Robotica (2013) volume 31, pp. 905–921. © Cambridge University Press 2013
doi:10.1017/S0263574713000131

SLAM-based maneuverability strategy for unmanned car-like vehicles

Fernando A. Auat Cheein*

Department of Electronics Engineering, Universidad Técnica Federico Santa María, Av. España 1680, Valparaíso, Chile

(Accepted January 22, 2013. First published online: March 7, 2013)

SUMMARY

In this work, an optimal maneuverability strategy for car-like unmanned vehicles operating in restricted environments is presented. The maneuverability strategy is based on a path planning algorithm that uses the environment information to plan a safe, feasible and optimum path for the unmanned mobile robot. The environment information is obtained by means of a simultaneous localization and mapping (SLAM) algorithm. The SLAM algorithm uses the sensors' information to build a map of the surrounding environment. A Monte Carlo sampling technique is used to find an optimal and safe path within the environment based on the SLAM information. The objective of the planning is to safely reach a desired orientation in a bounded space. Theoretical demonstrations and real-time experimental results (in indoor and outdoor environments) are also presented in this work.

KEYWORDS: Path planning; SLAM; Unmanned vehicles.

1. Introduction

This work addresses the problem of a safe and optimal path planning technique using the available surrounding environment information. The information is obtained by means of a simultaneous localization and mapping (SLAM) algorithm. A range laser sensor implemented on the vehicle extracts raw data from the environment which is then used to create a geometric model of such environment.

The SLAM algorithm concurrently estimates the pose – position and orientation – of a vehicle and the features of the environment in which the vehicle is located while minimizing estimation errors.[1–3] The set of features acquired from the environment constitutes a map (also known as *feature-based* map).[3] The nature of the map is intrinsically related with the kind of feature extracted and the sensor used.[4,5] Thus, a geometric-based map (such as the ones presented in refs. [1, 2, 6, 7]) models the environment as a set of geometric primitives. The work[8] shows a SLAM algorithm that extracts lines from the environment using a range laser sensor. With the same insight, refs. [9] and [10] show a SLAM algorithm that also models the environment with lines but using a range sonar sensor. Thus, a same feature can be acquired using different sensors. The work[7] models the trees from the environment as point-based features using a laser, whereas refs. [11] and [13] also extract points from the environment but using artificial vision. Although a same feature can be extracted using different sensors or extraction procedures, some map information remains dependent on the nature of the sensor used. For example, shape or texture information can be extracted by a vision system and not by range sensors.

Other kinds of maps are topological maps.[13] Topological maps do not concern the metric nor the geometry of features from the environment. Instead, topological maps extract patterns or labels that allow unique environment classification.

The SLAM algorithm for feature-based maps can be implemented on several estimation filters with their respective restrictions. As a first approach, the extended Kalman filter (EKF) uses linearized models of the process – i.e. vehicle's motion – and the features.[1–4,14] The EKF-SLAM also considers the robot and the features from the environment as Gaussian random variables. The extended information filter (EIF) and the unscented Kalman filter (UKF) are other Gaussian approaches to SLAM. The EIF[3,15] shows a linear processing time on its correction stage whereas on the EKF, the processing time is quadratic. On the contrary, the UKF shows a better performance when dealing with the non-linearities of the process and the measurement.[3,16] Non-parametric filters are also implemented to solve the SLAM problem. Thus, the particle filter (PF) and Markov chains do not have Gaussian restrictions nor non-linearity issues.[17,18]

Path planning algorithms provide to the mobile robot the coordinates of a plausible and free-collision path within the environment of navigation. Two main categories of path planning can be found in the scientific literature: local planning and global path planning.[14,19,20] The local planning is related to the maneuverability strategy from a local reference frame attached to the mobile robot motion. In some cases, reactive navigation (such as sensor-based navigation) is used instead of planning[4,19] because of the small amount of environment information needed by the mobile robot. An example of this is ref. [21], where the robot instead of planning uses range sensor laser information to instantaneously avoid obstacles within the environment. On the contrary, refs. [22] and [23] are two examples of local planning. In these works, a navigation objective is determined within the local point of view of the mobile robot and a trajectory controller drives the vehicle to the objective of navigation. The information of the environment used in the local path planning strategy is restricted to the sensor measurements referred to the current pose of the mobile robot.

* Corresponding author. E-mail: fernando.auat@usm.cl

https://doi.org/10.1017/S0263574713000131 Published online by Cambridge University Press

On the contrary, a global map is not attached to the mobile robot's pose. It is usually a fixed point of the environment to which all information should be referred to.[19] A global path planning strategy[24] uses the information provided by the global map to generate a plausible path between the vehicle and the objective of navigation based on the planning criterion. Different optimization criteria have been proposed by the scientific literature (see refs. [14] and [24]). For example, Pothal et al.[25] find minimal length paths between the mobile agent and the navigation objective; Tian et al.[26] find minimal energy paths. On the other hand, Becked et al.[27] propose a stronger condition for the optimal path; it should be kinematically compatible with the mobile robot and also be the shortest path between all possible paths.

Once the optimization criterion has been established, a search algorithm finds the best path among all the possibilities. For example, the Dijkstra algorithm[28] is widely used by the scientific community to find minimum cost paths;[24] also the $A*$,[29] the Graphs method[30] and grid-based techniques[3] are examples of global path planning methods (a complete description of global planning techniques can be found in ref. [24]). A common situation of the global path planning methods is that they use the environment information to generate the appropriate path but the way the information is acquired is usually not considered. Thus, the mobile robot follows a path generated by a path planning algorithm that uses prior map information[3] which is not necessarily related to the sensors available on the vehicle. Thus, for example, the Voronoi method[4] is not appropriate if the mobile robot has ultrasonic sensors to acquire range measurements because of the dispersion of the ultrasonic readings; the graph method requires a prior precise knowledge of the map of the environment. In order to use the environment information, a mapping process should also be included within the planning strategy.

In the case of path planning algorithms for car-like mobile robots, the path should be kinematically compatible with the robot's restrictions. For example, Dubins' results[31,32] present an optimal solution for car-like path planning in open spaces. Therefore, no environment restrictions are taken into account.[33] In addition, Dubins' results do not consider variable radius of turning of the mobile robot; Lee et al.[34] show a kinematic compatible path planning for car-like robots using clothoid curves. Although this work shows feasible planned paths, the method is restricted to the set of clothoids considered by the algorithm. Furthermore, the surrounding environment is assumed to be known. In ref. [35], the authors propose a path planning criterion based on a limited number of possible arcs of turning with an also limited length. Thus, for example, the authors consider that the robot will be able to drive straight forward, to the right or to the left – following a predefined radius of turning – no more than a previously specified distance, e.g. 1 m for each case. Then, the authors claim that the proposed technique will allow the robot to reach any point of the navigable space by successively generating arcs. With the same insight as the previous works, Suzuki et al.[35] use static map information previously loaded to the algorithm. Therefore, no sensor information is given to the path planning technique. In ref. [36] a probabilistic path planning based on a simulated and previously known environment is presented. The path planning technique is based on generating successive segments until reaching the desired location within the environment. On the other hand, Lee and Chen[37] consider a grid-based path planning where the robot is modeled as a rigid body. Also, an $A*$ algorithm is used to find the optimal path. In ref. [37], the environment is also previously known. As can be seen, in path planning algorithms no considerations are taken regarding the sensors of the robot and the surrounding environment. Furthermore, in the previous works, it is assumed that the environment is previously available for planning purposes.

The main contribution of this work is a car-like path planning technique, kinematically compatible with the vehicle and the restricted environment. The planning objective is to safely – and collision free under moderate uncertainty in the localization – reach a desired orientation within the environment. The path planning algorithm uses the environment information acquired by range laser data to model the environment – by means of a SLAM algorithm – using lines and corners. That information is then managed to generated a free-collision path by predicting collision free neighborhoods. Also, the proposed path planning algorithm guarantees a solution based on the SLAM-based map information. In addition, the path planning technique finds an optimal path according to a previously defined cost criterion.

In this work, a path planning strategy for car-like mobile robots operating in confined spaces is developed. The path is composed of semi-circle arcs whose lengths are limited by walls or obstacles of the environment. A Monte Carlo sampling technique is used to find the radius of these arcs in order to obtain the optimal path according to a previously defined cost function. A consistent SLAM algorithm is implemented to extract a map from the environment based on range laser sensor measurements, and to localize the vehicle. The map that the SLAM algorithm provides is used in the path planning considering the SLAM information (SLAM system state and its associated covariance matrix). Although this work uses an EKF-SLAM algorithm that manages extracted corners and lines from the environment, the path planning strategy presented herein does not depend on the type of SLAM algorithm used (as long as the SLAM algorithm allows a geometric-based reconstruction of the environment). The objective of the path planning is to turn the car-like mobile robot until a desired heading is attained in a safe manner. The last is achieved by using the SLAM information to determine and predict collision-free regions of the workspace. Then, a car-like trajectory controller uses the generated path to drive the robot to the desired heading. In this work, we show that the path planning technique for car-like robots proposed is optimal, kinematically compatible with the mobile robot model and that a solution always exists for any desired orientation. Many experiments and statistical analysis are performed herein to show the functionality of the proposed algorithm.

2. System Architecture

Figure 1 shows the general system architecture used in this work. Briefly, the system architecture can be

https://doi.org/10.1017/S0263574713000131 Published online by Cambridge University Press

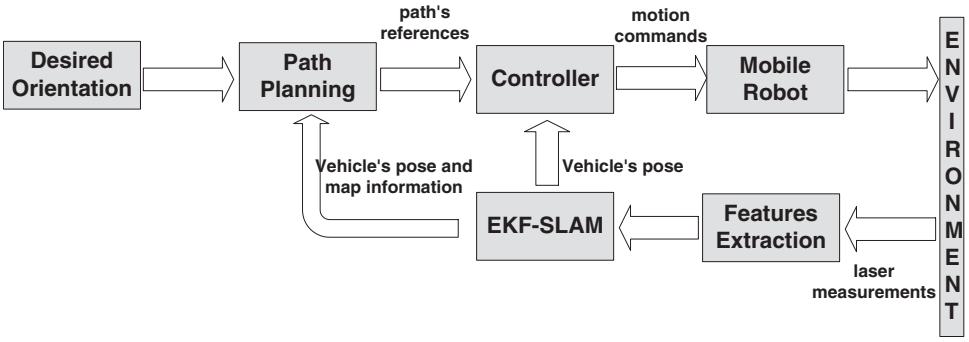


Fig. 1. General system architecture of the planning strategy presented in this work.

explained as follows. Once the robot begins its navigation, it remains navigating within the environment while the SLAM algorithm is performed at the same time, acquiring information from the environment and estimating the pose of the robot. If the maneuverability strategy is invoked, then the path planning algorithm uses the SLAM system state information to generate an optimal and collision-free path. Once the path is generated, a trajectory following controller is be able to drive the robot through the determined path until the vehicle reaches the desired angle.

The motion controller used in this work is a path following controller for car-like vehicles presented in ref. [38]. We also consider that the desired angle of turning is previously defined.

Following, each block of Fig. 1 will be introduced.

2.1. *Mobile robot model*

The discrete kinematic model of the mobile robot is shown in Eq. (1), where $[\xi_{x_t}\ \xi_{y_t}\ \xi_{\theta_t}]^T$ define the pose of the robot within the environment; V_t is the linear velocity imposed to the vehicle and ψ_t is the wheel-steering command. Both, the linear velocity and the wheel-steering, compose the motion control vector of the system: $u_t = [V_t\ \psi_t]^T$; L is the distance between the traction wheels and the steering wheel. Δt is the sampling time of the robot. In this work, we assumed $\Delta t = 0.2$ s – the sub-index t represents time. Figure 2 shows a graphical representation of the robot. In Fig. 2, R is the minimum radius of turning.[4] For the robot used in this work (shown in Fig. 2), $R = 1$ m:

$$\begin{pmatrix} \xi_{x_{t+1}} \\ \xi_{y_{t+1}} \\ \xi_{\theta_{t+1}} \end{pmatrix} = \begin{pmatrix} \xi_{x_t} \\ \xi_{y_t} \\ \xi_{\theta_t} \end{pmatrix} + \Delta t \begin{pmatrix} V_t \cos(\xi_{\theta_t}) \\ V_t \sin(\xi_{\theta_t}) \\ \frac{V_t}{L} \tan(\psi_t) \end{pmatrix} \tag{1}$$

2.2. *Feature extraction*

In this work, the features extracted from the environment – i.e. the environment modeling – correspond to lines, associated with walls and corners (convex and concave). Both features are extracted using raw laser data. The extraction of lines is based on a recursive mean-squares algorithm, previously presented in refs. [8, 39]. The corners from the environment are extracted by a recursive clustering algorithm applied on the raw laser data.[23,39] The models of the features of the environment – corners and lines – are shown in Eqs. (2) and (3). Figure 3 shows the graphical interpretation

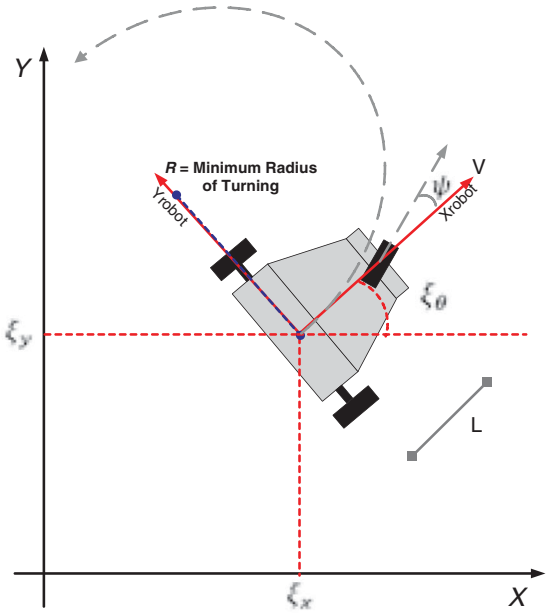


Fig. 2. (Colour online) Graphic representation of the car-like robot.

of the variables in Eqs. (2) and (3).

$$\begin{aligned} z_{corner_t} &= \begin{bmatrix} Z_{R_t} \\ Z_{\beta_t} \end{bmatrix} \\ &= \begin{bmatrix} \sqrt{(\xi_{x_t} - x_{corner_t})^2 + (\xi_{y_t} - y_{corner_t})^2} \\ \arctan \dfrac{\xi_{y_t}(k) - y_{corner_t}}{\xi_{x_t} - x_{corner_t}} - \xi_{\theta_t} \end{bmatrix} \\ &\quad + \begin{bmatrix} w_{R_t} \\ w_{\beta_t} \end{bmatrix} \end{aligned} \tag{2}$$

$$\begin{aligned} z_{line_t} &= \begin{bmatrix} Z_{\rho_t} \\ Z_{\alpha_t} \end{bmatrix} \\ &= \begin{bmatrix} \rho_t - \xi_{x_t}(k)\cos(\alpha_t) - \xi_{y_t}(k)\sin(\alpha_t) \\ \alpha_t - \xi_{\theta_t} \end{bmatrix} \\ &\quad + \begin{bmatrix} w_{\rho_t} \\ w_{\alpha_t} \end{bmatrix} \end{aligned} \tag{3}$$

In Eqs. (2) and (3), w_{ρ_t}, w_{α_t}, w_{R_t}, w_{β_t} are Gaussian noises associated with the measurement process.

https://doi.org/10.1017/S0263574713000131 Published online by Cambridge University Press

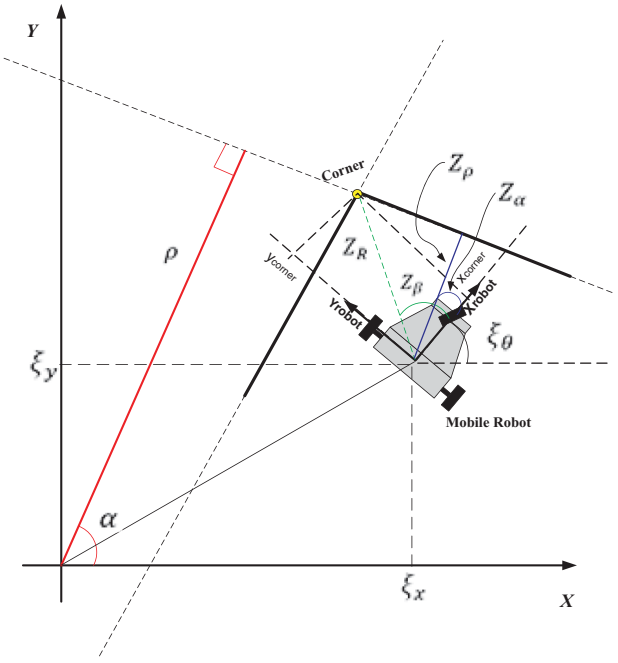


Fig. 3. (Colour online) Graphic representation of the features acquired from the environment.

In Eq. (2), x_{corner_t} and y_{corner_t} are the Cartesian coordinates of the corner from a local reference frame attached to the mobile robot (see Fig. 3).

2.3. *Simultaneous localization and mapping algorithm*

The SLAM algorithm implemented in this work is a sequential EKF feature-based SLAM.[39] The SLAM system state is composed of the vehicle estimated pose and the features extracted from the environment. The order in which features are added to the SLAM system state depends on the moment in which the feature – either a corner or a line – is detected. For visualization and map reconstruction purposes, a secondary map is maintained. This secondary map stores the beginning and ending points of the segments associated with the lines of the environment. Thus, the secondary map allows finite walls' representation. The secondary map is updated according to the feature correction in the EKF-SLAM system state, and if a new feature is added to that system state, it is also added in the secondary map (see ref. [23]). Equations (4) and (5) show the system state structure and its covariance matrix. All elements of the SLAM system state are referenced to a global coordinate system (see Fig. 3):

$$\hat{\xi}_{t|t} = \begin{bmatrix} \hat{\xi}_{v_t} \\ \hat{\xi}_{m_t} \end{bmatrix}, \tag{4}$$

$$\Sigma_t = \begin{bmatrix} \Sigma_{v,v_t} & \Sigma_{v,m_t} \\ \Sigma_{m,v_t} & \Sigma_{m,m_t} \end{bmatrix}. \tag{5}$$

In Eq. (4), $\hat{\xi}_t$ is the system state estimate; $\hat{\xi}_{v_t} = [\hat{\xi}_{v,x_t} \ \hat{\xi}_{v,y_t} \ \hat{\xi}_{v,\theta_t}]^T$ is the estimated pose of the vehicle; $\hat{\xi}_{m_t}$ represents the map of the environment and it is composed of the Cartesian coordinates that define corners and the polar coordinates that define lines in the environment (all of them referenced to the global reference system). The order in which lines and corners appear in $\hat{\xi}_{m_t}$ depends on the moment they are detected. Σ_t is the covariance matrix associated with the SLAM system state, Σ_{v,v_t} is the covariance of the vehicle pose estimation and Σ_{m,m_t} is the covariance of the estimated features of the environment. The rest of the elements of Σ_t are cross-correlation matrices.

The covariance matrix initialization techniques and the EKF definition can be found in refs. [1–3]. A sequential EKF[40] was implemented in order to reduce computational costs. It is worth mentioning that although in this work an EKF-based SLAM is implemented, the proposed methodology is not restricted to it and other SLAM approaches can be used instead.

2.4. *Path planning algorithm*

The planning strategy used in this work can be summarized as follows:

- The mobile robot follows a semi-circle path (forward or backward) until it reaches a desired orientation within the environment.
- The robot changes the direction of movement when it is near to a mapped element from the environment.
- The vehicle's radius of turning varies from its minimum radius of turning to infinity – a straight line.
- The first arc generation is based on the navigable space available regarding the robot's pose within the environment and the map built by the SLAM algorithm. One of the following situations may apply for the first arc: forward to left, forward to the right, backward to the left or backward to the right. Forward and backward are considered as *DIRECTION* in the algorithm of Fig. 4, whereas *to the left* or *to the right* are referred as *SIDE* in Fig. 4.

As a first approach, let us consider that the radius of turning of the robot remains constant during the execution of the proposed path planning strategy. The driving direction changes when the vehicle is close to an element of the environment. This process continues until the vehicle has successfully reached a desired angle. Considering that the vehicle follows a semi-circle trajectory, the final trajectory path is a car-like kinematic plausible path.[4,41] The turning strategy algorithm with constant radius of turning is shown in Fig. 4.

Once the turning algorithm starts, the initial pose of the robot and the map of the environment are stored and used in the planning process (lines of code 3–5 in Fig. 4). *ArcAngle* in lines of code (3), (10) and (15) represents the orientation that the robot would have at that precise position of the path. Thus, if an arc intersects an element from the environment – see line of code (11) – or if it reaches the desired orientation – line of code (15) – then *ArcAngle* represents the orientation that the robot would have at that position. In order to ensure that the vehicle will manoeuver only within the mapped area of the environment, the secondary map obtained by the SLAM (MAP in line 4) is circumscribed by a virtual rectangle. Thus, a new map, named *MAPv* in code line (6), composed of the elements of MAP and the virtual features of the circumscribing rectangle is used in the planning process. Following this, the system chooses which side and direction it

https://doi.org/10.1017/S0263574713000131 Published online by Cambridge University Press

1: Begin manoeuvrability strategy
2: Let Ψ be the desired angle
3: Let $ArcAngle = \hat{\xi}_{\theta_t}$
4: Let MAP be the map built by the SLAM algorithm
5: robot's pose $\hat{\xi}_{v_t} = [\hat{\xi}_{x_t}\ \hat{\xi}_{y_t}\ \hat{\xi}_{\theta_t}]^T$
6: Let $MAPv$ be the MAP built by the SLAM circumscribed between four virtual lines
7: $MAPv$ = CircumscribeMap(MAP)
8: [SIDE, DIRECTION] = FirstMovementProc($\hat{\xi}_{v_t}$, $MAPv$)
9: **for** $j = 1$ to ∞ **do**
10: [ARC, ArcAngle] = ArcGeneration($\hat{\xi}_{v_t}$, $MAPv$, SIDE, DIRECTION)
11: **if** ARC intersects with $MAPv$ **then**
12: $SIDE = \overline{SIDE}$
13: $DIRECTION = \overline{DIRECTION}$
14: **end if**
15: **if** $ArcAngle \approx \Psi$ **then**
16: break
17: **end if**
18: **end for**

Fig. 4. Trajectory generation algorithm. The overline implies the Boolean complement operator.

will generate the first semi-circle trajectory. This decision is made according to the *FirstMovementProc* function which uses the robot's current pose and the *MAPv* information in order to decide (see Section 2.4.2). Regarding the map information, the side of the turning and the direction, a semi-circle trajectory is generated (line 10 in Fig. 4). The center of the semi-circle remains at the y-axis of the mobile robot (see Fig. 2). Thus, if, e.g., the robot has to turn to its left, then the center of the semi-circle will be the point $[0, R]^T$ at the local reference frame attached to the robot, where R is the minimum radius of turning (see Fig. 2).

The semi-circle trajectory is calculated from a local reference frame attached to the mobile robot. Although the trajectory is generated from the point of control of the mobile robot, the neighborhood to the closest feature in the map is determined by considering the robot as a rigid body (see refs. [41, 42]. Then, if any vertex or edge of the mobile robot rigid body model is in the neighborhood of a feature (virtual or real from *MAPv*), the semi-circle trajectory stops its planning and a new trajectory is generated with both the direction and the side of turning opposite ($\overline{SIDE}$ and $\overline{DIRECTION}$) to their previous values. Thus, if the previous direction was reverse, then the next direction will be forward. The *for* loop – in the algorithm of Fig. 4 – repeats until the path has reached the desired orientation. If during the path planning procedure, the path reaches the desired orientation, the algorithm exits (line 16). Figure 5 shows an example of applying the algorithm shown in Fig. 4 within a real environment.

In Fig. 5, magenta points represent raw laser data; solid black segments are associated with walls from the environment and green circles are detected corners. The robot's pose is $\hat{\xi}_{v_t} = [2.5 \quad 12.5 \quad 2.1]^T$. The desired

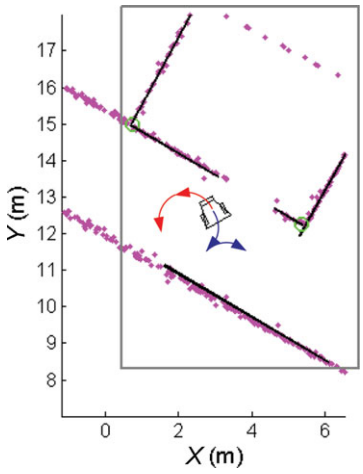
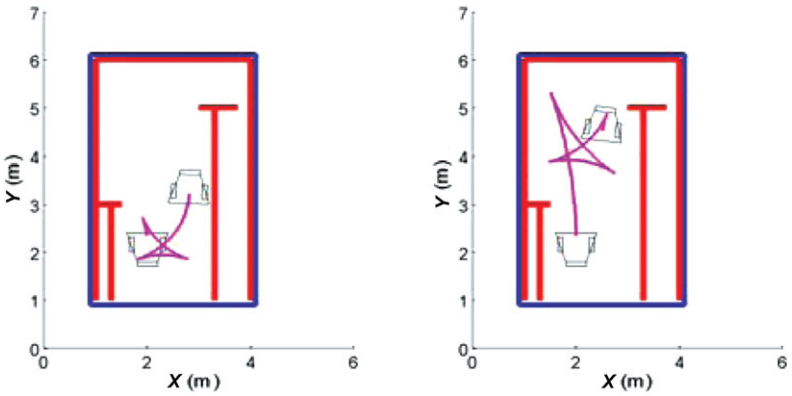


Fig. 5. (Colour online) Example of path planning with fixed radius of turning.

destination angle is $\Psi = -0.3$ rad. As can be seen, if the robot tries to turn following a circle with its minimum radius of turning (solid red arrows in Fig. 5), it will probably crash against the walls of the environment. On the other hand, by applying the algorithm shown in Fig. 4, the robot is able to generate a feasible path that will lead it to reach the desired angle (solid blue arrows in Fig. 5). The solid grey segments in Fig. 5 represent the virtual rectangle that circumscribes the map obtained from the environment by the SLAM algorithm. The example shown in Fig. 5 shows that with two movements (the first one is backward and the second is forward) following the minimum radius of turning, the vehicle finds a collision-free solution to reach the desired orientation.

https://doi.org/10.1017/S0263574713000131 Published online by Cambridge University Press



(a) Fixed radius of turning (b) Variable radius of turning

Fig. 6. (Colour online) Examples of paths based on the maneuverability strategy presented in this paper. Panel (a) shows the path encountered by using the minimum radius of turning of the mobile robot; panel (b) shows the path encountered by using a variable radius.

The use of the information provided by the map of the SLAM algorithm within the planning strategy is crucial, given that the mobile robot does not have proximity or range sensors at the back of its chassis. Therefore, when the robot is moving backward, the pose of the vehicle and the map of the environment – both provided by the SLAM algorithm – are its only references for a free-collision motion planning. Thus, the SLAM algorithm should be consistent during the planning and motion process.[39]

2.4.1. Planning with variable radius: Monte Carlo method. The fact of using a fixed turning radius, as presented before, does not ensure that the robot will follow the most appropriate path according to the environment disposition – e.g. performing risky maneuvers, wasting energy, etc. Figure 6 shows this situation in a simulated environment. The vehicle's initial orientation is $\hat{\xi}_{\theta_t} = -\pi/2$; the map's features are represented by solid red segments (associated with walls); the blue segments are the virtual features that correspond to a rectangle that circumscribes the mapped environment. As stated before, the virtual features of the circumscribing rectangle are used to restrict the area of planning, thus avoiding open map situations; the desired angle is $\Psi = \pi/2$. The solid magenta line is the path planned by the maneuverability strategy presented in this work. Figure 6(a) shows the path generated by the system using only the minimum radius of turning of the mobile robot ($R = 1$ m). On the other hand, Fig. 6(b) shows the path generated by the system using a variable radius. As can be seen, the case shown in Fig. 6(a) finishes the path planning too near to a wall of the environment, compromising in that way future movements. On the contrary, Fig. 6(b) shows one of several paths that can be obtained by using a variable radius.

Let f^i_{arc} be the function that generates a semi-circle path from the robot's posture. This function accepts as parameters the robot's estimated pose ($\hat{\xi}_{v_t}$) and takes into account the map of the environment ($MAPv$) and r_i – it is the radius of the ith execution of the algorithm – value, where $r_i \in [R, \infty)$. The function f^i_{arc} returns the posture of collision of the semi-circle arc with a feature of the environment (virtual or real feature) and the following arc is calculated according to this posture. Thus, if a path requires n arcs to reach the desired angle (Ψ), it can be expressed as shown in Eq. (6).

$$\begin{aligned} & f^n_{arc}\left(f^{n-1}_{arc}\left(\ldots\left(f^1_{arc}(\hat{\xi}_{v_t}, MAPv, r_1)\right)\ldots\right)\right) \\ & \quad = f^n_{arc} \circ f^{n-1}_{arc} \circ \ldots \circ f^1_{arc}(\hat{\xi}_{v_t}, MAPv, r_1) \\ & \quad = [x_{final}\ y_{final}\ \Psi_{final}]; \\ & \Psi_{final} \geq \Psi. \end{aligned} \tag{6}$$

Each semi-circle path is described by r_i, $\hat{\xi}^i_{v_t}$ and $\hat{\xi}^{i-1}_{v_t}$, where $\hat{\xi}^i_{v_t} = f^i_{arc}(\hat{\xi}^{i-1}_{v_t})$ for $i < n$ and the final arc is described by r_n, $\hat{\xi}^{n-1}_{v_t}$ and the desired heading Ψ.

In this paper, we propose the generation of the possible radius by the Monte Carlo method. Thus, for each f^i_{arc} in Eq. (6), a new radius value is generated by the Monte Carlo method. The radius value remains in the following domain: $r \in [R, \infty)$, where R is the minimum radius of turning. The Monte Carlo radius generation method uniformly generates a radius value in that domain. Thus, the robot can turn from following an arc with the minimum radius to following a straight line. Equation (7) shows the Monte Carlo radius generation procedure:

$$\begin{aligned} & \mu \sim U(0, 1), \\ & r = g(R, \mu). \end{aligned} \tag{7}$$

In Eq. (7), μ is an outcome of the uniform distribution U and $g(R, \mu)$ generates a value of the radius of turning larger than, or equal to, the minimum radius (R). The function g can be of the form $r = R + \tau\mu$, where τ is a constant value sufficiently large ($\tau \geq 10R$). Thus, if μ is near zero, then $r \cong R$ and if μ is near 1, $r \cong \tau$. For example, if the robot is navigating within a 10×10 m environment, then $\tau = 100$ will be appropriate for representing a straight line within that environment.

As stated before, at each arc generation in Eq. (6), a new radius is determined by the Monte Carlo method. Executing Eq. (6) several times, we can obtain several plausible paths. Let us suppose that we have obtained P_{paths} different paths. Then, from the set of plausible P_{paths} paths, only the one that is more appropriate according to a previously established criterion will be chosen to be followed

https://doi.org/10.1017/S0263574713000131 Published online by Cambridge University Press

1: **for** $j = 1$ to P_{paths} **do**
2: $Cost_j = 0$
3: **for** $i = 1$ to A_{arcs} **do**
4: radius = MonteCarloRadiusGeneration()
5: [ARC, $ArcCost_i$, BoolAngle] = ArcGeneration(robot's position, radius, $\hat{x}_{v,\theta}$, $MAPv$, P, SIDE, DIRECTION)
6: **if** ARC intersects with $MAPv$ **then**
7: $SIDE = \overline{SIDE}$
8: $DIRECTION = \overline{DIRECTION}$
9: break
10: **end if**
11: $Cost_j = Cost_j + length(\text{ARC})$
12: $P_{paths,j} = [P_{paths,j}, \text{ARC}]$
13: **if** BoolAngle = true **then**
14: break
15: **end if**
16: **end for**
17: $Cost_j = Cost_j + (i - 1)$
18: **if** $Cost_j < Cost$ **then**
19: $Cost = Cost_j$
20: SelectedPath = j
21: **end if**
22: **end for**

Fig. 7. Path planning algorithm with variable radius.

by the mobile robot. Figure 7 shows the *for* loop of the algorithm for the maneuverability strategy with variable radius of turning presented in this work. The variables' definition and initialization are the same as those shown in the algorithm of Fig. 4 (lines of code 2–8). In Fig. 4, $Cost_j$, $j = 1 \ldots P_{paths}$ – the maximum number of feasible paths to be generated – is the cost associated with each feasible path and $ArcCost$ is the cost associated with a given arc of a path; A_{arcs} is the maximum number of arcs that each feasible path should have. *SelectedPath* is the path with minimum cost selected among all possible paths found by the algorithm; $MonteCarloRadiusGeneration$ is the Monte Carlo sampling method for radius generation shown in Eq. (7). In this work, the cost associated to the jth path ($j = 1 \ldots P_{paths}$) was defined as follows: $Cost_j = \sum_{i=1}^{A_{arcs}} length(ARC_i) + (A_{arcs} - 1)$. Therefore, the cost of a given path is its total length plus the number of times that the path changes its direction. Nevertheless, other criteria can be applied.

Thus, for example, if in Fig. 7 $P_{paths} = 100$ and $A_{arcs} = 10$, the algorithm will search for 100 feasible paths that allow the mobile robot to reach the desired orientation Ψ. Each path will have no more than 10 arcs. The number of arcs associated with each path will depend on the arc in which the path reaches the desired orientation. If a path does not reach the desired orientation within 10 arcs, then it is discarded and a new path is searched.

2.4.2. First movement decision procedure. The *first movement procedure* is associated with both path planning strategies: with fixed radius of turning and with variable radius of turning. This procedure determines which is the most convenient SIDE (left or right) and DIRECTION (backward or forward) to be taken for the first arc generation. The decision is based on the available space according to the mapped environment and the robot's pose within it. The first movement decision procedure can be summarized as follows:

(1) The $MAPv$ is divided into four quadrants from the mobile robot's reference frame (blue lines in Fig. 8). Each semi-axis starts at the robot's pose.
(2) From the mobile robot's pose, four arcs are generated with the minimum radius of turning. One arc per each quadrant. The arc generation stops when the arc reaches an element from $MAPv$ or an axis that divides quadrants.
(3) The chosen quadrant is the one that has the largest arc. If two or more arcs have the same length, then the quadrant is chosen arbitrarily.

Figure 8 shows the quadrants' division.

https://doi.org/10.1017/S0263574713000131 Published online by Cambridge University Press

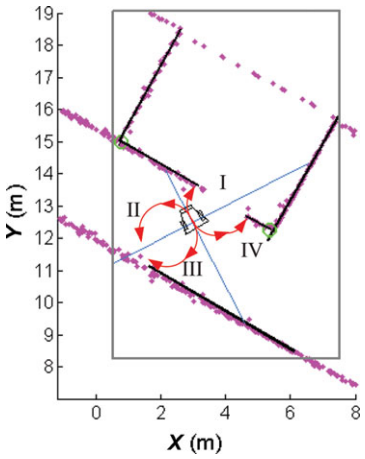
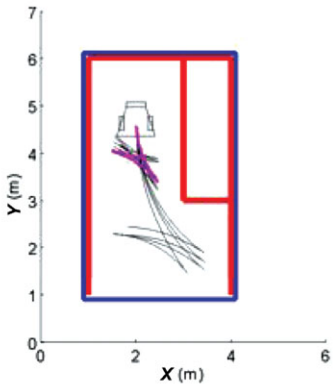
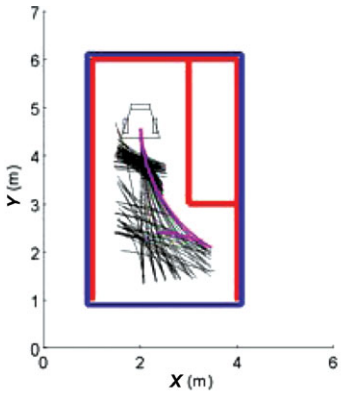


Fig. 8. (Colour online) Quadrants determination for the first movement decision.

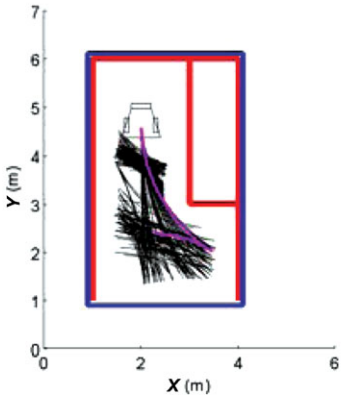
2.4.3. Statistical analysis of the Monte Carlo method for variable radius. As stated in Eqs. (6) and (7), the Monte Carlo method is used for the determination of the radius of turning of the mobile robot. Let us consider that the best path is chosen according to the cost criterion presented in Section 2.4.1. Figure 9 shows a simulated environment, where the solid red lines represent walls from the environment and solid blue lines represent the virtual lines that circumscribe the map. Figure 9(a) shows ten possible paths ($P_{paths} = 10$) – in solid black – the chosen path is in solid magenta; Fig. 9(b) shows fifty possible paths and the chosen one (also in solid magenta); Fig. 9(c) shows 100 possible paths (in solid black) and the chosen path in solid magenta. As can be seen, for the simulated environment, the path found in Fig. 9(b) is very close to the one found in Fig. 9(c). Figure 10 shows a statistical analysis of the paths found by the Monte Carlo method. Given the environment shown in Fig. 6, Fig. 10 shows that after 100 realizations, the minimum path found by the Monte Carlo method remains with an almost constant standard deviation. According to this, for the environment shown in Fig. 6, 100 realizations are sufficient to find an appropriate minimum path. Figure 10 represents the mean value and the standard deviation of the cost associated with the minimum path. The mean and standard deviation associated with each number of paths were obtained after 100 executions of the turning strategy algorithm presented in this work. Thus, e.g., for the case of 50 Monte Carlo minimum path generation shown in Fig. 10, 100 different executions of the algorithm shown in Fig. 7 were run.



(a) Generation of 10 feasible paths



(b) Generation of 50 feasible paths



(c) Generation of 100 feasible paths

Fig. 9. (Colour online) Best path chosen from different sets of realizations. Panel (a) shows the best path (solid magenta) chosen from 10 different feasible paths obtained by the algorithm shown in this paper. Panel (b) shows the best path from a set of 50 feasible paths whereas panel (c) shows the best path – also in solid magenta – encountered from a set of 100 feasible paths.

https://doi.org/10.1017/S0263574713000131 Published online by Cambridge University Press

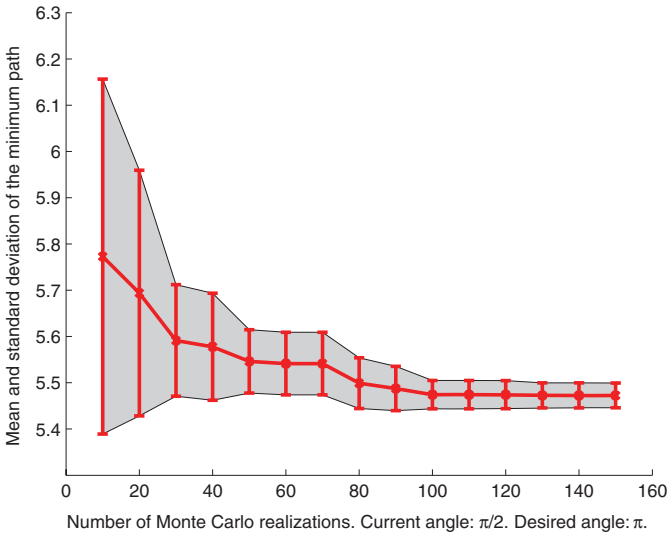


Fig. 10. (Colour online) Statistical analysis of the Monte Carlo minimum path generation. After 100 realizations, the cost associated with the minimum path shows a non-changing standard deviation.

In addition, Fig. 11 shows the computational cost associated with the statistical analysis of Fig. 10 for three different cases: $P_{paths} = 100,\ 1000$ and $10,000$. As can be seen, the computational cost increases linearly as increases the number of feasible paths found by the planning strategy. The solid red line in Fig. 11 represents the mean computational cost and its associated standard deviation for each case; the solid black lines show the upper and lower bounds of the computational cost evolution. In all cases, the maximum computational time is 0.08 s, which is smaller than the system sampling time $\Delta_t = 0.2$ s as stated in Section 2.1 – this is crucial in order to avoid delayed open loop situations.[4,19] Also, the hardware used to perform the computational cost analysis is the one mentioned in Section 2.1.

2.4.4. Worst-case path: Safe collision regions. In this work, the mobile robot is considered as a rigid body, then the path planning must consider all the points of the vehicle. Thus, we have considered the mobile robot to be circumscribed by a rectangle. If any vertex or edge of the circumscribing rectangle intersects an element of the map obtained by the SLAM algorithm, then the path changes direction as it is shown in lines of code (7)–(8) in the algorithm of Fig. 7.

Considering that in this work the SLAM algorithm is used for localizing the vehicle within an environment and no odometric information is used, the robot pose has a variance associated with it. Thus, if the SLAM algorithm is consistent (the error of the SLAM system state remains bounded; see ref. [3]) then the pose estimation of every point of the vehicle behaves consistently. Furthermore, in Eq. (4) the SLAM algorithm is estimating the pose of the point of control of the vehicle. Thus, the rest of the points of the robot have a variance value propagated from $\Sigma_{v,v}$ in Eq. (5). The propagation of the variance can be found in ref. [3]. According to this, it is possible to assign a variance to any point of the rectangle that circumscribe the mobile robot. For example, Fig. 12 shows the covariance ellipse associated with the four vertexes of the robot's circumscribing rectangle. In this work, the covariance ellipse of the uncertainty of the points belonging to the circumscribing rectangle was used as collision risk zones of the vehicle. Thus, if the ellipses associated with the covariance of the points of the mobile robot circumscribing rectangle intersect an element of the map obtained by the SLAM system state, then the generated path changes its direction (see Fig. 12).

Let us consider Eq. (8) which corresponds to the correction stage of the EKF:

$$\begin{cases} K_t = \Sigma_t^- H_t^T (H_t \Sigma_t^- H_t^T + R_t)^{-1}, \\ \hat{\xi}_t = \hat{\xi}_t^- + K_t(z_t - h(\hat{\xi}_t^-)), \\ \Sigma_t = (I - K_t H_t)\Sigma_t^-. \end{cases} \tag{8}$$

In Eq. (8), K_t is the Kalman gain at time instant k, Σ_t^- is the predicted SLAM system state covariance matrix, H_t is the Jacobian matrix of the measurement model (see Eqs. (2)–(3)), R_t is the covariance matrix associated with the uncertainty of the measurement model, $\hat{\xi}_t^-$ is the predicted SLAM system state whereas $\hat{\xi}_t$ is the corrected SLAM system state, z_t is the current observation, Σ_t is the corrected covariance matrix of the SLAM system state, and I is an identity matrix of the size of Σ_t.

According to Eq. (8), it is possible to see that

$$\Sigma_t = (I - K_t H_t)\Sigma_t^- = \Sigma_t^- - K_t H_t \Sigma_t^-. \tag{9}$$

Furthermore, in Eq. (9), Σ_t^-, Σ_t and $K_t H_t \Sigma_t^-$ are positive definite matrices (see ref. [3]). Thus,

$$|\Sigma_t| = |(I - K_t H_t)\Sigma_t^-| = |\Sigma_t^- - K_t H_t \Sigma_t^-| \leq |\Sigma_t^-|. \tag{10}$$

As can be seen in Eq. (10), if the SLAM algorithm remains consistent, then the determinant of the corrected SLAM system state covariance matrix ($|\Sigma_t|$) is smaller than the determinant of the predicted SLAM system state covariance matrix ($|\Sigma_t^-|$) for time instant t.

Let us consider $\gamma_i, i = 1 \ldots n$ the eigenvalues of the $n \times n$ matrix Σ_t and $\delta_i, i = 1 \ldots n$ the eigenvalues of $n \times n$ matrix Σ_t^-. Considering that $|\Sigma_t| \leq |\Sigma_t^-|$, then

$$\gamma_1 \times \gamma_2 \times \cdots \times \gamma_n \leq \delta_1 \times \delta_2 \times \cdots \times \delta_n, \tag{11}$$

where the "$\times$" is the product symbol.

Furthermore, under consistency and due to the positive definiteness of Σ_t and Σ_t^-, each eigenvalue of Eq. (11) can be expressed as shown in Eq. (12):.[3]

$$0 \leq \gamma_i \leq \delta_i; \quad i = 1 : n. \tag{12}$$

Then, the eigenvalues of Σ_{vv} in $\Sigma_t - \Sigma_{vv}$ is a principal sub-matrix of the corrected covariance matrix of the mobile robot – will be equal to or smaller than the eigenvalues of Σ_{vv}^- in Σ_t^- – the predicted covariance matrix of the mobile robot (see Eq. (5)). Thus,

$$|\Sigma_{vv}| \leq |\Sigma_{vv}^-|. \tag{13}$$

Equation (13) shows that, when the SLAM algorithm is corrected then the covariance ellipse associated with the pose

https://doi.org/10.1017/S0263574713000131 Published online by Cambridge University Press

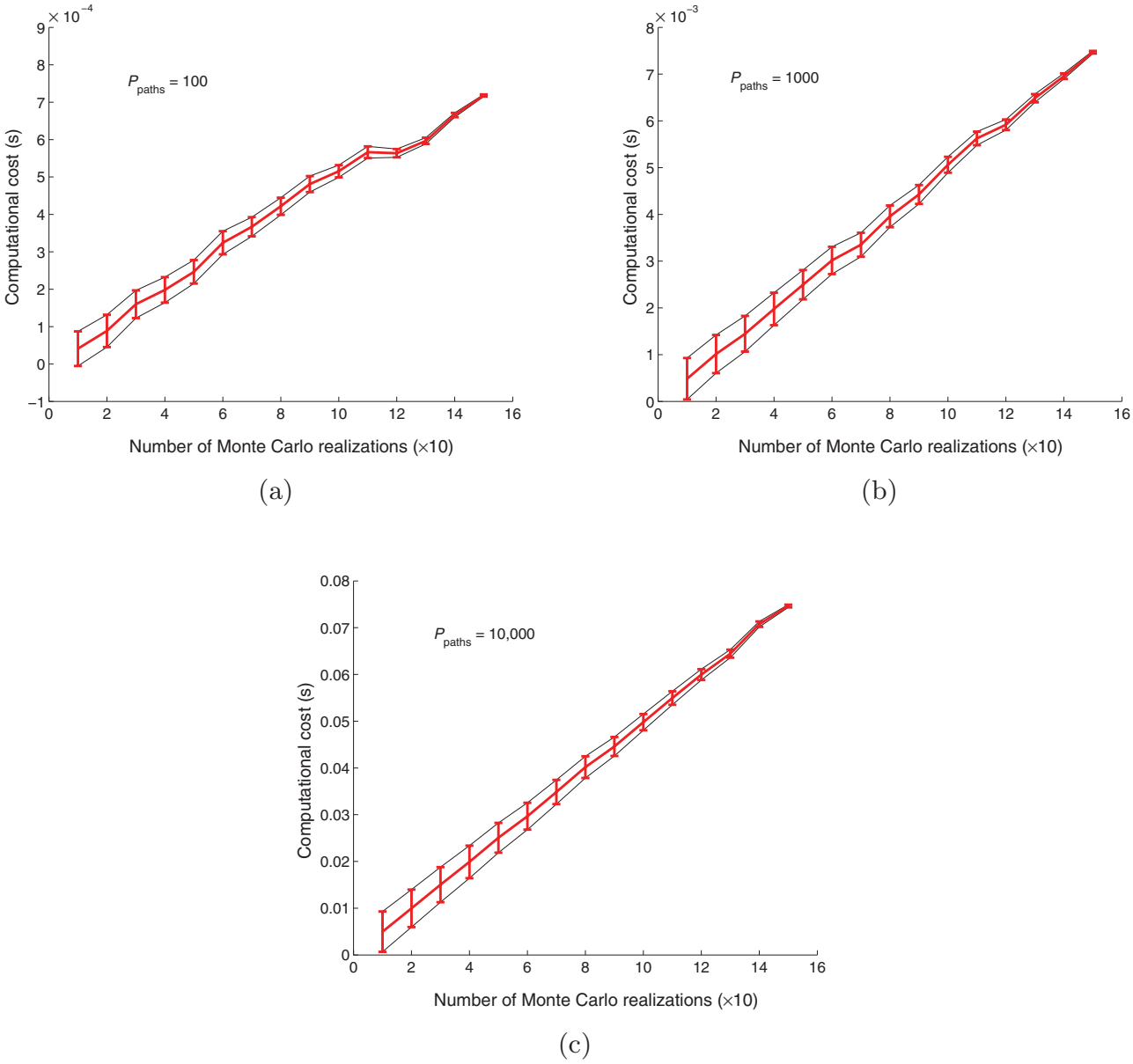


Fig. 11. (Colour online) Computational cost analysis. Panel (a) shows the case for $P_{paths} = 100$; panel (b) shows the case for $P_{paths} = 1000$ and panel (c) shows the case for $P_{paths} = 10,000$. In all cases, the maximum computational cost remains smaller than the sampling time of the robotic system.

of the vehicle has smaller area when compared with the ellipse of the predicted covariance matrix of the pose of the mobile robot.

Let us now consider the prediction stage of the EKF (Eq. (14)):

$$\begin{cases} \hat{\xi}_t^- = f(\hat{\xi}_{t-1}, u_{t-1}), \\ \Sigma_t^- = A_t \Sigma_{t-1} A_t^T + W_t Q_t W_t^T. \end{cases} \tag{14}$$

In Eq. (14), ξ_t^- is the predicted state of the system at time t; u_t is the input control commands; ξ_{t-1} is the corrected state of the system at time $t-1$ and f describes the motion model of the elements of x; A_t is the Jacobian of g in the elements of x; Q_t is the covariance matrix of the noise associated with the process and W_t its Jacobian.

As stated before, the SLAM algorithm is used to localize the robot within the environment that it is navigating during the maneuverability strategy. If no feature with correct feature association is extracted from the environment, then the correction stage of the EKF (see Eq. (8)) will not be executed; therefore, the SLAM system state is only predicted using Eq. (14). Let us suppose that the last correction of the SLAM algorithm was executed at time t; from $t+1$ to $t+k$, there was no correction of the SLAM algorithm, then the evolution of the SLAM system state and its covariance can be expressed as shown in Eq. (15).

As stated in Eq. (10), $|\Sigma_t| \leq |\Sigma_t^-|$ when the SLAM algorithm has correct features association and the correction stage can be performed. According to Eq. (15) if there is no feature association, then $|\Sigma_t| = |\Sigma_t^-|$. Furthermore, according to ref. [2], $|\Sigma_{t+1}^-| \geq |\Sigma_t|$. Thus, when there

https://doi.org/10.1017/S0263574713000131 Published online by Cambridge University Press

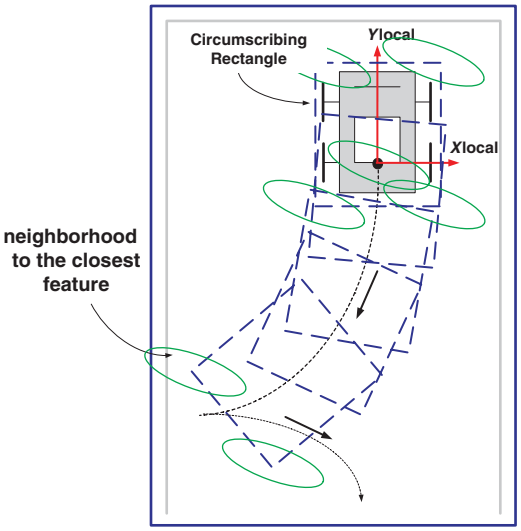


Fig. 12. (Colour online) Collision risk zones of the mobile robot circumscribing rectangle based on the ellipses associated with the SLAM system state covariance.

is no features association, the covariance matrix of the SLAM system state tends to grow, which implies that the covariance ellipse associated with the robot's pose also grows ($|\Sigma^-_{vv,t+1}| = |\Sigma_{vv,t+1}| \geq |\Sigma_{vv,t}|$).

In this work, the determination of the collision's zones is given by the area of the uncertainty ellipse associated with the robot's pose. Considering that the worst case is presented when the SLAM algorithm is not corrected (because the covariance matrix of the SLAM system state tends to grow, as previously stated), then during the planning process the free collision's zones are calculated based on the covariance matrix of the SLAM system state prediction stage (Σ^-_t) which is calculated based on the motion that the vehicle would have given the path being planned.

$$
\begin{aligned}
&\textit{for time } t+1: \\
&\quad \hat{\xi}^-_{t+1} = f(\hat{\xi}_t, u_t) \\
&\quad \Sigma^-_{t+1} = A_{t+1}\Sigma_t A^T_{t+1} + W_{t+1}Q_{t+1}W^T_{t+1} \\
&\textit{for time } t+2: \\
&\quad \hat{\xi}_{t+1} = \hat{\xi}^-_{t+1} \\
&\quad \Sigma_{t+1} = \Sigma^-_{t+1} \\
&\quad \hat{\xi}^-_{t+2} = f(\hat{\xi}_{t+1}, u_{t+1}) \\
&\quad \Sigma^-_{t+2} = A_{t+2}\Sigma_{t+1}A^T_{t+2} + W_{t+2}Q_{t+2}W^T_{t+2} \\
&\quad \vdots \\
&\textit{for time } t+k: \\
&\quad \hat{\xi}_{t+k-1} = \hat{\xi}^-_{t+k-1} \\
&\quad \Sigma_{t+k-1} = \Sigma^-_{t+k-1} \\
&\quad \hat{\xi}^-_{t+k} = f(\hat{\xi}_{t+k-1}, u_{t+k-1}) \\
&\quad \Sigma^-_{t+k} = A_{t+k}\Sigma_{t+k-1}A^T_{t+k} + W_{t+k}Q_{t+k}W^T_{t+k}
\end{aligned}
\tag{15}
$$

2.4.5. Kinematic compatibility test. The compatibility test of the algorithm proposed in this work can be found in other work of the author (see ref. [41]).

2.4.6. Existence of the solution. Considering that the planning strategy is based on following a circle trajectory until the path has reached the desired angle, it seems intuitive for one to think that the solution always exists because of the 360 degrees of the circle. In this section, we show that the maneuverability planning strategy with variable radius based on the Monte Carlo method – proposed in this work – reaches a neighborhood of the desired angle.

Formally, let us suppose that the vehicle's initial orientation is $\hat{\xi}_{\theta_t} = 0$ and the desired orientation is $\Psi \in (0, 360]$. Let us also suppose that at the end of each arc generation – as explained in the algorithms of Figs. 4 and 7 – the final orientation of the vehicle would be of β_i, where i is the ith arc of the planning strategy. Furthermore, $\beta_i \geq 0$ and $\beta_i - \beta_{i-1} = \Delta_{\beta,i} \geq 0$. Thus, the vehicle in the planning strategy is turning in a positive angle (clockwise direction).

According to Eq. (6), we have

$$
\begin{aligned}
&f^n_{arc}\left(f^{n-1}_{arc}\left(\ldots\left(f^1_{arc}(\hat{\xi}_{v_t}, MAPv, r_1)\right)\ldots\right)\right) \\
&\quad = f^n_{arc} \circ f^{n-1}_{arc} \circ \ldots \circ f^1_{arc}(\hat{\xi}_{v_t}, MAPv, r_1) \\
&\quad = [x_{final}\ y_{final}\ \Delta_{\beta,n} \\
&\qquad + \Delta_{\beta,n-1} + \Delta_{\beta,n-2} + \cdots + \Delta_{\beta,1}] \\
&\Delta_{\beta,n} + \Delta_{\beta,n-1} + \Delta_{\beta,n-2} + \cdots + \Delta_{\beta,1} \geq \Psi.
\end{aligned}
\tag{16}
$$

In Eq. (16), n is the number of arcs needed to reach an orientation bigger than Ψ. Taking into account that the Monte Carlo method for the radius determination used in this work implies that some radius might be equal to infinity, then not all the terms of the second member of Eq. (16) are zero; $\beta_i = 0$ implies radius equal to infinity.

In addition, let us suppose now that $\beta_i \leq 0$, $\beta_i - \beta_{i-1} = \Delta_{\beta,i} \leq 0$ and $\Psi \in (0, 360]$. The last means that the planning algorithm is turning in an opposite direction to that shown in Eq. (16). Thus,

$$
\begin{aligned}
&f^m_{arc}\left(f^{m-1}_{arc}\left(\ldots\left(f^1_{arc}(\hat{\xi}_{v_t}, MAPv, r_1)\right)\ldots\right)\right) \\
&\quad = f^m_{arc} \circ f^{m-1}_{arc} \circ \ldots \circ f^1_{arc}(\hat{\xi}_{v_t}, MAPv, r_1) \\
&\quad = [x_{final}\ y_{final}\ 360 + (\Delta_{\beta,m} + \Delta_{\beta,m-1} \\
&\qquad + \Delta_{\beta,m-2} + \cdots + \Delta_{\beta,1})] \\
&360 + (\Delta_{\beta,m} + \Delta_{\beta,m-1} + \Delta_{\beta,m-2} \\
&\qquad + \cdots + \Delta_{\beta,1}) \leq \Psi.
\end{aligned}
\tag{17}
$$

From Eqs. (16) and (17) it is possible to see that either from a clockwise or counterclockwise direction of turning, the robot will reach a neighborhood of the desired angle, which proves the existence of a solution of the planning strategy presented in this work.

2.5. Motion controller

The path following control law for non-holonomic vehicles proposed in ref. [38] is used in the present control system. This is a stable control law based on an improved linearization technique. The inputs to the vehicle are the reference posture $[x_r\ \ y_r\ \ \theta_r]^T$ and the reference path ($P_{paths,j}$ in Fig. 7). The controller implementation details can be found in ref. [38].

https://doi.org/10.1017/S0263574713000131 Published online by Cambridge University Press



(a) (b)

Fig. 13. (Colour online) Two views of the car-like robot used in this work.

3. Experimental Results

Two experimental cases are shown herein. The indoors experimentation was carried out at the facilities of the Universidad Tecnológica Nacional Regional Córdoba, Argentina, whereas the outdoors experimentation was carried out at an experimental olive grove. The mobile robot used in this work is the non-holonomic car-like mobile robot shown in Fig. 13. The vehicle has a range laser (built by SICK) incorporated that acquires 181 measurements from 0 to 180 degrees. The maximum range was set to 20 m. The laser is mounted at the front of the vehicle's chassis. Also, the mobile robot does not have proximity or range sensors at the back of its structure. The mobile robot processing was performed by a laptop computer mounted on it (see Fig. 13a) with an AMD Athlon X2 of 3200 MHz microprocessor. All algorithms presented herein were developed in Visual C++ using OpenCV libraries.

3.1. First case: Indoor environment

Figure 14 shows three different experiments within the same environment. As can be seen, the map is successfully built. The solid blue lines represent the lines associated with walls of the environment; the green circles are detected corners; magenta points are raw laser data and the dotted blue line is the path traveled by the vehicle.

The navigation strategy of the robot was performed by the local frontier points determination method presented in ref. [23]. The turning strategy was activated once a "dead-end" situation was detected by the robot.

Figure 15 shows the evolution of the standard deviation associated with the features of the map shown in Fig. 14(a) and with the robot's position. As can be seen, during the experimentation the estimation of the parameters of the features (see Eqs. (3) and (2)) was consistent.

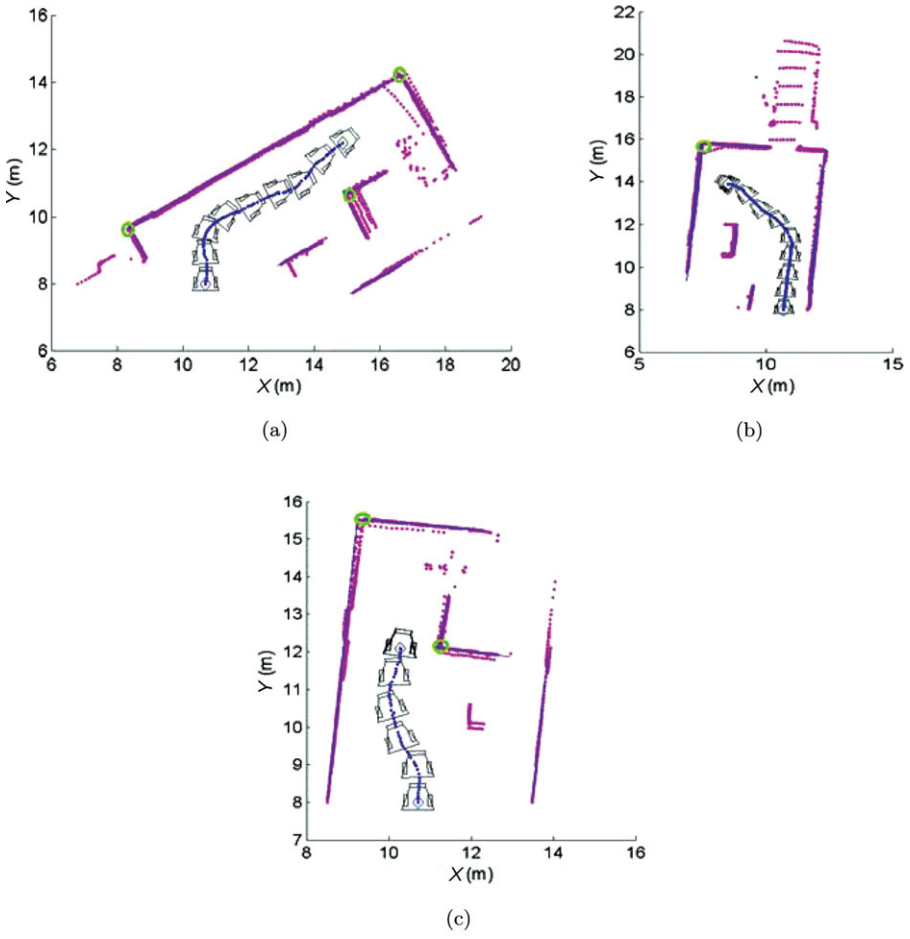


Fig. 14. (Colour online) Three different map reconstructions by the SLAM algorithm within the same environment. Solid blue lines represent walls of the environment; green circles are extracted corners; magenta points are raw laser data and the dotted blue line is the path traveled by the mobile robot.

https://doi.org/10.1017/S0263574713000131 Published online by Cambridge University Press

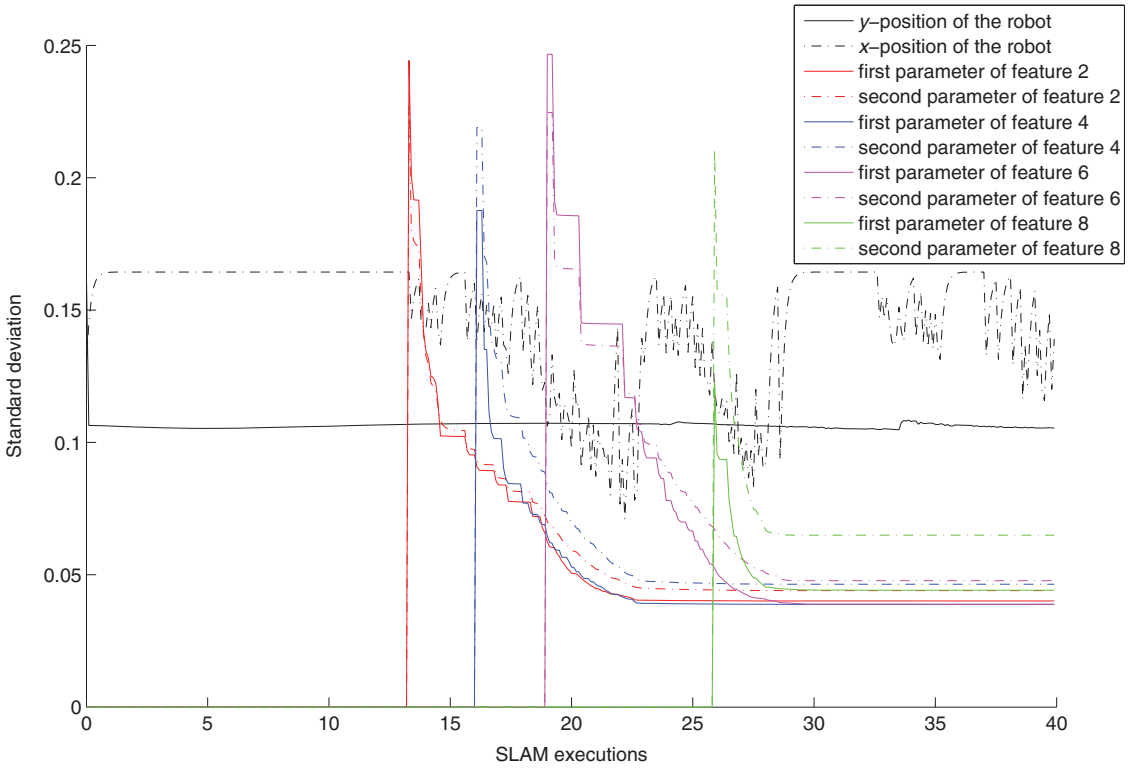


Fig. 15. (Colour online) Evolution of the standard deviation associated with the robot's position and the parameters of the features of the environment. The estimation of the features remains consistent as their standard deviations tend to decrease.

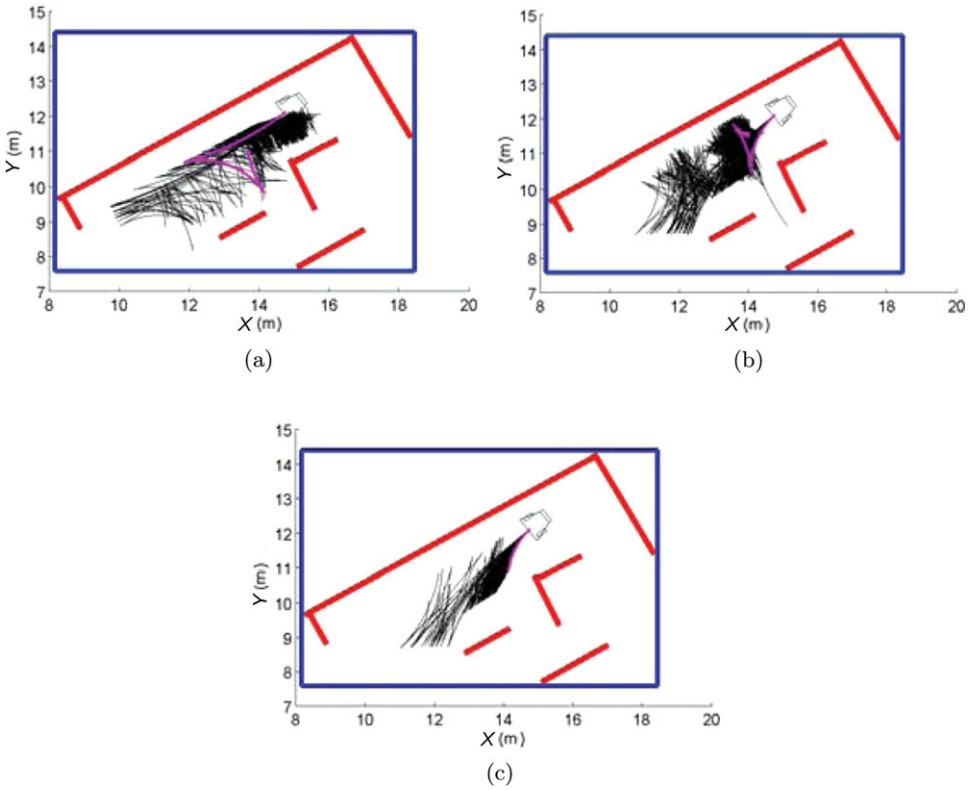


Fig. 16. (Colour online) Path planning for different desired angles over the same environment. Panel (a) shows the optimal path from 200 realizations for $\Psi = -\pi/2$; panel (b) shows the optimal path for $\Psi = -\pi$ and panel (c) for $\Psi = \pi/2$.

https://doi.org/10.1017/S0263574713000131 Published online by Cambridge University Press

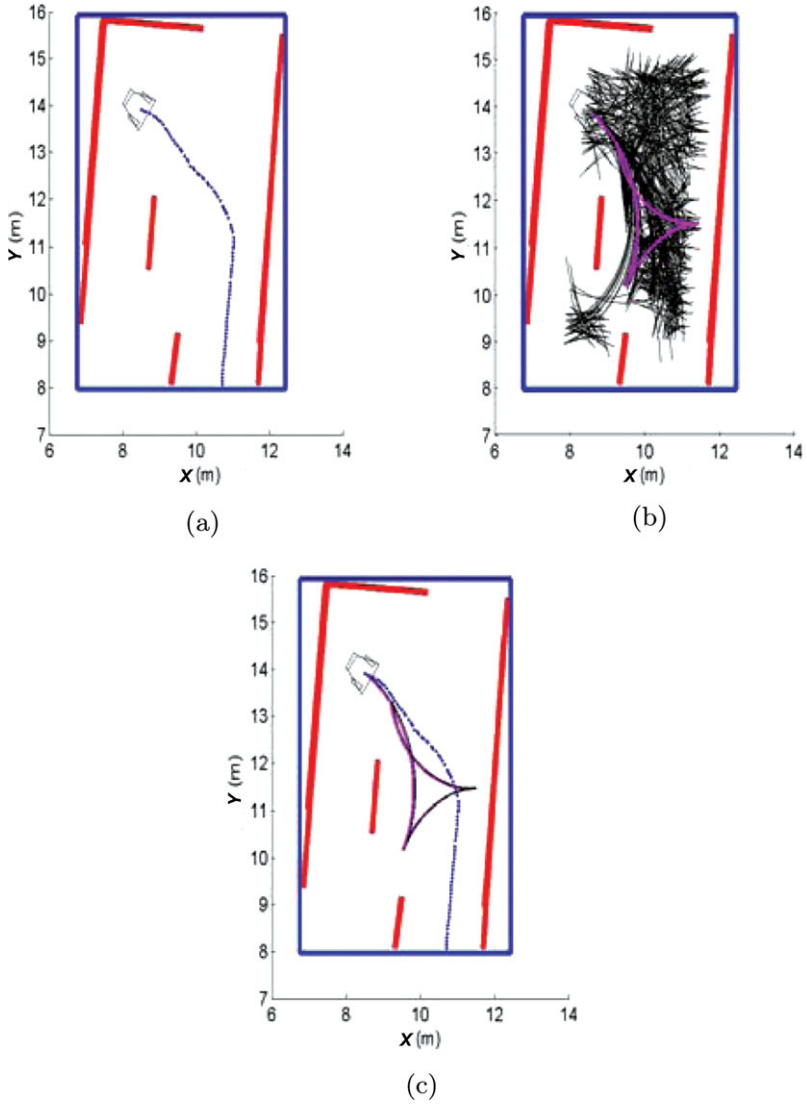


Fig. 17. (Colour online) Path planning for a specific desired angle: the vehicle must returns through the path it was driving. Panel (a) shows the map reconstruction of the environment; panel (b) shows 200 feasible paths and the optimal path (magenta); panel (c) shows the path planned (solid magenta) and the path traveled by the vehicle (in solid black).

3.1.1. Path planning results. Figure 16 shows the path planning algorithm results for the case shown in Fig. 14(a) for different desired angles. Thus, Fig. 16(a) shows the optimal path from 200 feasible paths found by the algorithm presented in Fig. 7. The number of paths (200) was determined by the statistical analysis of the Monte Carlo method given the dimensions of the environment that the robot was navigating through. The solid red lines represent walls of the environment – extracted from the SLAM system state; the solid blue lines represent the environment circumscribing rectangle; the solid black lines are the 200 realizations of the path planning algorithm and the solid magenta line is the optimal path chosen by the system. For the case shown in Fig. 16, the robot's pose is $\hat{x}_t = [14.87,\ 12.19,\ 0.68]$ within the environment. In Fig. 16(a), the desired angle is $\Psi = -\pi/2$; in Fig. 16(b), $\Psi = -\pi$, whereas in Fig. 16(c), $\Psi = \pi/2$. The minimum radius of turning of the vehicle during the experiments was of 2 m. As can be seen in Figs. 16(a–c), the planning algorithm takes into account the safe collision regions in order to generate a feasible path.

Figure 17 shows another case of the path planning algorithm. For this situation, the vehicle must return through the passageway it was driving. Figure 17(a) shows the path traveled by the vehicle and the map reconstructed by the SLAM algorithm. The solid red segments correspond to lines of the SLAM system state associated with walls of the environment; the solid blue lines correspond to the rectangle that circumscribes the environment; the dotted blue line is the path traveled by the vehicle. Figure 17(b) shows 200 feasible paths (solid black lines) and the optimal path (solid magenta); Fig. 17(c) shows the path planned and the path traveled by the mobile robot driven by the trajectory controller. For the case shown in Fig. 17, the pose of the robot at the moment of the planning process is $\hat{x}_t = [8.50,\ 13.90,\ 2.55]$. The desired angle is $\Psi = -\pi/2$.

3.2. Second case: Outdoor environment

An implementation of the maneuverability algorithm presented in this work within an olive grove environment (shown in Fig. 18) using the vehicle shown in Fig. 13 is presented herein. The SLAM algorithm acquired only point-

https://doi.org/10.1017/S0263574713000131 Published online by Cambridge University Press



Fig. 18. (Colour online) Picture of the olive grove environment used in the experimentation.

based features – associated with the stems of the grove – modeled as shown in Eq. (2). The vehicle navigated through the furrows until the turning strategy was manually activated. The desired orientation was previously established. The olive grove consisted of 36 furrows of 80 m long and approximately 10 m wide. The separation between stems is approximately 2.5 m. The environment is an experimental grove from the National Institute of Agriculture Technology (INTA, Instituto Nacional de Tecnologia Agropecuaria), Argentina. Details of the environment, the navigation strategy and the mapping process can be found in a previous work of the author (see ref. [43]).

Figure 19 shows the map built by the SLAM algorithm. The mapped area is a portion of the experimental grove. The solid dark line is the path followed by the vehicle (estimated by the SLAM algorithm); the blue triangles represent the GPS locations of the stems in the environment, whereas the detected ones are represented by red crosses. Each detected stem has a covariance matrix associated with it (red ellipses). Figure 19(a) shows the map built by the SLAM algorithm; Fig. 19(b) shows a close-up of Fig. 19(a). The robot's position within the environment is represented by the solid red triangle.

Figures 20 shows four realizations of the proposed maneuverability strategy within the olive grove environment. In Fig. 20(a), the starting pose of the robot is the one as shown in Fig. 19(a) (solid red triangle). The minimum radius of turning was set to $R = 5$ m in order to show that the proposed maneuverability strategy can be applied to other car-like vehicles, as shown in ref. [43]. Figure 20(b) shows a close-up of Fig. 20(a). The path planned by the system is shown in solid magenta, whereas the path followed by the vehicle is drawn over the planned path. As can be seen, the vehicle does not collide with the detected stems nor the virtual rectangle that circumscribes the environment (solid blue segments). Figures 20(c) and (d) show two cases at different vehicle's positions where the desired angle was set to $\Psi = \pi/2$; in Fig. 20(e), $\Psi = -\pi/2$. As can be seen, in all cases the turning strategy finds a feasible and collision-free path for the vehicle's maneuvers.

4. Conclusion

This paper has presented the implementation of a SLAM algorithm for a mobile robot path planning algorithm. The SLAM algorithm was based on an EKF implementation that uses geometric features extracted from the environment. In particular, this work has shown two cases: an indoor application, where the map of the environment was composed by lines (associated with walls) and corners (convex and concave) extracted by a range laser sensor mounted on the mobile robot; and an outdoor application in an olive grove. The feature extraction procedures were based on the cluster processing of raw laser data.

The map built by the SLAM algorithm was used to generate an optimal and kinematically compatible path for the mobile

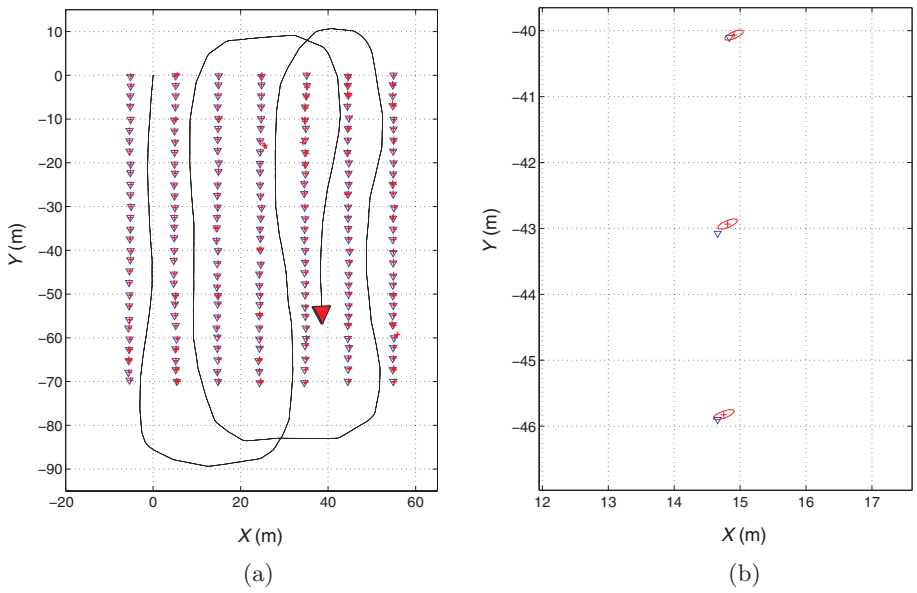


Fig. 19. (Colour online) Map obtained by the SLAM algorithm. Panel (a) shows the map built by the SLAM algorithm of the olive grove environment; panel (b) shows a close-up of (a).

https://doi.org/10.1017/S0263574713000131 Published online by Cambridge University Press

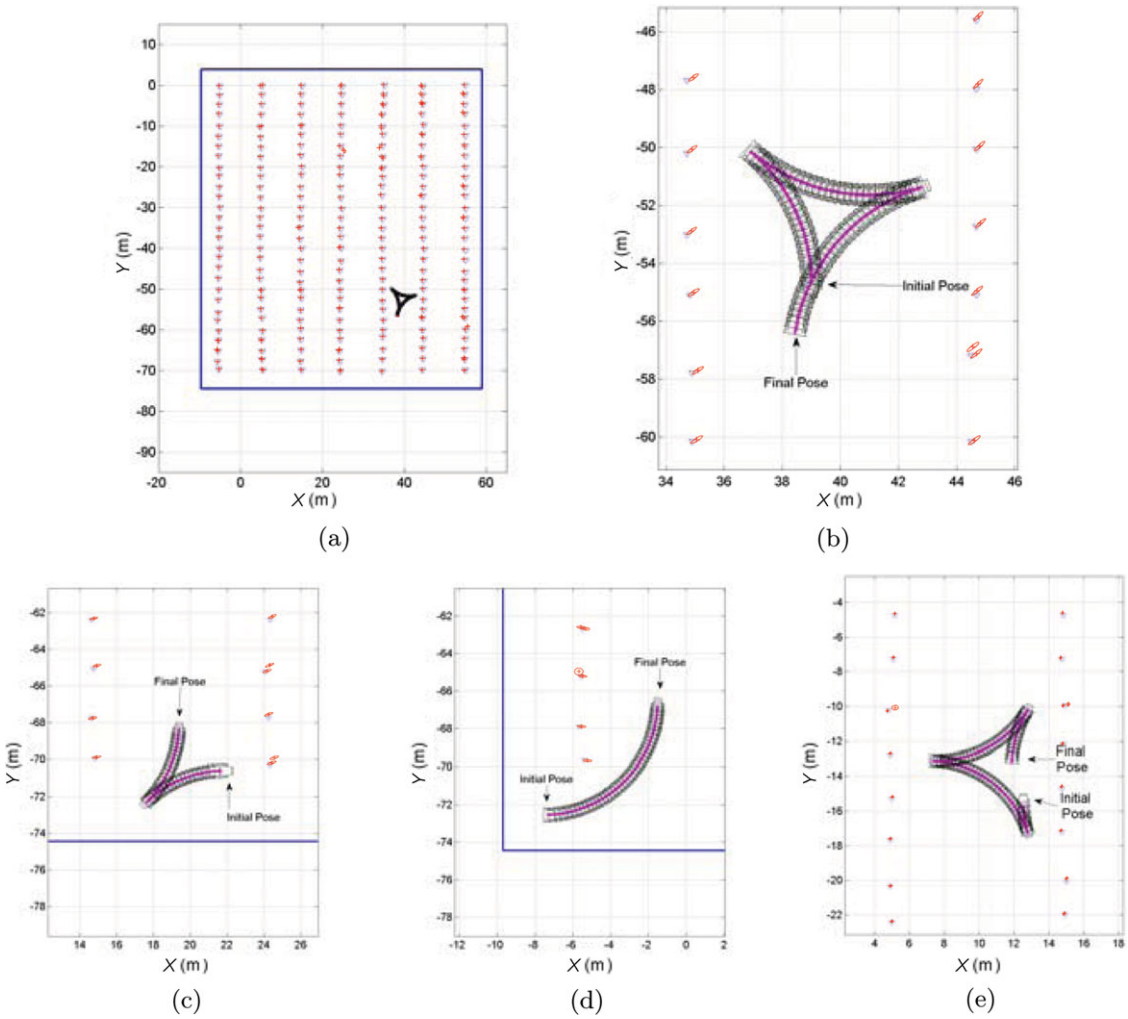


Fig. 20. (Colour online) Path planning results. Panel (a) shows the case when the vehicle has to turn back within the furrow shown in Fig. 19; whereas panel (b) shows a close-up of panel (a). Panels (c–e) show three additional cases of the turning strategy.

robot. The vehicle used was a car-like type mobile robot with a minimum radius of turning of 1 m. The path planning algorithm consisted of generating multiple arcs of different radius, from the minimum radius of turning to a straight line – which represents an infinite radius of turning. When a generated arc reaches a neighborhood of a mapped element from the environment – *posture of collision* – another arc is generated from that point but with a different direction and side of turning. The arc generation continues until the path reaches a previously given desired angle.

A Monte Carlo sampling algorithm – and its corresponding statistic analysis – has allowed the determination of the optimal path from a set of several feasible paths according to a cost function. Nevertheless, this cost function can be redesigned in order to fulfill the system requirements. In addition, a mathematical formulation was included in this work to show that the proposed path planning algorithm guarantees to always reach a solution.

Acknowledgment

The author would like to thank the Instituto de Automatica, National University of San Juan, Argentina, the Engineering Department of the Universidade Federal do Espírito Santo, ES, Brazil, the Universidad Tecnológica Nacional Regional Córdoba, Argentina, and the Department of Electronics Engineering, at the Universidad Técnica Federico Santa María, Valparaíso, Chile (DGIP grant 23.12.09, DGD and DGF), for their support.

References

1. H. D. Whyte and T. Bailey, "Simultaneous localization and mapping (SLAM): Part I. Essential algorithms," *IEEE Robot. Autom. Mag.* **13**, 99–108 (2006).
2. H. Whyte and T. Bailey, "Simultaneous localization and mapping (SLAM): Part II. State of the art," *IEEE Robot. Autom. Mag.* **13**, 108–117 (2006).
3. S. Thrun, W. Burgard and D. Fox, *Probabilistic Robotics* (MIT Press, Cambridge, 2005).
4. R. Siegwart and I. Nourbahsh, *Introduction to Autonomous Mobile Robots* (MIT Press, Cambridge, 2004).
5. S. Theodoridis and K. Koutroumbas, *Pattern Recognition* (Elsevier Academic Press, USA, 2003).
6. B. Howarth, J. Katupitiya, J. Guivant and A. Szwec, "Novel Robotic 3D Surface Mapping Using Range and Vision Fusion," **In**: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* Taipei, Taiwan (Oct. 2010) pp. 1539–1544.
7. J. E. Guivant and E. M. Nebot, "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," *IEEE Trans. Robot. Autom.* **17**, 242–257 (2001).

https://doi.org/10.1017/S0263574713000131 Published online by Cambridge University Press

8. A. Garulli, A. Giannitrapani, A. Rossi and A. Vicino, "Mobile Robot SLAM for Line-Based Environment Representation," **In**: *44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'05)* (Dec. 2005) pp. 2041–2046.
9. M. A. K. Jaradat and R. Langari, "Line Map Construction Using a Mobile Robot with a Sonar Sensor," **In**: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics* (July 2005) pp. 1251–1256.
10. D. Navarro, G. Benet and M. Martinez, "Line Based Robot Localization Using a Rotary Sonar," **In**: *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)* (Sep. 2007), pp. 896–899.
11. W. Zhow, J. Valls Miro and G. Dissanayake, "Information-efficient 3D visual SLAM for unstructured domains," *IEEE Trans. Robot.* **24**(5), 1078–1087 (2008).
12. R. Munguia and A. Grau, "Closing loops with a virtual sensor based on monocual slam," *IEEE Trans. Instrum. Meas.* **58**(8), 2377–2384 (2009).
13. F. Masson, J. Guivant and E. Nebot, "Hybrid Architecture for Simultaneous Localization and Map Building in Large Outdoor Areas," **In**: *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2002), vol. 1, pp. 570–575.
14. S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, H. Choset, K. M. Lynch and S. Thrun, *Principles of Robot Motion: Theory, Algorithms and Implementations* (MIT Press, Cambridge, 2005).
15. S. An, J. Kang, W. Choi and S. Oh, "Results for Outdoor-SLAM Using Sparse Extended Information Filters," **In**: *IEEE/RSJ International Conference on Robotics and Automation (ICRA)* (2003) pp. 1227–1233.
16. N. Sunderhauf, S. Lange and P. Protzel, "Using the Unscented Kalman Filter in Mono-SLAM with Inverse Depth Parametrization for Autonomous Airship Control," **In**: *Proceedings of the IEEE International Workshop on Safety Security and Rescue Robotics* (2007) pp. 1–6.
17. J. Welle, D. Schulz, T. Bachram and A. Cremers, "Optimization Techniques for Laser-Based 3D Particle Filter SLAM," **In**: *IEEE/RSJ International Conference on Robotics and Automation (ICRA)* (2010) pp. 3525–3530.
18. E. Eade, P. Fong and M. Munich, "Monocular Graph Slam with Complexity Reduction," **In**: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2010) pp. 3017–3024.
19. R. C. Arkin, *Behavior-based Robotics* (MIT Press, Cambridge, 1998).
20. M. Gonzalez and T. Recio, "Path tracking in motion planning," *Comput. J.* **3**, 515–524 (1993).
21. A. Ferreira, W. Celeste, F. Auat Cheein, T. Bastos Filho, M. Sarcinelli Filho and R. Carelli, "Human–machine interfaces based on EMG and EEG applied to robotic systems," *J. NeuroEng. Rehabil.* **5**, 5–10 (2008).
22. F. Sun, T. Tao, Y. Huang and T. Wang, "Motion Planning for SLAM Based on Frontier Exploration," **In**: *International Conference on Mechatronics and Automation* (2007) vol. 3, pp. 2907–2912.
23. F. A. Auat Cheein, C. De la Cruz, T. Freira-Bastos and R. Carelli, "SLAM-based cross-a-door solution approach for a robotic wheelchair," *Int. J. Adv. Robot. Syst.* **6**, 239–248 (2010).
24. S. Lavalle, *Planning Algorithms* (Cambridge University Press, Cambridge, 2006).
25. J. K. Pothal, D. R. Parhi and M. K. Singh, "Navigation of Multiple Mobile Robots Using Swarm Intelligence," *World Congress on Nature Biologically Inspired Computing* (2009).
26. Y. Tian, M. Yang and X. Yin, "Distributed Optimal Control Based on Internal Average Kinetic Energy for Multi-robot System," *4th International Conference on Autonomous Robots and Agents* (2009).
27. M. Becked, K. Macek and R. Siegwart, "Motion Planning for Car-like Vehicles in Dynamic Urban Scenarios," **In**: *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2006).
28. D. Fan and P. Shi, "Improvement of Dijkstra's Algorithm and Its Application in Route Planning," **In**: *Seventh International Conference on Fuzzy Systems and Knowledge Discovery* (2010).
29. Y.-H. Qu, Q. Pan and J.-G. Yan, "Flight Path Planning of UAV Based on Heuristically Search and Genetic Algorithms," **In**: *31st Annual Conference of IEEE on Industrial Electronics Society (IECON 2005)* (Nov. 2005) pp. 5.
30. J. Y. Hwang, J. S. Kim, S. S. Lim and K. H. Park, "A fast path planning by path graph optimization," *IEEE Trans. Syst. Man Cybern.* **33**(1), 121–129 (2003).
31. L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *Am. J. Math.* **79**, 497–516 (1957).
32. C.-W. Lim, S. Park, C.-K. Ryoo, K. Choi and J.-H. Cho, "A Path Planning Algorithm for Surveillance UAVS with Timing Mission Constraints," **In**: *International Conference on Control Automation and Systems (ICCAS),* (Oct. 2010) pp. 2371–2375.
33. E. Ferranti and N. Trioni, "Practical issues in deploying mobile agents to explore a sensor-instrumented environment," *Comput. J.* **54**, 309–320 (2011).
34. S. K. Lee, S. Lee, C. Nam and N. L. Doh, "Local Path Planning Scheme for Car-like Vehicle's Shortest Turning Motion Using Geometric Analysis," **In**: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* (Oct. 2010) pp. 4761–4768.
35. Y. Suzuki, S. Kagami and J. J. Kuffner, "Path Planning with Steering Sets for Car-like Robots and Finding an Effective Set," **In**: *IEEE International Conference on Robotics and Biomimetics (ROBIO'06),* (Dec. 2006) pp. 1221–1226.
36. Z. Lee and X. Chen, "Path Planning Approach Based on Probabilistic Roadmap for Sensor Based Car-like Robot in Unknown Environments," **In**: *IEEE International Conference on Systems, Man and Cybernetics* (Oct. 2004) vol. 3, pp. 2907–2912.
37. R. Linker and T. Blass, "Optimal Path Planning for Car-like Off-road Vehicles," **In**: *IEEE Conference on Robotics, Automation and Mechatronics* (Sep. 2008) pp. 150–154.
38. S. Rezaei, J. Guivant and E. Nebot, "Car-like Robot Path Following in Large Unstructured Environments," **In**: *IEEE Proceedings of the International Conference on Intelligent Robots and Systems* (Oct. 2003), pp. 2468–2473.
39. F. Auat Cheein, G. Scaglia, F. di Sciascio and R. Carelli, "Towards features updating selection based on the covariance matrix of the SLAM system state," *Robotica* **29**, 271–282 (2011).
40. F. Auat Cheein and R. Carelli, "Analysis of different feature selection criteria based on a covariance convergence perspective for a slam algorithm," *Sensors (Basel)* **11**, 62–89 (2011).
41. F. Auat Cheein, R. Carelli, C. De la Cruz and T. F. Bastos-Filho, "SLAM-based Turning Strategy in Restricted Environments for Car-like Mobile Robots," **In**: *IEEE International Conference on Industrial Technology (ICIT)* (Mar. 2010) pp. 602–607.
42. C. De la Cruz, T. F. Bastos, F. A. A. Cheein and R. Carelli, "SLAM-based Robotic Wheelchair Navigation System Designed for Confined Spaces," **In**: *IEEE International Symposium on Industrial Electronics (ISIE)* (July 2010) pp. 2331–2336.
43. F. Auat Cheein, G. Steiner, G. Perez and R. Carelli, "Optimized EIF-SLAM algorithm for precision agriculture mapping based on visual stems detection," *Comput. Electron. Agric.* **78**, 195–207 (2011).

https://doi.org/10.1017/S0263574713000131 Published online by Cambridge University Press