as a local planner, thus gaining in reactivity. However, the planned trajectories can be biased by a global planner so as to guide the robot towards the goal. Therefore, this method can be easily adapted to such constraints of Fig. 1. These constraints are the most challenging with regard to the safety guarantee criterion, which is the main purpose of PASSPMP-PSO.

ISS is a straightforward approach intended to generate feasible motion planning search space, where all motions sampled in the input space are inherently executable. Each input is simulated using the predictive motion model (dynamics of the robotic system) to determine the shape of the resulting trajectory. The generated trajectories can be evaluated according to a given cost function and can be tested for collision-free.

PASSPMP-PSO and ISS have been tested in the same conditions (as stated in Section 5.1.1) and the respective results have been compared with regard to the safety guarantee issue, the local minima problem and the algorithms running time.

The above sections demonstrate the passive motion safety of PASSPMP-PSO in partially observable environment while considering occlusions, field-of-view limits and objects future trajectories.

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

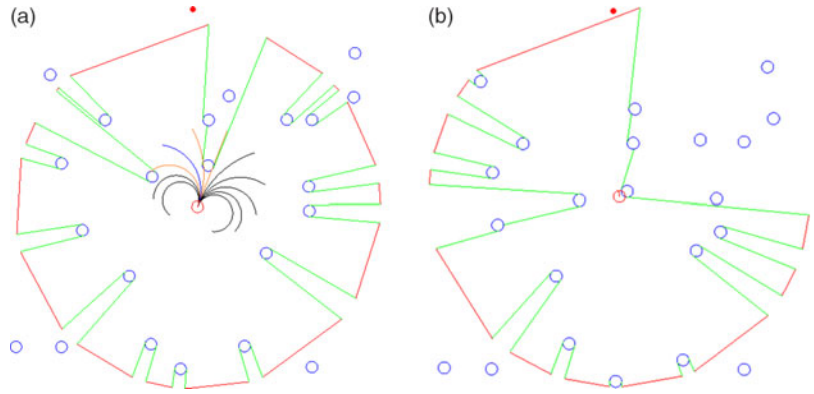

Fig. 15. ISS at work in the scenario of Fig. 11: (a) Snapshot at a given time t_k : $\mathcal{A}$ is in a collision-free state, orange trajectories are not safe and the blue trajectory is the selected collision-free trajectory to apply. (b) Snapshot at time $t_k + \zeta$ (ζ duration of the executed trajectory part): $\mathcal{A}$ is in a collision state with a moving object [which is a state of the previously selected trajectory (blue trajectory in the left snapshot)].

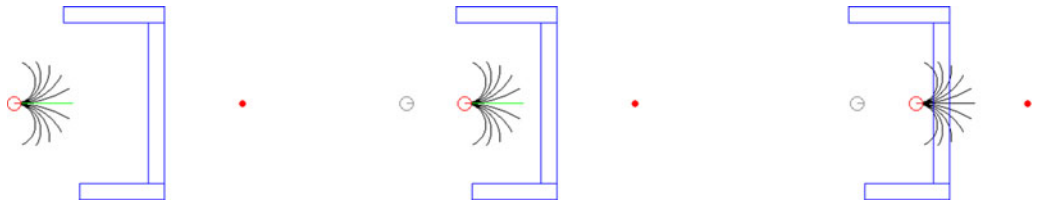

Fig. 16. The behaviour of ISS in the presence of a U-shaped object, a case where the robot is trapped in an unsafe state. The selected trajectory is depicted in green. In the first snapshots (from left), low cost collision-free trajectories to the goal are selected. In the last snapshot (on the right), the robot is in a state where no collision-free trajectory is available. For more clarity, the robot's planning state of the previous snapshot is represented by the grey disc.

To evaluate the safety of ISS approach, it has been tested in a same scenario, namely the environment of Fig. 11. In Fig. 15a, the robot is in a collision-free state and generates a set of 11 trajectories to drive it towards the goal. The trajectories depicted in orange are not safe. Accordingly, a collision-free trajectory with a low cost to the goal is selected (represented in blue). When executing this trajectory, the robot will end up in a collision situation that cannot be avoided as shown in Fig. 15b. Even if the ISS approach can deal with moving objects, their future behaviour (objects dynamics) is not considered. Therefore, even if the robot is in a collision-free state at the present time and its generated trajectory is collision-free, there is no guarantee that collision will not occur in the future. Reasoning about the future is a safety criterion. Missing this criterion, ISS approach will fail at guaranteeing safety.

The second issue for which ISS approach has been evaluated is the local minima problem. In Section 5.1.4, PASSPMP-PSO has been compared to PASSPMP *wrt* this problem. It has been found that PASSPMP-PSO has good performances to find a global optimum, while PASSPMP has been trapped in a local minimum. However, PASSPMP's safety guarantee remains valid. ISS approach has been tested in the same scenario of Fig. 14, with a U-shaped obstacle and for the same conditions and constraints.

Two situations are highlighted: in the first case, the robot can end up in a situation where no collision-free trajectory is available. Figure 16 illustrates ISS results for such a situation. In the two first snapshots (on the left), a collision-free trajectory to the goal is selected (the closest to the goal). However, in the last snapshot, even if the previous trajectory drives the robot to a collision-free state, no collision-free trajectory is available from this state. Therefore, the robot is trapped in a local minimum which, in addition, is unsafe.

In the second situation, the robot has found a collision-free trajectory, which has been selected closest to the goal, as illustrated in Fig. 17. However, the robot is trapped in the concave region of the

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

Table VII. Average running time of PASSPMP-PSO and ISS (for scenario of Fig. 11).

	PASSPMP-PSO	**ISS**
Running time (ms)	29	65

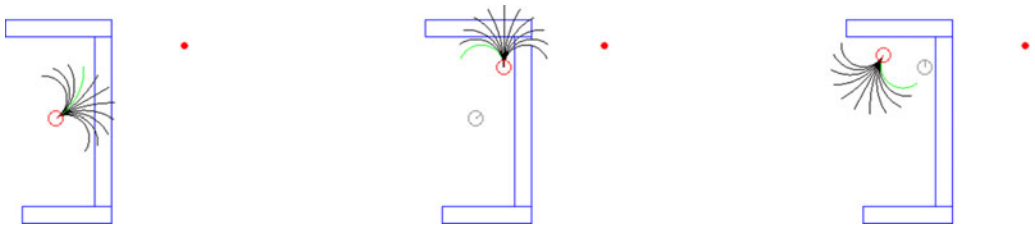

Fig. 17. The behaviour of ISS in the presence of a U-shaped object, a case where the robot is trapped in the concave region of the object (snapshots are depicted in a chronological order from left to right). The selected trajectory is represented in green. The grey disc is the robot's planning state of the previous snapshot.

object indefinitely or until it finds an exit or is driven to an unsafe state (case of the first situation). This confirms that the ISS method is not optimal and no safety guarantee can be provided.

Finally, the running times of PASSPMP-PSO and ISS algorithms are evaluated in Table VII. The tests have been carried out for several runs of each algorithm, for the scenario of Fig. 11 with 22 moving objects. For PASSPMP-PSO, the swarm size is set to 100 particles and $\delta_c = 2.4\,\text{s}$, while for ISS, collision check is performed for a set of 11 trajectories. From Table VII, both algorithms are suitable for real-time applications. However, PASSPMP-PSO presents more performances than ISS. This is due to the parallel process between planning (decision making) and execution (executed trajectory is planned in the previous cycle), where the time cycle is appropriately set.

Noting that PASSPMP-PSO depends mainly on the number of particles (for loops in Algorithm 2 and its corresponding diagram of Fig. 5). However, even if the swarm size can grow exponentially, the computing time of the algorithm can easily decrease, thanks to the use of sub-swarms. The computations are therefore performed for each sub-swarm, where the whole sub-swarms act in parallel (using threads). At the end of the process, the final solution is computed, thanks to the sub-swarms' solutions.

To conclude, after evaluating the results of PASSPMP-PSO and ISS with respect to the motion safety guarantee, the local minima problem and the algorithms running time (for the same test conditions), it has been confirmed that the ISS method is not optimal and no safety guarantees can be provided. By contrast, PASSPMP-PSO generates a trajectory that answers both the safety and optimality conditions. Furthermore, the average running time of the PASSPMP-PSO algorithm is less than 50% of the average running time of the ISS algorithm.

5.2. Experimental results

PASSPMP-PSO has been implemented on a real platform; a robotic wheelchair (presented in Fig. 18) to demonstrate its performances from real experiments. These experiments allow showing PASSPMP-PSO at work in a partially observable environment where the wheelchair uses its onboard sensors only. However, the most challenging issue is to verify in real conditions the passive safety of PASSPMP-PSO. The time cycle is set to $\delta_{cycle} = 0.4\,\text{s}$ and the number of particles is set to $N_p = 50$. The experimental platform is a robotic wheelchair that has been developed at CDTA[3]. It is basically a manual wheelchair that we have motorized by replacing its classic wheels with a motorized chassis based on Brushless motors and have equipped with a hardware and a software architecture and a perception system. The computing resources are mainly partitioned into two parts; (1) the low level part whose main functions are the data acquisition and processing, the control (PID[4]), and the communication of all low level components (e.g., motors, encoders, electronic cards,

[3]CDTA; Center for the Development of Advanced Technologies.

[4]PID, proportional-integral-derivative controller.

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

Fig. 18. Experimental platform; a robotic wheelchair.

etc.), (2) the on-board computer: a computing unit that is obviously essential to support the intelligent part (i.e., the implementation of PASSPMP-PSO). The on-board computer has been selected powerful enough to support all the algorithms developed in this work. It is an Intel Core i7-6500U 2.5 GHz × 4 CPU, 8GB RAM, Intel HD Graphics 520 (Skylake GT2) GPU. The PASSPMP-PSO algorithm has been implemented in C++ under robot operating system[5], a tool that greatly eases implementation, and is very suitable for real-time applications. The robotic wheelchair is mainly equipped with a Laser sensor UST-10LX covering 270° field-of-view and 10 m maximum range, placed in the front part of the wheelchair. The dynamics of the robotic wheelchair is described below.

5.2.1. Robot model. The wheelchair is a Differential drive system, composed of two independently controlled non-steerable fixed wheels. This system is characterized by a 5-tuple state $s = (x, y, \theta, v_r, v_l)$; (x, y) the Cartesian coordinates, θ the orientation and v_r, v_l the right and left wheel velocities. u_r and u_l are respectively the acceleration of the right and left wheels. L denotes $\mathcal{A}$'s wheelbase. The motion of the system is governed by the following dynamics:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v_r} \\ \dot{v_l} \end{bmatrix} = 1/2 \begin{bmatrix} (v_r + v_l)\cos\theta \\ (v_r + v_l)\sin\theta \\ 2(v_r - v_l)/L \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u_r + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_l \tag{23}$$

The system constraints are : $|v_r|, |v_l| \leq 4.16$ m/s and $|v_r|, |v_l| \leq 1.35 m/s^2$.

5.2.2. PASSPMP-PSO *at work.* At first, PASSPMP-PSO is tested in static indoor and outdoor environments. In Fig. 19, the wheelchair evolves in an outdoor environment and, given its initial state, it has to pass through a narrow passage to reach the goal. The obtained results show good performances of PASSPMP-PSO in such a situation.

In Fig. 20, the robotic wheelchair evolves in a corridor and successfully reaches the goal while passing through static objects. Sequential snapshots for wheelchair motion from the initial position to the goal position are represented in this figure. In this case, PASSPMP-PSO generates a safe trajectory to the goal.

However, to better verify the passive motion safety of PASSPMP-PSO, in the same scenario of Fig. 20, a moving object is introduced, where a person obstructs the wheelchair trajectory. In this case, and due to the narrow free space, the wheelchair stops to avoid collision with this moving object (see Fig. 21), where an arbitrary braking trajectory has been executed. Noting that, as no

[5] www.ros.org

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

Fig. 19. PASSPMP-PSO at work in an outdoor environment; sequential snapshots (from top left to bottom right) of wheelchair motion towards a goal while passing through a narrow passage (the red arrow designates the goal).

Fig. 20. PASSPMP-PSO at work in an indoor environment. Sequential snapshots of wheelchair motion (from top left to bottom right); it evolves in a corridor and reaches its goal while avoiding static objects. The goal is just before the tripods.

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

Fig. 21. PASSPMP-PSO at work in the presence of a moving object, sequential snapshots of wheelchair motion (from top left to bottom right). When the pedestrian is detected, an arbitrary collision-free braking trajectory is executed (the first two snapshots). In the third snapshot, the wheelchair is at rest. In the last snapshot, the wheelchair has pursued its trajectory as the passage is free.

Fig. 22. PASSPMP-PSO's behaviour in a dead-end situation. A collision-free braking trajectory is executed to guarantee p-safety. In the last snapshot (on the right), the wheelchair is at rest (there is no issue, even at right).

object's trajectory prediction module is available, the object trajectory is conservatively modelled. Therefore, all object's possible orientations are considered, the fact that explains generating arbitrary braking trajectory since the object orientation cannot be predicted (using a conservative model). This experiment shows how PASSPMP-PSO remains p-safe in the presence of a moving object even in a narrow environment (a corridor).

Another case where safety should be verified is when no safe trajectory driving the system to its goal is available. In Fig. 22, the wheelchair is placed in a dead-end situation; when it is in a state from which no safe trajectory is available, a collision-free braking trajectory is executed to guarantee passive motion safety.

Noting that most state-of-the-art approaches dealing with the motion safety issue (e.g. refs. [9, 14]) are validated in simulation and they are not implemented in real experiments. However, through the different tests presented in this section, the results that have been obtained demonstrate the effectiveness of PASSPMP-PSO, particularly its ability of guaranteeing motion safety in real experiments.

6. Discussion

In this discussion, PASSPMP-PSO is evaluated with respect to other approaches in the field according to motion safety, trajectory optimization and real-world constraints. A theoretical comparison is carried out with the PASSPMP,[9] the ISS[49] and the conservative viability[14] approaches. On the one

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

hand, it is important to evaluate PassPMP-PSO with other reactive planning approaches such as PassPMP and ISS that have the ability to deal with real-world constraints being similar to those addressed in this paper (e.g., partial knowledge of the environment due to sensory limitations). An experimental comparison has already demonstrated the performances of PassPMP-PSO over PassPMP and ISS (see Sections 5.1.4 and 5.1.5). However, in this section, the obtained results are justified in a formal way. On the other hand, PassPMP-PSO is evaluated *wrt* eminent guaranteed safe navigation approaches such as the conservative viability.[14]

PassPMP[9] is a previous work based on PMP and passive motion safety, for which the problem of trajectory optimization has not been addressed. ISS[49] is a straightforward technique for generating a feasible motion planning search space because all motions sampled in the input space are inherently executable. It is designed as a local planner within the sensors' field-of-view. A global planner is also used to guide the robot towards the goal. In ref. [14], the conservative viability algorithm has addressed the issue of controlling dynamical systems subject to viability constraints, which generally define states within which the system should remain. A viable state is guaranteed to have at least one sequence of controls which, when applied from this state, will keep the robotic system within the viability constraint set indefinitely.

6.1. Motion safety

In order to analyse the safety issue, three criteria must be considered:[94] system dynamics, reasoning about the future and an appropriate time horizon. Each navigation scheme should verify these three criteria. If one of them is violated, the risk of collision will be inevitable between the robot and the environment's objects.

Thanks to the definitions and properties of PassPMP-PSO, presented in previous sections, it is possible to prove that PassPMP-PSO verifies all the three safety criteria. From Eqs. (5) and (8), it is clear that the optimal solution has to verify the safety condition. To guarantee safety, PassPMP-PSO is based on the passive motion safety defined by duality *wrt*, the braking ICS concept (braking ICS). Definitions 2 and 3 determine, respectively, what a passively safe and a braking ICS state are. These definitions are, respectively, formulated in Eqs. (6) and (7). According to these two equations, it is clear that passive safety (respectively braking ICS) is defined from $\mathcal{A}(\tilde{s}(s, \tilde{u}_b, t))$ and $\mathcal{W}(t)$.

(1) $\mathcal{A}(\tilde{s}(s, \tilde{u}_b, t))$ is obtained by integrating the differential equation of the dynamics of $\mathcal{A}$ (Eq. (1)), that is, both the kinematic and dynamic models of the system are considered. *The first safety criterion is then verified.*
(2) $\mathcal{W}$ designates the space $\times$ time model of the future evolution of the environment with an unknown behaviour. The collision check is performed for $\mathcal{W}(t)$ at each time t belongs to a well-defined time interval (test $\mathcal{A}(\tilde{s}(s, \tilde{u}_b, t)) \cap \mathcal{W}(t)$, $\forall t \in [0, t_b[$). *The second safety criterion regarding the reasoning about the future is, thus, verified.*
(3) Finally, *the last criterion about using an appropriate time horizon is also verified by* PassPMP-PSO. The principle is already rooted in the braking ICS concept, which considers trajectories of finite duration (given Eqs. (6) and (7)). The collision check, for a given braking trajectory $\tilde{u}_b$ is limited to the time interval $[0, t_b[$. For the whole set of possible braking trajectories, the collision check is limited to the time interval $[0, T_h[$ (where T_h is a function of t_b as formalized in Eq. (21)) and the model of the future is considered until this time T_h, which is clearly illustrated by Property 3.

As for PassPMP,[9] because it is based on a concept similar to PassPMP-PSO to guarantee safety, namely passive motion safety, the three safety criteria are also verified.

Similarly to PassPMP-PSO and PassPMP, ISS approach[49] is evaluated *wrt* the three safety criteria:

(1) In ref. [49], ISS approach is applied for vehicles, where their motion is governed by a differential equation as in Eq. (1). Therefore, *the first safety criterion is satisfied as the dynamics of the system is considered.*
(2) For the second criterion which concerns the reasoning about the future, ISS approach handles moving objects but their dynamics (future evolution) is not considered at all.[49] In this case,

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

collision check is performed at current time without any consideration of the future. However, even if a state is collision-free at the present time, there is no guarantee that collision will not occur in the future. Therefore, *the second safety criterion is not verified*.

(3) As the second safety criterion is not verified, *the third criterion about the appropriate time horizon (collision check up to a limited time) is automatically not verified either*, where ISS approach basically does not reason about the future.

Finally, the three safety criteria are applied for the conservative viability.[14]

(1) The approach is applied for a continuous-time dynamical system whose dynamics is described by a differential equation having the form of Eq. (1). Therefore, *the first safety criterion is satisfied*.

(2) In viability terms, the viability constraint set K_c within which $\mathcal{A}$ must be kept is the set of states where $\mathcal{A}$ is not in collision with any of the workspace obstacles, such that:[14]

$$K_c = \{s(t) \in \mathcal{S} | \mathcal{A}(s(t)) \cap B(t) = \emptyset\} \tag{24}$$

where $\mathcal{A}(s(t))$ the closed subset of the workspace occupied by system $\mathcal{A}$ when it is in the state $s(t)$. B denotes the union of the workspace objects and $B_i(t)$ is the closed subset of the workspace occupied by an object B_i at time t, with

$$B = \bigcup_{i=1}^{b} B_i = \bigcup_{i=1}^{b} B_i([0, \infty)) = \bigcup_{i=1}^{b} \bigcup_{t \in [0,\infty)} B_i(t) \tag{25}$$

where the trajectory of B_i is *a priori* known (e.g., a periodic behaviour with constant velocity). From the viability constraint set of Eq. (24), it is clear that the reasoning about the future is considered. However, as this approach uses objects with *a priori* known behaviours, *the second safety criterion cannot be satisfied in the real world, where objects' trajectories are unknown.*

(3) According to Eqs. (24) and (25), the viability constraints are defined for objects with infinite trajectories (i.e., the model of the future is considered up to infinity). Therefore, the collision check is done for an infinite time interval, which is ideal for guaranteeing absolute safety, but it is not possible in real situations. Nevertheless, it is possible to identify two classes of situations for which it is possible to set a limited time horizon: the freezing case and the periodic case.[14] In these two cases, *the third safety criterion about the use of an appropriate time horizon is satisfied*.

To sum up, under real-world constraints (robot with a limited field-of-view, environment containing moving objects with unknown future behaviour, etc.), the PassPMP-PSO and the PassPMP approaches satisfy all the three safety criteria. The ISS approach satisfies only the first criterion. As for the conservative viability, the first criterion is always satisfied, the second criterion is satisfied under some assumptions (*a priori* known future behaviour of objects up to infinity) and the third one depends on the applications at hand (freezing case or periodic case).

6.2. Trajectory optimization

The trajectory optimization issue is challenging in partially unknown environment due to the fact that the system has only a partial knowledge of its environment.

PassPMP-PSO computes partial trajectories to the goal so as to deal with real-world situations. In this case, it is not possible to ensure optimality. An alternative, though, is to subdivide the optimization problem to sub-problems where the optimal solution is defined for each sub-problem. Consequently, given the planning concept and the harsh constraints imposed by the environment and the robot, PassPMP-PSO can generate a near-optimal partial trajectory at each planning cycle (see Definition 5 about the near-optimality of PassPMP-PSO) thanks to the tree encoding-based PSO technique.

For PassPMP and conservative viability, the problem of trajectory optimization is not addressed where the priority of these approaches is to guarantee motion safety at a first stage. Instead, PassPMP selects the trajectory to apply by minimizing a simple cost function which depends on

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

the trajectory's distance and time costs. As for the conservative viability, the choice of the control is performed by minimizing the distance to a goal.

The ISS approach is mainly based on generating local motion plans by sampling in the control space. In this case, the trajectory optimization problem is not solved. However, a global planner is used to find the optimal (the lowest-cost) path among the vehicle's local paths. It is based on Field D*[95] (a grid-based planner) that provides global guidance to the local motion planner; the path with the lowest Field D* distance to the goal (based on the part of the world that is known at that point) is selected. Like most grid-based planners, it produces "jerky" paths that consist of straight and diagonal line segments through a grid. In fact, the true optimal path cannot typically be represented in the grid, since all paths are required to pass through grid vertices.[49] The resulting path is therefore not the optimal solution.

6.3. Real-world constraints

When a mobile robot is led to navigate in the real world, it is necessary to mainly handle (1) moving objects with unknown future behaviours (to prevent possible collisions in the future) and (2) sensory limitations, when using the embedded sensors of the robot to perceive the environment, only a partial view of its surroundings is possible, leading to the presence of known and unknown regions that represent a risk in case they are not handled on time.

PASSPMP-PSO takes into consideration the unknown future behaviour of objects and solves the problem by considering all the possible trajectories of an object, thanks to a conservative model of the future (as previously explained). To guarantee safety, PASSPMP-PSO computes braking ICS states that must be avoided at all times, thanks to Eq. (7) that can be rewritten in the following form:

$$\mathrm{ICS}^b(\mathcal{W}) = \Big\{ s \in \mathcal{S} | \forall \tilde{u}_b \in \tilde{\mathcal{U}}_b^s, \exists t \in [0, t_b[, \\ \mathcal{A}(\tilde{s}(s, \tilde{u}_b, t)) \cap (\delta FoV(t) \cup FoV(t) \cup FoV^c(t)) \neq \emptyset \Big\} \tag{26}$$

Such that $\mathcal{W} = \delta FoV \cup FoV \cup FoV^c$, with δFoV the boundary of the field-of-view, FoV the subset of the workspace perceived by $\mathcal{A}$ (the region inside the field-of-view) and FoV^c the region outside the field-of-view. δFoV contains seen objects δFoV^s and unseen objects δFoV^u (sensing limits and occluded objects), that is, $\delta FoV = \delta FoV^s \cup \delta FoV^u$. Therefore, PASSPMP-PSO is conceptually designed to handle seen and unseen moving objects with unknown future behaviours.

Nevertheless, based on a conservative model of the future, a high level of safety can be guaranteed as all possible trajectories of an object are considered. However, it could be interesting to think about a less restrictive solution while maintaining a high level of safety (which is not yet the case with other existing models) and which is applicable in real-world experiments.

The second approach, the PASSPMP, is also able to deal with such constraints as a similar concept has been adopted (a conservative model of the future has been used).

In the ISS approach, the first point about the dynamics of moving objects is not addressed. Indeed, their future behaviour is not taken into account. For the second constraint, the ISS approach deals with sensory limitations, where the known part of the environment is the perceived region. Only objects present in this region are considered in collision avoidance. Unseen objects (unknown region) are not handled even though they represent a risk of collision.

In the conservative viability approach, two cases of study have been considered: pursuit and evasion.[14] In the pursuit case (the system $\mathcal{A}$ is the observer), it is assumed that $\mathcal{A}$ is equipped with an omnidirectional sensor and it should always maintain one or more workspace targets B_i^t (the target) within its field-of-view at all times while avoiding collisions. The corresponding viability constraint set in this case is

$$K_c = \left\{ s(t) \in \mathcal{S} | \bigwedge_i FoV(s(t)) \cap B_i^t(t) \neq \emptyset \right\} \tag{27}$$

In the evasion case ($\mathcal{A}$ is the target), the moving obstacles are assumed to be sentinels on patrol duty, and they are equipped with omnidirectional sensors with a limited field-of-view. $\mathcal{A}$ should alwaysstay

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

out of the field-of-view of one or more workspace observers B_i^0 at all times. The corresponding viability constraint set in this case is

$$K_c = \left\{ s(t) \in \mathcal{S} | \bigwedge_i \mathcal{A}(s(t)) \cap FoV(B_i^0(t)) = \emptyset \right\} \tag{28}$$

Therefore, for the first constraint about the unknown behaviour of moving objects, Eqs. (27) and (28) confirm that the conservative viability considers the dynamics of objects but in return, it is assumed that they have a periodic behaviour with constant velocity. That is, the future evolution of an object is *a priori* known. For the second constraint, an advantage of this work is that sensory limitations are considered. However, according to the two addressed cases, it is clear from Eqs. (27) and (28) that only the known region (either *FoV* or B_i^t) is implied to determine the viability constraint set. The unknown regions and their future evolution are not considered, despite being one of the real-world constraints representing an imminent risk of collision which may not be avoided on time.

From this study, it can be concluded that PASSPMP-PSO has the particularity over the other methods to propose a solution that solves both motion safety and trajectory optimization issues; it meets the three safety criteria, offers a near-optimal solution and deals with real-world constraints. As a drawback, PASSPMP-PSO is based on a conservative model of the future to guarantee a high level of safety, but it can be constraining in some applications. Therefore, PASSPMP-PSO can be improved further by exploring other models (which remains an open issue).

7. Conclusion and Future Works

Motion safety and trajectory optimization are two key issues that have always been addressed separately even if they are both required to solve a motion planning problem. This paper has proposed the PASSPMP-PSO motion planner that integrates both the two issues in a formal way. In its general form, PASSPMP-PSO is based on the PMP concept, a periodic process that interleaves planning and execution for a regular update of the world model. A set of partial trajectories is computed to drive the robot from its initial state to the goal state. These trajectories are generated, thanks to a new tree encoding technique based on PSO. Unlike previous approaches, addressing encoding-based shortest path problem for the case of a network graph, the proposed encoding technique is applied to RRTs expanded in the state × time space. From a set of nodes, partial trajectories are built and encoded into particles using a modified priority-based encoding. The candidate solution (best particle) is selected, thanks to an objective function which is based on three important constraints: passive motion safety guarantee, minimum trajectory time cost and minimum distance to the goal. Passive motion safety is obtained by avoiding braking ICS at all times that are defined as states for which, whatever the future braking trajectory followed by the robot, a collision occurs before it is at rest.

The simulation and real experiment results show that the PASSPMP-PSO algorithm is able to generate a passively safe near-optimal trajectory, while dealing with the robot's limited field-of-view, occlusions and the unknown future behaviour of perceived and unexpected moving objects at once. For a population between 20 and 100 particles, the statistical result related to the frequency of obtaining optimal trajectory over 100 runs is 100%, and the risk that collision happens when the robot is moving is zero (i.e., passive motion safety is guaranteed). Furthermore, the encoding technique has been tested for different tree expansions from 13 to more than 200 nodes, and the results are very encouraging as the algorithm converged rapidly to the optimal solution.

PASSPMP-PSO has been evaluated with respect to other motion planning approaches. An experimental comparison has been carried out with the ISS approach. The results obtained demonstrate that, for the same test conditions, the ISS algorithm fails to guarantee safety and to fulfil the optimality condition, compared to PASSPMP-PSO which is able to verify the two conditions. Furthermore, it has been confirmed that both the PASSPMP-PSO and ISS algorithms are suitable for real-time applications. However, the average running time of PASSPMP-PSO algorithm is less than 50% of the average running time of the ISS algorithm. As for PASSPMP algorithm, unlike ISS, the safety condition is verified, but the comparison results prove that PASSPMP-PSO has more performances to find a global optimum while PASSPMP is easily trapped in a local minimum.

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

This work could be extended in the following directions:

- To guarantee motion safety, PASSPMP-PSO uses a natural behaviour by braking when collision is inevitable. It could also be worth exploring other levels of motion safety, for instance, the passive friendly motion safety[34,96] which guarantees that, in case a collision takes place, the robot will be at rest and the colliding object could have the time to stop or avoid the collision. Nevertheless, in this case, the safety of the robot is strongly related to the decision taken by the other colliding objects, the fact that could make it too limited in some applications. In general, other forms of motion safety can be explored depending on the particularity of the navigation problem at hand.
- In this work, a conservative model of the future is adopted to provide a strict guarantee of motion safety. This model considers all possible future motions for an object with an arbitrary dynamics. In return, this representation is constraining with respect to the ICS^b-free space and is directly related to the time horizon value. In real-world situations, the suitable solution is either conservative or probabilistic. The probabilistic model is appropriate to support uncertainties, but approaches using this model offer no strict motion safety guarantees. Instead, the risks of collision are minimized. An extension to this work would be to find a compromise between conservative and probabilistic models by proposing a model of the future capable of overcoming the drawbacks of the two models, that is, propose an alternative solution, other than existing models, which would allow to keep a high level of safety while respecting real constraints (e.g., the unknown future behaviour of objects).
- It is now more obvious, from experiences like self-driving, that the safety issue is not a single technological area. Safety guarantee requires coordinating multi-disciplinary areas: software, hardware, human–computer interaction, robotics, etc. For example, a collision can happen due to a software crash, a hardware failure or reasoning errors (decision-making errors), where the latter is the concern of robotic field. Therefore, existing motion safety-based approaches should be expanded to the other fields to really get closer to an absolute motion safety.
- It could also be interesting to implement the PASSPMP-PSO algorithm in a real self-driving system where the question "will the driver seat ever be empty?" always remains open.
- This work is a first step towards solving safe trajectory optimization problems with bio-inspired approaches. However, in biology, there are more pertinent systems from which we can learn and propose new concepts.

Acknowledgment

We thank Tarik Ouacine and Samir Benabadji for their help in real-world experiments. This work has been tested on an autonomous wheelchair developed as part of the project FAURSA financially supported by the Directorate General for Scientific Research and Technological Development (DGRSDT).

References

1. D. Althoff, J. J. Kuffner, D. Wollherr and M. Buss, "Safety assessment of robot trajectories for navigation in uncertain and dynamic environments," *Auton. Robots* **32**(3), 285–302 (2012).
2. S. Bouraine, T. Fraichard and H. Salhi, "Provably safe navigation for mobile robots with limited field-of-views in dynamic environments," *Auton. Robots* **32**(3), 267–283 (2012).
3. S. Bouraine, T. Fraichard and H. Salhi, "Provably Safe Navigation for Mobile Robots with Limited Field-of-Views in Unknown Dynamic Environments," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (2011).
4. L. Heon-Cheol, Y. Touahmi and L. Beom-Hee, "Grafting: A path replanning technique for rapidly-exploring random trees in dynamic environments," *Adv. Robot.* **26**(18), 2145–2168 (2012).
5. J. van den Berg and M. Overmars, "Roadmap-based motion planning in dynamic environments," *IEEE Trans. Robot.* **21**(5), 885–897 (2005).
6. F. A.Yaghmaie, A. Mobarhani and H. D.Taghirad, "A New Method for Mobile Robot Navigation in Dynamic Environment: Escaping Algorithm," *Proceeding of the 2013 RSI/ISM International Conference on Robotics and Mechatronics* (2013) pp. 212–217.
7. M. Seder and I. Petrovic, "Dynamic Window Based Approach to Mobile Robot Motion Control in the Presence of Moving Obstacles," *IEEE International Conference on Robotics and Automation* (2007).
8. D. Wilkie, J. van den Berg and D. Manocha, "Generalized Velocity Obstacles," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (2009) pp. 5573–5578.
9. S. Bouraine, T. Fraichard, O. Azouaoui and H. Salhi, "Passively Safe Partial Motion Planning for Mobile Robots with Limited Field-of-Views in Unknown Dynamic Environment," *IEEE International Conference on Robotics and Automation* (2014) pp. 3576–3582.

https://doi.org/10.1017/S0263574720000806 Published online by Cambridge University Press

10. J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *IEEE International Conference on Neural Networks*, vol. 4 (1995) pp. 1942–1948.
11. M.C. Hager and G.S. Helfman, "Safety in numbers: Shoal Size choice by minnows under predatory threat," *Behav. Ecol. Sociobiol.* **29**(4), 271–276 (1991).
12. T. Fraichard, "Will the driver seat ever be empty?," *ERCIM News, ERCIM* (2017) pp. 39–40.
13. T. Fraichard and H. Asama, "Inevitable collision states. A step towards safer robots?," *Adv. Robot.* **18**(10), 1001–1024 (2004).
14. M.A. bouguerra, T. Fraichard and M. Fezari, "Viability-based guaranteed safe robot navigation," *J. Intell. Robot. Syst.* **95**(2), 459–471 (2019).
15. M. Blaich, S. Weber, J. Reuter and A. Hahn, "Motion Safety for Vessels: An Approach Based on Inevitable Collision States," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1077–1082 (2015).
16. N. Bohorquez, A. Sherikov, D. Dimitrov and P. B. Wieber, "Safe Navigation Strategies for a Biped Robot Walking in a Crowd," *IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)* (2016) pp. 379–386.
17. J. Snape, V. Berg, S. J. Guy and D. Manocha, "Independent Navigation of Multiple Mobile Robots with Hybrid Reciprocal Velocity Obstacles," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (2009) pp. 5917–5922.
18. B. Gopalakrishnan, A. Singh and K. Krishna, "Time Scaled Collision Cone Based Trajectory Optimization Approach for Reactive Planning in Dynamic Environments," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (2014) pp. 4169–4176.
19. C. Shuai, X. Junhao and L. Huimin, "Real-Time Obstacle Avoidance Using Subtargets and Cubic B-spline for Mobile Robots," *IEEE International Conference on Information and Automation (ICIA)* (2014) pp. 634–639.
20. Y. Lu, Z. Xi and J. M. Lien, "Collision Prediction Among Polygons with Arbitrary Shape and Unknown Motion," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (2014) pp. 4148–4153.
21. A. Lawitzky, A. Nicklas, D. Wollherr and M. Buss, "Determining States of Inevitable Collision Using Reachability Analysis," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (2014) pp. 4142–4147.
22. J. Pan, L. Zhang and D. Manocha, "Collision-free and smooth trajectory computation in cluttered environments," *J. Robot. Res.,* 1155–1175 (2012).
23. S. Sarid, B. Xu and H. Kress-Gazit, "Guaranteeing High-Level Behaviors while Exploring Partially Known Maps," **In:** *Robotics: Science and Systems* (MIT Press, 2013) pp. 377–384.
24. H. Taubig, U. Frese, C. Hertzberg, S. Mohr, E. Vorobev and D. Walter, "Guaranteeing functional safety: design for provability and computer-aided verification," *Auton. Robots* **32**(3), 303–331 (2012).
25. L. Pallottino, V. Scordio, A. Bicchi and E. Frazzoli, "Decentralized cooperative policy for conflict resolution in multivehicle systems," *IEEE Trans. Robot.* **23**(6), 1170–1183 (2007).
26. J. Van den Berg and M. Overmars, "Planning time-minimal safe paths amidst unpredictably moving obstacles," *Int. J. Robot. Res.* **27**(11–12), 1274–1294 (2008).
27. E. Lalish and K. Morgansen, "Decentralized Reactive Collision Avoidance for Multivehicle Systems," *IEEE Conference on Decision and Control* (2008).
28. K. Bekris, K. Tsianos and L. Kavraki, "Safe and Distributed Kinodynamic Replanning for Vehicular Networks Mobile Networks and Applications," *IEEE Conference on Decision and Control*, vol. 14(3) (2009).
29. D. Hsu, R. Kindel, J.C. Latombe and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *Int. J. Robot. Res.* **21**(3), 233–255 (2002).
30. K. Bekris and L. Kavraki, "Greedy but Safe Replanning under Kinodynamic Constraints," *IEEE International Conference on Robotics and Automation* (2007).
31. D. Seward, C. Pace and R. Agate, "Safe and Effective Navigation of Autonomous Robots in Hazardous Environments Autonomous Robots," *IEEE International Conference on Robotics and Automation*, vol. 22 (2007) pp. 223–242.
32. J. van den Berg, P. Abbeel and K. Goldberg, "LQG-MP: Optimized Path Planning for Robots with Motion Uncertainty and Imperfect State Information," *Robot. Sci. Syst.* **30**(7), 895–913 (2011).
33. S. LaValle and J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.* **20**(5), 378–400 (2001).
34. K. Macek, D. Vasquez-Govea, T. Fraichard and R. Siegwart, "Towards safe vehicle navigation in dynamic urban scenarios," *Automatika* **50**(3–4), 184–194 (2009).
35. D. Fergusson, N. Kalra and A. Kuffner, "Anytime Path Planning and Replanning in Dynamic Environment," *IEEE International Conference on Robotics and Automation* (2006) pp. 2366–2371.
36. H. Kurniawati and T. Fraichard, "From Path to Trajectory Deformation," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (2007) pp. 159–164.
37. V. Delsart and T. Fraichard, "Navigating Dynamic Environments Using Trajectory Deformation," *Proceedings IEEE International Conference on Intelligent Robots and Systems* (2008) pp. 226–233.
38. J. Horwood, N. Aragon and A. Poore, "Gaussian sum filters for space surveillance: Theory and simulation," *J. Guid. Control Dynam.* **34**(6), 1839–1851 (2011).
39. Y. Bestaoui Sebbane, "Planning and decision making for aerial robots," *Intell. Syst. Cont. Autom. Sci. Eng.* **71** (2014).

40. A. Stentz, "The Focussed d* Algorithm for Real-Time Replanning," *Proceedings of the International Joint Conference on Artificial Intelligence* (1995) pp. 1652–1659.
41. D. Ferguson and A. Stentz, "Anytime RRTs," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (2006) pp. 5369–5375.
42. M. Zucker, J. Kuffner and M. Branicky, "Multipartite RRTs for Rapid Replanning in Dynamic Environments," *IEEE International Conference on Robotics and Automation* (2007) pp. 1603–1609.
43. K. Macek, M. Becked and R. Siegwart, "Motion Planning for Car-Like Vehicles in Dynamic Urban Scenarios," *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2006) pp. 4375–4380.
44. L. Heon-Cheol, Y. Touahmi and L. Beom-Hee, "Grafting: A path replanning technique for rapidly exploring random trees in dynamic environments," *Adv. Robot.* **26**(18), 2145–2168 (2012).
45. L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma and N. Zheng, "Efficient sampling-based motion planning for on-road autonomous driving," *IEEE Trans. Intell. Transp. Syst.* **16**(4), 1961–1976 (2015).
46. S. Karaman, and E. Frazzoli, "Sampling based algorithms for optimal motion planning," *Int. J. Robot. Res.* **16**, 846–894 (2011).
47. T. Fraichard and T. Howard, "*Iterative Motion Planning and Safety Issue,*" **In:** *Handbook of Intelligent Vehicles* (Springer, 2012) pp. 1433–1458.
48. A. Kelly and T. Stentz, "Rough terrain autonomous mobility - Part 2: An active vision and predictive control approach," *Auton. Robots* **5**(2), 163–198 (1998).
49. A. Kelly, T. Stentz, O. Amidi, M. Bode, D. Bradley, R. Mandelbaum, T. Pilarski, P. Rander, S. Thayer, N. Vallidis and R. Warner, "Toward reliable off-road autonomous vehicles operating in challenging environments," *Int. J. Robot. Res.* **25**(5–6), 449–483 (2006).
50. R. Knepper, S. Srinivasa and M. Mason, "An Equivalent Relation for Local Path Sets," *Proceedings of the Ninth International Workshop on the Algorithmic Foundations of Robotics*, vol. **68** (2010) pp. 19–35.
51. T. Howard, C. Green, A. Kelly and D. Ferguson, "State space sampling of feasible motions for high performance mobile robot navigation in complex environments," *J. Field Robot.* **25**(6–7), 325–345 (2008).
52. F. von Hundelshausen, M. Himmelsbach, F. Hecker, A. Mueller and H.-J. Wuensche "Driving with tentacles-integral structures of sensing and motion," *J. Field Robot.*, 64–673 (2008).
53. A. Cherubini, F. Spindler and F. Chaumette, "A New Tentacles-Based Technique for Avoiding Obstacles During Visual Navigation," *IEEE International Conference on Robotics and Automation*, vol. 20 (2012) pp. 4850–4855.
54. B. Oliver and O. Khatib, "High-Speed Navigation Using the Global Dynamic Window Approach," *IEEE International Conference on Robotics and Automation*, vol. 1 (1999).
55. E. Demeester, M. Nuttin, D. Vanhooydonck, G. Vanacker and H. Van Brussel, "Global dynamic window approach for holonomic and non-holonomic mobile robots with arbitrary cross-section," *IEEE Trans. Neural Netw.,* 2357–2362 (2005).
56. S. Petti and T. Fraichard, "Safe Motion Planning in Dynamic Environments," *Proceeding of the IEEE-RSJ International Conference on Intelligent Robots and Systems* (2005) pp. 2210–2215.
57. J. Richalet, A. Rault, T. Testud and J. Papon, "Model predictive heuristic control: Applications to industrial processes," *Automatica* **14**(5), 413–428 (1978).
58. M. Farrokhsiar and H. Najjaran, "Unscented Predictive Motion Planning of a Nonholonomic System," *IEEE International Conference on Robotics and Automation* (2011) pp. 4480–4485.
59. S. Maniatopoulos, D. Panagou and K. Kyriakopoulos, "Model Predictive Control for the Navigation of a Nonholonomic Vehicle with Field-of-View Constraints," *American Control Conference (ACC)* (2013) pp. 3967–3972.
60. M. Hoy, and A. S. Matveev and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey," *Robotica* **33**(3), 463–497 (2015).
61. P. Hart, N. Nilsson and B. Raphael, "Formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.* **4**(2), 100–107 (1968).
62. E. W.Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.* **1**(1), 269–271 (1959).
63. M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor and S. Schaal, "STOMP: Stochastic Trajectory Optimization for Motion Planning," *IEEE International Conference on Robotics and Automation* (2011) pp. 4569–4574.
64. N. D. Ratliff, M. Zucker, J. A. Bagnell and S. S. Srinivasa, "CHOMP: Gradient Optimization Techniques for Efficient Motion Planning," *IEEE International Conference on Robotics and Automation* (2009) pp. 489–494.
65. M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell and S. S. Srinivasa, "CHOMP: Covariant Hamiltonian optimization for motion planning," *Int. J. Robot. Res.* **32**(9–10), 1164–1193 (2013).
66. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs* (Springer, New York, NY, USA, 1996).
67. A. Colorni, M. Dorigo and V. Maniezzo, "Distributed Optimization by Ant Colonies," *European Conference on Artificial Life, France* (1991) pp. 134–142.
68. M. Dorigo, *Optimization, Learning and Natural Algorithms* (Politecnico di Milano, Italie, 1992).
69. J. Freeman and D. Skapura, *Neural Networks: Algorithms, Applications and Programming Techniques* (Addison Wesley, 1991).
70. A. Ghorbani, "Using Genetic Algorithm for a Mobile Robot Path Planning," *International Conference on Future Computer and Communication* (2009) pp. 164–166.

71. K. Sugawara, "Foraging Behavior of Interacting Robots with Virtual Pheromone," *Proceedings of the International Conference on Intelligent Robots and Systems*, vol. 3 (2004) pp. 3074–3079.
72. H. Qu, "Real-time robot path planning based on a modified pulse-coupled neural network model," *IEEE Trans. Neural Netw.* **20**(11), 1724–1739 (2009).
73. X. Chen and Y. Li, "Smooth Path Planning of a Mobile Robot Using Stochastic Particle Swarm Optimization," *Proceedings of the IEEE International Conference on Mechatronics and Automation* (2006) pp. 1722–1727.
74. Q. Qin, "Path Planning for Mobile Robot Using the Particle Swarm Optimization with Mutation Operator," *Proceedings of International Conference on Machine Learning and Cybernetics*, vol. 4 (2004) pp. 2473–2478.
75. M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: A review," *Evol. Comput.* **25**(1), 1–54 (2016).
76. M. Munemoto, Y. Takai and Y. Sato, "A Migration Scheme for Genetic Adaptive Routing Algorithm," *IEEE International Conference on Systems, Man, and Cybernetics* (1998) pp. 2774–2779.
77. J. Inagaki, M. Haseyama and H. Kitajima, "A genetic Algorithm for Determining Multiple Routes and Its Applications," *IEEE International Symposium on Circuits and Systems* (1999) pp. 137–140.
78. C. Ahn and R. Ramakrishna, "A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations," *IEEE Trans. Evol. Comput.*, 566–579 (2002).
79. M. Gen, R. Cheng and D. Wang, "Genetic Algorithms for Solving Shortest Path Problems," *Proceeding of the IEEE International Conference on Evolutionary Computation* (1997) pp. 401–406.
80. G. Raidl, "A Weighted Coding in a Genetic Algorithm for the Degree Constrained Minimum Spanning Tree Problem," *Proceeding of the 2000 ACM Symposium on Applied Computing* (2000) pp. 440–445.
81. A. W. Mohemmed and N. C. Sahoo, "Particle Swarm Optimization Combined with Local Search and Velocity Re-Initialization for Shortest Path Computation in Networks," *Proceedings of the 2007 IEEE Swarm Intelligence Symposium* (2007) pp. 266–272.
82. A. W. Mohemmed and N. C. Sahoo, "Efficient Computation of Shortest Paths in Networks Using Particle Swarm Optimization and Noising Metaheuristics," *Discrete Dynamics in Nature and Society* (2007) pp. 1–25.
83. S. V. Ragavan, S. G. Ponnambalam and C. Sumero, "Waypoint-Based Path Planner for Mobile Robot Navigation Using PSO and GA-AIS," *2011 IEEE Recent Advances in Intelligent Computational Systems* (2011) pp. 756–760.
84. K. Su, Y. Wang and X. Hu "Middle node optimization algorithm for global optimal path planning," *Int. J. Adv. Comput. Sci. Appl. (IJACSA)* **6**(4) (2015).
85. A. Bautin, L. Martinez-Gomez and T. Fraichard "Inevitable Collision States: A Probabilistic Perspective," *IEEE International Conference on Robotics and Automation* (2010) pp. 4022–4027.
86. D. Althoff, M. Althoff, D. Wollherr and M. Boss "Probabilistic Collision State Checker for Crowded Environments," *IEEE International Conference on Robotics and Automation* (2010) pp. 1492–1498.
87. T. Fraichard and T. M. Howard "Iterative Motion Planning and Safety Issue," **In:** *Handbook of Intelligent Vehicles* (Springer, London, 2012) pp. 1433–1458.
88. F. Rohrmuller, M. Althoff, D. Wollherr and M. Buss "Probabilistic Mapping of Dynamic Obstacles Using Markov Chains for Replanning in Dynamic Environments," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (2008) pp. 2504–2510.
89. D. Vasquez, T. Fraichard and C. Laugier, "Growing hidden Markov models: An incremental tool for learning and predicting human and vehicle motion," *Int. J. Robot. Res.* **28**(11–12), 1486–1506 (2009).
90. A. Broadhurstr, S. Baker and T. Kanade "Monte Carlo Road Safety Reasoning," *Proceedings IEEE Intelligent Vehicles Symposium* (2005) pp. 319–324.
91. A. Kushleyev and M. Likhachev "Time-Bounded Lattice for Efficient Planning in Dynamic Environments," *IEEE International Conference on Robotics and Automation* (2009) pp. 1662–1668.
92. T. Schouwenaars, "Safe Trajectory Planning of Autonomous Vehicles," *Thesis* (Massachusetts Institute of Technology, 2006).
93. Y. Shi and R. Eberhart, "A Modified Particle Swarm Optimizer," *IEEE World Congress on Computational Intelligence* (1998) pp. 69–73.
94. T. Fraichard, "A Short Paper About Motion Safety," *IEEE International Conference on Robotics and Automation (ICRA)* (2007) pp. 1140–1145.
95. D. Ferguson and A. Stentz, "Field D*: An Interpolation-Based Path Planner and Replanner," Carnegie Mellon Robotics Institute, *Technical Report* (CMU-RI-TR-05-19, 2005).
96. S. Mitsch, K Ghorbal and A. Platzer,"On provably safe obstacle avoidance for autonomous robotic ground vehicles," *Robotics Science and Systems (RSS)* (2013).