Real-time Safe Path Planning for Robot Navigation in Unknown Dynamic Environments

Sara BOURAINE [1], Thierry FRAICHARD [2], Ouahiba AZOUAOUI [1]

[1] Centre de Développement des Technologies Avancées (CDTA), Baba Hassen, Algeria.
[2] Univ. Grenoble Alpes, Inria, CNRS, LIG, Grenoble, France.
s_bouraine@yahoo.fr, thierry.fraichard@inria.fr,
oazouaoui@cdta.dz

Abstract. This paper solves a motion planning problem from a motion safety perspective, where a variant of the classical Rapidly exploring Random Tree (RRT) approach [1] called p-safe RRT is proposed. The exploration of the search space is similar to RRT, however, the highlight of p-safe RRT is the integration of passive motion safety. The basic principle of this safety level is to guarantee that the system can brake down and stop before collision. P-safe RRT extends a tree through the state time space, where tree's nodes and primitives are checked for passive motion safety. The computed trajectory is passively safe and drives the robot from its initial state to the goal state. The developed algorithms have been tested in simulation scenarios; featuring both fixed and moving objects with unknown trajectories for a car-like robot with a limited field of view.

Keywords: mobile robotics; motion planning; motion safety; dynamic environments.

1 INTRODUCTION

Few years ago, Autonomous Ground Vehicles (AGVs) were a challenge, but, today it becomes a reality: many car dealers currently work on AGVs projects (e.g. BMW, Toyota, Renault, Volkswagen, Peugeot, General Motors, Mercedes-Benz, or Tesla). Several demonstrations have been carried out on real roads, i.e. in the presence of other cars and pedestrians (both with unknown behaviour), having only a partial knowledge about the environment. However, the risk of accident remains present (see [2]). Even if these systems are autonomous, up to this day, cars are never unmanned [3]; will the driver seat ever be empty? Therefore, guaranteeing motion safety remains an open problem in such situations.

There is a rich literature on motion planning, most of approaches compute a compete path from an initial position of the robot to the goal position based on the environment representation (which is usually a priori known or built during a first phase of exploration) [4, 5, 6, 7, 8, 9]. Generally, the environment is considered as static. These methods can be adapted to dynamic environments but adding the temporal dimension increases the complexity of the problem. Other methods as PRM (*Probabilistic Roadmap*) [10] and RRT [1, 11] have more performances in high dimensional configuration spaces. However, PRM requires a priori knowledge about the environment or an initial map constructed off-line. Otherwise, RRT is more suitable for unknown environments. Besides, the kinodynamic constraints of the system are explicit-

ly taken into account and the incremental tree construction features better the strong changes of the environment compared to PRM. Many variants of RRT have been developed over years. First works [11, 1] were very expensive in computing time as the tree is expanded over the entire workspace. More recent works have combined RRT with other methods like PRM to solve the problem [12, 13]. All these works are applied in static environments. Other extensions of RRT have been also proposed to drive the robot in dynamic environments; in [14], RRT has been extended to ERRT (*Extended RRT*), where environment's changes are featured by interleaving planning and execution phases. An *anytime RRT* version was proposed in [15]; at each time instance, a new plan is computed based on the environment information update. Another variant of RRT is *Multipartite RRT*, which combines ERRT and anytime RRT [16]. It is suitable for environments containing moving objects with a priori known behaviour. There are other extensions of RRT (ex. [17, 18, 19]), but, in most cases, the future behaviour of moving objects is not considered at all or assumed a priori known.

Generally, guaranteeing motion safety requires a priori knowledge about the overall trajectory of each moving object to ensure that a collision will never occur: that is what is called *absolute motion safety*. However, when the future behaviour of objects is unknown, this form of safety is impossible to guarantee [20]. Therefore, following the opinion of some authors [21], it is better to settle for weaker levels of motion safety: *it is better to guarantee less than to guarantee nothing*. In [22], the proposed solution is to guarantee that the robot will be at rest if a collision took place. This is what we called *passive motion safety.*

The contribution of this paper is a variant of the classical RRT method [1]; a new approach *p-safe RRT* is proposed. It is based on building an incremental tree in the state time space. The generated nodes and trajectory primitives are checked for passive motion safety based on a verification algorithm of passively safe states that is proposed in [22]. So, the planner computes a passively safe trajectory to the goal.

The paper is organized as follows. Section 2 explains the paradigm collision-free vs. passive motion safety. In Section 3, the *classical RRT* principle is first presented, then the proposed extension *p-safe RRT* is described in detail. Finally, the validation of the developed algorithms is illustrated in Section 4 thanks to a simulation implementation.

2 COLLISION-FREE VS. SAFETY OF MOTION

From a motion safety perspective, previous works (e.g. [23-29]) proposed approaches that guarantee collision-free motion. However, it is not always sufficient; collision occurs regardless the undertaken action. For example, in the case of a vehicle traveling at high speed, confronted to a wall, it can be in a situation where collision is inevitable given its dynamic constraints (it is not possible to avoid the wall or to stop before collision). In other words, the vehicle is placed in *an inevitable collision state*, i.e. whatever the future trajectory of the vehicle, collision occurs. Therefore, to guarantee motion safety, *inevitable collision states* should be avoided.

Given the perceptual limitations of the robot and the unknown future behaviour of the moving objects present in the environment, it is not possible to guarantee absolute motion safety (collision will never occur given a priori known model of the future). This is why, we settled for weaker level of motion safety, but stronger given the harsh constraints related to the robot and the environment. This level of safety is called *passive motion safety* (p-safety). It is passive in a sense that the robot takes its own responsibility to not to be harmful with respect to its surrounding environment, regardless if the other obstacles will collide with him. Passive safety guarantees that if a collision takes place, the robot will be at rest. From this strategy, the *braking inevitable collision states concept* (braking ICS) was proposed [22]. A braking ICS (denoted ICS^b) is a state for which whatever the future trajectory of the robot, a collision occurs before the robot is stopped. A state that is ICS^b-free is a p-safe state.

To verify if a state is ICS^b or not (by duality p-safe or not), a braking ICS checker (called ICS^b-CHECK) was developed in [22]. The developed planner presented in section 3 is based on ICS^b-CHECK to check the p-safety of the planned trajectory.

3 THE DEVELOPED MOTION PLANNING TECHNIQUE

This paper proposes a new motion planning approach called *p-safe RRT*. It is inspired from RRT method [1]. However, unlike RRT, *p-safe RRT* has to guarantee passive motion safety criterion and takes into account the field-of-view limits and occlusions besides to the future behaviour of moving objects.

3.1 The classical RRT

RRT is a diffusion technique based on probabilistic sampling of the search space [11, 1]. It has been used to solve motion planning problems in high dimensional configuration spaces. A tree is incrementally expended through the space (set of nodes related by primitives) for planning a path from an initial state s_0 to a goal state s_g. The basic principle of this technique is explained in the algorithm presented in figure 1.

First, the tree (noted TREE) is initialized by the initial state of the robot. It represents the root node η_{root}. The function TREE_EXPANSION grows the tree from η_{root} toward randomly selected nodes (using the function RANDOM_TARGET): when a random target η_{target} is generated, the function NEAREST_NEIGHBOR selects the nearest node $\eta_{nearest}$ in the tree relatively to η_{target}. $\eta_{nearest}$ is then extended toward η_{target} by applying a control $u_{nearest}$ that is generally selected from a set of possible controls U_{ctrl} (thanks to NEAREST_CTRL). This selection is based on a minimized Euclidean distance between η_{neare} and η_{target}. The resulting node is η_{new}; it is added to the tree with its corresponding new primitive δp_{new}. If the expansion of the tree is obstructed by an obstacle, η_{new} is not generated. This process is repeated until the state s_g is reached (i.e. when the distance between η_{new} and s_g is lower than a certain threshold).

Generally, RRT and its variants [1, 11, 12, 13] solve motion planning problem only in the configuration space or the state space, the time parameter is not considered at all. Furthermore, most of these works are applicable in static environments. Even if

RRT has been extended for dynamic environments [14-19], as the future behaviour of moving objects is not considered or a priori known, motion safety cannot be guaranteed.

To solve such issue, we propose a modified version of RRT; p-safe RRT. It extends a tree in the state time space and computes a p-safe trajectory that drives the robot to the goal in a safe manner.

RRT_CONSTRUCTION()
1. Initialization: TREE $= \{\eta_{root}\}$;
2. **While** s_g not reached **do**
3. $\eta_{target} \leftarrow$ RANDOM_TARGET();
4. TREE_EXPANSION(TREE, η_{target});
5. **end while**
6. **return** TREE;

TREE_EXPANSION(TREE, η_{target})
7. $\eta_{nearest} \leftarrow$ NEAREST_NEIGHBOR(TREE, η_{target});
8. $u_{nearest} \leftarrow$ NEAREST_CTRL($\eta_{nearest}, \eta_{target}, U_{ctrl}$);
9. $(\eta_{new}, \delta p_{new}) \leftarrow$GENERATE_PRIM($\eta_{nearest}, u_{nearest}$);
10. **if** $\eta_{new} \neq \emptyset$ **then**
11. TREE = TREE $\cup$ η_{new};
12. TREE = TREE $\cup$ δp_{new};
13. **end if**

Fig. 1. The classical RRT algorithm.

3.2 P-safe RRT

The developed technique *p-safe RRT* constructs a tree thanks to an exhaustive exploration of the state time space, i.e. using a fixed set of feasible controls U_{ctrl}. The expanded tree is checked for passive motion safety using the algorithm ICS^b-CHECK proposed in [22] based on a model of the future MF (this point is detailed in [22]). Besides ICS^b-CHECK, an important property should be established concerning the p-safety guarantee of a trajectory, it is expressed as follow:

Property 1:
A trajectory P is p-safe (i.e. not a braking ICS) if P is collision-free and the final state of P is p-safe.

For more details concerning the proof of this property, or other p-safety guarantee properties and definitions, the reader is referred to [30].

The *p-safe RRT* principle is explained through the algorithm presented in figure 2. As for RRT, p-safe RRT is initialized by the root node η_{root}. In addition, the algorithm has as inputs U_{ctrl} and MF. During the planning time, each node η_i in the tree is expanded thanks to the function PSAFE_TREE_EXPANSION according to a set of controls U_{ctrl}. Each control leads to the generation of a new primitive δp_{new} and a

new node η_{new} (i.e. the final state of δp_{new}) (via GENERATE_TRAJ_PRIM). δp_{new} is checked for p-safety based on ICS^b-CHECK (BRAKING_ICS_CHECK) and property 1. In the case no p-safe primitive is found for a given node η_i, a braking trajectory is generated to guarantee the motion safety (using GENERATE_BRAKING_TRAJ). At the end of the process, different p-safe trajectories are possible, but only one trajectory is selected. The function SELECT_BEST_TRAJ is responsible of this task using an optimization function f which is based on the Euclidean distance (Δd_{traj}) between the trajectory final state s_f and the goal state s_g and the time cost of the trajectory (ΔT_{traj}). Weighting factors w_d and w_t are respectively associated to Δd_{traj} and ΔT_{traj}. f is expressed as follows:

$$f = w_d \Delta d_{traj} + w_t \Delta T_{traj} \qquad (1)$$

PSAFE_RRT_CONSTRUCTION()
1. Initialization: TREE $= \{\eta_{root}\}$, U_{ctrl}, MF;
2. **While** $t < t_{plan}$ **do**
3. **for** $i = 0$ to m **do**
4. PSAFE_TREE_EXPANSION(η_i, U_{ctrl});
5. **if** $\eta_{new} = \emptyset$ **then**
6. GENERATE_BRAKING_TRAJ();
7. **end if**
8. **end for**
9. **end while**
10. **return** TREE;
11. SELECT_BEST_TRAJ(TREE, s_g);

PSAFE_TREE_EXPANSION(η_i, U_{ctrl})
12. **for** $u_j \in U_{ctrl}$ **do**
13. $(\eta_{new}, \delta p_{new}) \leftarrow$GENERATE_TRAJ_PRIM($\eta_i, u_j$);
14. **if** BRAKING_ICS_CHECK(η_{new},MF)=False **then**
15. **if** δp_{new} is collision-free **then**
16. TREE = TREE ∪ η_{new};
17. TREE = TREE ∪ δp_{new};
18. **end if**
19. **end if**
20. **end for**

Fig. 2. P-safe RRT algorithm.

This algorithm illustrates the general principle of p-safe RRT technique, however, for motion safety reasons, the planning process can be interleaved with the execution process like in [14] or instead a partial motion planning can be applied like in [30, 31], where a new plan is periodically computed until the robot reaches its goal.

4 RESULTS

In order to illustrate *p-safe RRT* performances and demonstrate its motion safety guarantee, *p-safe RRT* algorithm has been implemented in simulation scenarios that feature fixed and moving objects with arbitrary trajectories. The application has been developed in C++ and it has been tested on a laptop equipped with an Intel Core i7 (1.6GHz, 4GB RAM, OS: Linux).

4.1 The system's dynamic model

P-safe RRT has been tested for a car-like robot that operates in a state time space. The robot state is a 5 tuple $s = (x, y, \theta, v, \xi)$. (x, y) are the Cartesian coordinates (related to the midpoint of the rear axle of the robot), θ is the robot's orientation, v its linear velocity and ξ its steering angle. The dynamic of the robot is governed by the following differential equations:

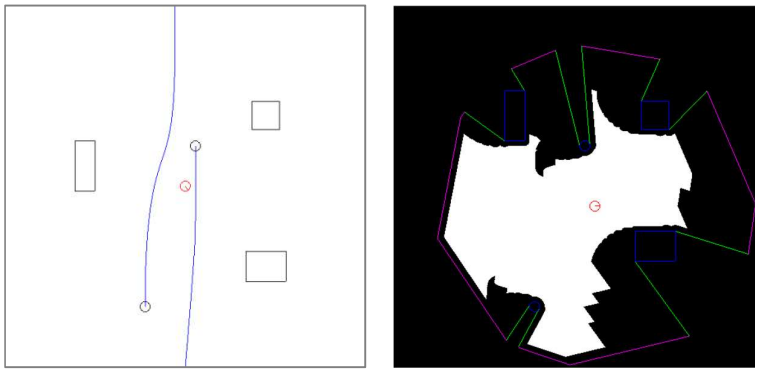
$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ v\tan\xi/L \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u_\alpha + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_\xi \qquad (2)$$

Where L denotes the robot's wheelbase, u_α is the linear acceleration and u_ξ is the steering angle velocity. These last two parameters represent the control inputs that should be applied to the system. The robot behaviour is limited by kinodynamic constraints that should be respected, namely:
$|v| \leq v_{max}, |\xi| \leq \xi_{max}, |u_\alpha| \leq u_{\alpha_{max}}, |u_\xi| \leq u_{\xi_{max}}$, where $v_{ma} = 20m/s, \xi_{max} = 0.314rad, u_{\alpha max} = 7m/s^2, u_{\xi_{max}} = 0.314rad/s.$

4.2 P-safe RRT

To illustrate how the developed technique works, it has been tested in a scenario called simple urban scenario (see figure 3.a); it contains fixed and moving objects with an arbitrary behaviour. In this case, no information concerning the future behaviour of the moving objects is available, only the maximum velocity is considered ($v_{B_{max}} = 20m/s$). The robot has a limited field-of-view with maximum radius of $80m$ and it moves in a 2D workspace with $180 \times 180\ m$ dimensions. The radius of the disk objects is $2.5m$. *P-safe RRT* has to drive the robot starting from initial position until a predefined goal while considering the passive motion safety guarantee. Both seen and unseen objects (perception limits and occlusions) are considered. For safety purpose, *p-safe RRT* computes at first ICS^b set (see figure 3.b), the black region represents the forbidden part of the environment that should be avoided by the robot.

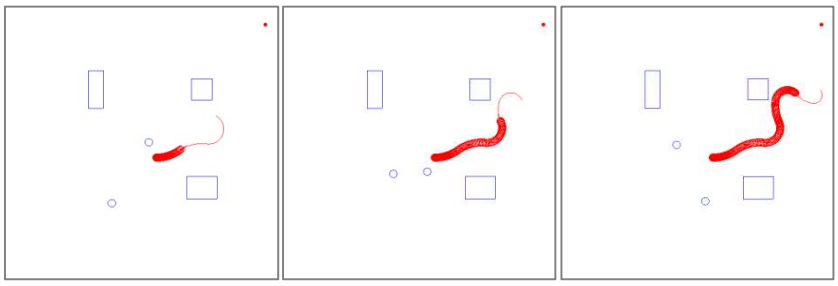
The tree is expanded in the space following a set of controls (u_α, u_ξ) selected as follows: u_α=$u_{\alpha_{max}}$, i.e. a constant maximum linear acceleration is considered and $u_\xi \in [-u_{\xi_{max}}, u_{\xi_{max}}]$, i.e. a constant steering angle velocity is selected in this interval of values.



(a) (b)

Fig. 3. (a) Simple urban scenario; polygons are the fixed objects and black discs are the moving objects with their corresponding trajectories represented in blue. The robot is the red disc at the center. (b) ICS^b set for a given position of the robot; the forbidden states (ICS^b) are represented in black while the p-safe states are represented in white (see [22] for more details). The robot's field-of-view is represented by the magenta segments (perception limits) and the green segments (occlusions).

Figure 4 illustrates the behaviour of the robot further to the implementation of *p-safe* RRT in the simple urban scenario. The executed trajectory is represented by the thick trace behind the system while the trajectory in front of the robot is the planned part. Based on the ICS^b set computed given an updated model of the future, the robot moves until it reaches the goal while avoiding seen and unseen objects (the limits of the field of view). For example, in the second snapshot, the robot escaped an ICS^b region resulting from perception limits (unexpected objects). In the third snapshot, it escaped a fixed object and an unexpected objects region. The built p-safe RRT trees corresponding to these two snapshots are respectively illustrated in figure 5.a and figure 5.b. Each tree's primitive and node is checked for passive safety. Only p-safe nodes are expandable. For an ICS^b node, a collision-free braking trajectory is generated. This experiment shows the performance of *p-safe RRT* to guarantee a passively safe behaviour for the robot. Even the avoidance is not possible, the robot brakes down to remain in a p-safe state.



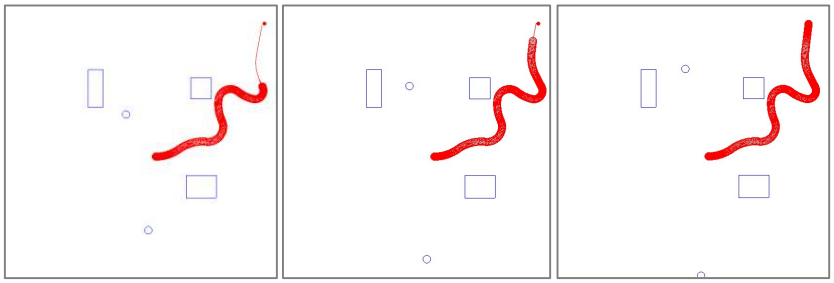
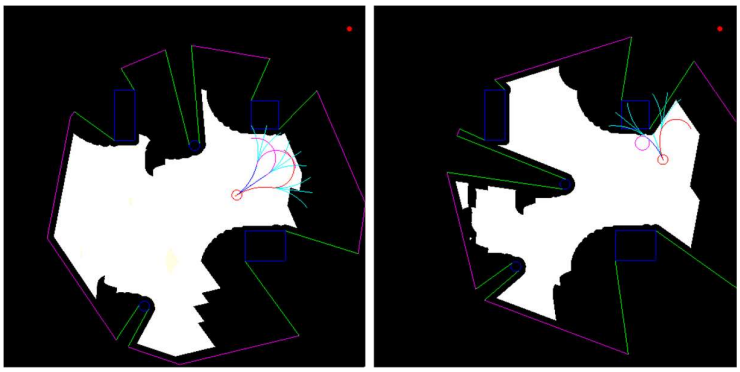


Fig. 4. Snapshots of p-safe RRT at work in the simple urban scenario (the goal to reach is represented by the red point) (see text).



(a) (b)

Fig. 5. P-safe RRT trees corresponding to the second (a) and third snapshot (b) of figure 4. Tree's primitives represented in blue are p-safe, those in cyan are ICS^b (not p-safe) and those in magenta characterize collision-free braking trajectories (for each ICS^b node, a braking trajectory is generated). The selected trajectory (to be executed) is represented in red.

5 CONCLUSION

AGVs navigation in real road is still challenging up to now. In this paper, we proposed a solution by developing a passively safe motion planner; *p-safe RRT*. It is a variant of the classical RRT technique that is based on the guarantee of a passively safe motion. This navigation system takes into account the dynamics of both the robot and the environment, where an unknown model of the future is considered. *P-safe RRT* has been validated through simulation tests; the robot showed a passively safe behaviour by avoiding perceived objects and unexpected objects that are potentially dangerous from a motion safety point of view.

A logical future work to this one is the implementation of this planning system on an experimental platform. On the other hand, concerning the approach concept, the tree expansion strategy used in this paper is basic. The exploration of the space time could be improved by means of other existing expansion techniques or strategies.

REFERENCES

1. S. M. LaValle and J. J. Kuffner, “Randomized kinodynamic planning”, International Journal of Robotics Research, V. 20, Issue 5, pages 378–400, 2001.
2. L. Fletcher, S. Teller, E. Olson, D. Moore, Y. Kuwata, J. How, J. Leonard, I. Miller, M. Campbell, D. Huttenlocher, A. Nathan and F.R. Kline, “The MIT—Cornell collision and why it happened”, International Journal of Field Robotics, V. 25, Issue 10, pages 775-807, 2008.
3. S. Thrun, “What we’re driving”, at Google official blog, (9 October 2010).
4. J.-C. Latombe, “Robot Motion Planning”, Livre, Kluwer, Boston, MA, 1991.
5. H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki and S. Thrun, “Principles of Robot Motion: Theory, Algorithms, and Implementation”, Livre, MIT Press, 2005.
6. S. LaValle, “Planning Algorithms”, Livre. Cambridge University Press, 2006.
7. J.P. Laumond, “Robot Motion Planning and Control”, Livre, ISBN 978-3-540-76219-5, 1998.
8. R. Siegwart, I. Nourbakhsh and D. Scaramuzza, “Introduction to Autonomous Mobile Robots”, Livre, The MIT Press, 2004.
9. S. S. Ge and F. L. Lewis, “Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications”, Livre, SBN 9780849337482, 2006.
10. L. E. Kavraki, P. Svestka, J. C. Latombe and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”, IEEE Transactions on Robotics and Automation, V. 12, Issue 4, pages 566–580, June 1996.
11. S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning”, Technical Report (Computer Science Department, Iowa State University) (TR 98-11), October 1998.
12. K. E. Bekris, B. Chen, A. Ladd, E. Plaku and L. Kavraki, “Multiple query motion planning using single query primitives”, In IEEE/RSJ International Conference on Intelligent Robots and Systems, V. 1, pages 656–661, 2003.
13. M. Akinc, K.E. Bekris, B.Y. Chen, A.M. Ladd, E. Plaku, L.E. Kavraki, “Probabilistic Roadmaps of Trees for Parallel Computation of Multiple Query Roadmaps”, The International Symposium on Robotics Research, V.15, pages 80-89, 2005.
14. J. Bruce and M. Veloso, “Real-time randomized path planning for robot navigation”, In RoboCup 2002: Robot Soccer World Cup VI, Lecture Notes in Computer Science, V.2752, pages 288–295, 2003.
15. D. Ferguson and A. Stentz, “Anytime RRTs”, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 5369-5375, 2006.
16. M. Zucker, J. Kuffner and M. Branicky, “Multipartite RRTs for Rapid Replanning in Dynamic Environments”, IEEE International Conference on Robotics and Automation, pages 1603 – 1609, 2007.
17. 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, pages 4375 – 4380, 2006.
18. L. Heon-Cheol, Y. Touahmi and L. Beom-Hee, “Grafting: A Path Replanning Technique for Rapidly-Exploring Random Trees in Dynamic Environments”, Advanced Robotics, V.26, Issue 18, pages 2145-2168, 2012.
19. L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma, and N. Zheng, "Efficient Sampling-Based Motion Planning for On-Road Autonomous Driving”, IEEE Transactions on Intelligent Transportation Systems, V.16, Issue 4, pages 1961-1976, 2015.
20. T. Fraichard, “Will the driver seat ever be empty?” INRIA” Research Report, 2014.

21. K. Macek, D.A. Vasquez-Govea, Th. Fraichard and R. Siegwart, “Towards safe vehicle navigation in dynamic urban scenarios”, Automatika, V. 50, no. 3-4, pages 184-194, 2009.
22. S. Bouraine, T. Fraichard, and H. Salhi, “Provably safe navigation for mobile robots with limited field-of-views in dynamic environments,” Autonomous Robots, vol. 32, no. 3, pp. 267–283, 2012.
23. L. Pallottino, V. Scordio, A. Bicchi and E. Frazzoli, “Decentralized cooperative policy for conflict resolution in multivehicle systems”, IEEE Transations on Robotics, 23(6), 2007.
24. J. Van den Berg, M. Lin and D. Manocha ,”Reciprocal velocity obstacles for real-time multi-agent navigation”, In *IEEE* International Conference on Robotics and Automation, 2008.
25. E. Molinos, A. Llamazares, M. Ocana and F. Herranz, “Dynamic obstacle avoidance based on curvature arcs”, IEEE/SICE International Symposium on System Integration (SII), pages 186-191, 2014.
26. E. Lalish and K. Morgansen, “Decentralized reactive collision avoidance for multivehicle systems”, In IEEE conf. decision and control, Cancun, 2008.
27. K. Bekris, K.Tsianos and L. Kavraki, “Safe and distributed kinodynamic replanning for vehicular networks”, Mobile Networks and Applications, 14(3), 2009.
28. B. Gopalakrishnan, A.K. Singh and K.M. Krishna, “Time scaled collision cone based trajectory optimization approach for reactive planning in dynamic environments”, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014), pages 4169-4176, 2014.
29. J. L. Blanco, J. González and J. A. Fernández-Madrigal, “Extending obstacle avoidance methods through multiple parameter-space transformations”, Autonomous Robots, V. 24, Issue 1, pages 29-48, 2008.
30. S. Bouraine, T. Fraichard, and O. Azouaoui and H. Salhi, “Passively Safe Partial Motion Planning for Mobile Robots with Limited Field-of-Views in Unknown Dynamic Environment”, 2014 IEEE International Conference on Robotics and Automation (ICRA 2014), pages 3576-3582, 2014.
31. S. Bouraine, “contrıbutıon a la planıfıcatıon de mouvements en envıronnements dynamıques pour des robots mobıles de type voıture : cas du robucar”, PhD thesis, 2016.