Probabilistic motion planning among moving obstacles following typical motion patterns.

Chiara Fulgenzi, Anne Spalanzani, and Christian Laugier
LIG, INRIA Rhône-Alpes, France

***Abstract*— The paper presents a navigation algorithm for dynamic, uncertain environment. The static environment is unknown, while moving pedestrians are detected and tracked on-line. Pedestrians are supposed to move along typical motion patterns represented by HMMs. The planning algorithm is based on an extension of the Rapidly-exploring Random Tree algorithm, where the likelihood of the obstacles future trajectory and the probability of collision is explicitly taken into account. The algorithm is used in a partial motion planner, and the probability of collision is updated in real-time according to the most recent estimation. Results show the performance for a car-like robot in a simulated environment among multiple dynamic obstacles.**

I. INTRODUCTION

Autonomous navigation in populated environments represents still an important challenge for robotics research. The key of the problem is to guarantee safety for all the agents moving in the space (people, vehicles and the robot itself). In contrast with static or controlled environments, where path planning techniques are suitable [1] [2], high dynamic environments present many difficult issues: the detection and tracking of the moving obstacles, the prediction of the future state of the world and the on-line motion planning and navigation. The decision about motion must be related with the on-line perception of the world and take into account the sources of uncertainty involved:

1) The limits of the perception system: occluded zones, limited range, accuracy and sensibility, sensor faults;
2) The future behaviour of the moving agents: model error, unexpected changes of motion direction and velocity;
3) New agents entering the workspace;
4) Errors of the execution system.

Many real world applications rely on reactive strategies: the robot decides only about its immediate action with respect to the updated local estimation of the environment [3]–[5]. These strategies present however some major drawback: first of all the robot can be stuck in local minima; secondly, most of the developed approaches do not take into account the dynamic nature of the environment and the uncertainty of perception, so that the robot can be driven in dangerous or blocking situations.

To face these problems, reactive techniques are combined with global planning methods: a complete plan from present state to goal state is computed on the basis of the a priori information; during execution, the reactive algorithm adapts the trajectory in order to avoid moving and unexpected obstacles [6]–[8]. If the perception invalidates the planned path replanning is performed. In all the cited methods however, uncertainty is usually not taken explicitly into account.

From the more theoretical point of view instead, many works handle a non-deterministic or probabilistic representation of the information and the planning under uncertainty problem is solved using Markov Decision Processes (MDP), Partially Observable MDPs or game theory [9]–[11]. For an overview see [2]. These approaches are however very expensive from the computational perspective, and are limited to low dimensional problems or to off-line planning. In [12] and [13] a navigation strategy based on typical pattern based and probabilistic prediction is used in a planning algorithm based on a complete optimization method, A^* . However, the problem of A^* and of all complete methods is that the computational time depends on the environment structure and obstacles: these methods are more adapted to a low dynamic environments, where the information does not change frequently, the obstacles velocity is limited and the robot can stop often and plan its future movements. Also, they require a discretization of both the state and the control space, which reduces drastically the space for finding a feasible solution, expecially for robots with non-holonome or car-like constraints. Some recent work proposes to integrate uncertainty in randomised techniques, such as Probabilistic Road Maps [14] and Rapidly-exploring Random Trees (RRT) [15] [16].

In a highly dynamic environment an *anytime* algorithm is needed, which is able to give a feasible solution at ”anytime” it is asked to. We address the problem of taking explicitly into account the uncertainty in sensing and in prediction. We want our navigation algorithm to integrate new information coming from the perception system and to be able to react to the changes of the environment. In previous work [17] we developed a probabilistic extension of the RRT algorithm to handle a probabilistic representation of the static environment and of the moving obstacles prediction. The search algorithm has been integrated in a navigation algorithm which updates the probabilistic information and chooses the best partial path on the searched tree. The navigation algorithm is based on the architecture of Partial Motion Planning (PMP, [18]), where execution and local planning work in parallel to assure safe behaviour. The static environment is initially unknown and the robot explores it and builds an occupancy grid. While in [17] motion patterns were represented by Gaussian Processes, in this paper we consider the case where prelearned motion patterns are represented by Markov chains and prediction is based on Hidden Markov Models. We

present how we adapted the algorithm to the new kind of prediction, new ideas to take into account new entering obstacles and simulation results.
The reminder of this paper is structured as follows: section III describes the representation of the static and dynamic world and how the probability of collision of a configuration is computed. Section IV recalls the RRT basic algorithm and the proposed approach. Section V recalls the PMP method and describes the planning and navigation algorithm developed. Results are presented in Section VI: an experiment with a laser scan dataset with moving pedestrians is presented and results in a simulated environment are shown. Section VII ends the paper with remarks and ideas for future work.

II. The Robot and the State Space

We consider a car-like robot moving in $\mathbb{R}^2$. The configuration space $\mathcal{C} = \{x, y, \theta, v, \omega\}$ described respectively by the position, orientation, linear and angular velocities of the robot in the workspace. The robot moves according to its motion model $q(t+1) = F(q(t), u(t))$ where input u is given by pairs (a, α) with a the linear and α the angular acceleration. The robot is subjected to kynematic and dynamic constraints: the linear velocity v is limited in the interval $[0, v_{max}]$ and the angular velocity ω is limited in $[-\omega_{max}, \omega_{max}]$. a and $alpha$ are also bounded: $a \in [a_{min}, a_{max}]$ and $\alpha \in [\alpha_{min}, \alpha_{max}]$.
Time is represented by the set $\mathcal{T} = (0, +\infty)$, which is the infinite set of discrete instants with measure unit the timestep τ. We define state space $\mathcal{X}$ of the robot, the space that represents the configuration of the robot at a certain instant in time $\mathcal{X} = \mathcal{C} \times T$. In the workspace there are static and moving obstacles. The task of the robot is to move from the initial configuration q_0 to a goal configuration q_{goal} in finite time without entering in collision with any obstacle. A solution trajectory is a sequence of states from q_0 to q_{goal} that is feasible according to the motion model of the robot and that is collision free: ie each configuration and each transition of the sequence are collision free. We assume that the position of the robot is known at each instant and that the robot moves following according to its motion model without error. In the deterministic case, the configuration space $\mathcal{C}$ can be divided in $\mathcal{C}_{free}$, the set of free configurations of the robot and $\mathcal{C}_{obs}$, the set of configurations where the robot is in collision with an obstacle. In our case instead we want to give a probabilistic representation of environment perception and prediction uncertainty and we need to define a probability of collision for each robot configuration. In the following paragraphs we explain how this probability is computed for a considered state of the robot X_r.

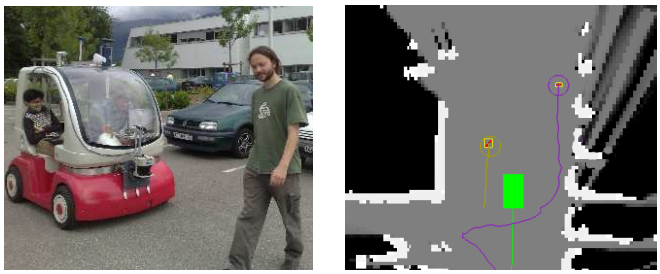
III. Probability of Collision

A. The Static environment

The 2D static environment is represented by an occupancy grid [19]: the space is divided in square cells. The environment is initially unknown, and the probability of occupation P_{occ} of each cell is fixed at 0.5. During navigation the space is observed by mean of a distance sensor (laser range finder). Assuming static environment, the probability of occupation of each cell is recursively updated according to the observations and estimated using a Bayesian filter. The probability of occupation of a point in the space is retrieved by the probability of the correspondent cell. For the set of N cells $S = (i, j)_N$ covered by the robot in state X_r the probability of collision P_c with static obstacles is given by the max probability over the set.

$$P_c(X_r, \mathcal{G}) = \max_S(P_{occ}(i, j)) \tag{1}$$

Since the grid represent the static world, there is not need of prediction and the probability of collision does not depend on the time at which the robot is in a certain configuration.



(a) (b)

Fig. 1. (a) The cycab in the parking at INRIA Rhône-Alpes and (b) an occupancy grid with the robot (green rectangle) and 2 moving obstacles (coloured circles) along with their estimated trajectories.

B. Moving obstacles

Lets assume that the moving obstacles O_i can be approximated by circles of fixed radius. The state of an obstacle is $X = (x, y, \theta, v)$, its position in the 2D space, orientation and linear velocity. Given an object observation Z, the belief state X and the prediction are estimated using Bayesian inference. The moving objects must be detected by the robot and are tracked using a Multi Hypothesis target Tracking (MHT) algorithm based on a set of Kalman Filters as in [20]: the obstacles motion can be represented with M linear motion models hypotheses A_m, each affected by zero-mean white Gaussian noise $\mathcal{N}(0, Q_m)$. At a considered instant t, the estimation of the state of an object is represented by a weighted sum of Gaussian (Gaussian mixture):

$$P(O_i^t) \leftarrow \sum_{m=1}^{M} \alpha_m \cdot \mathcal{N}(\hat{X}_m^t, \Sigma_m^t) \tag{2}$$

The state of an obstacle at future time can be predicted applying recursively the motion models, and the prediction $\hat{X}$ is always a mixture of Gaussian. We consider obstacles moving according to Hidden Markov Models as in [21]. The 2D plane is non uniformly partitioned in Voronoi regions. The probability that an obstacle O_i is in a specific region s_k is given by the integral of the probability distribution over the area of the region enlarged by the radius of the obstacles:

$$P(O_i \in s_k) = \iint_S P(O_i^t) = \sum_{m=1}^{M} \iint_{s_k} \mathcal{N}(\hat{X}_m^t, \Sigma_m^t)$$

The integral is approximated sampling the distribution uniformly with the probability and considering the ratio between the number of samples inside and outside area S.
Now, the belief of the state at time t is given by a discretized distribution over the states of the Markov model. The prediction at time horizon $t+k$ is recursively estimated propagating the estimated state:

$$P(X^{t+k}|Z^t) = \sum_{X^{t+k-1}} P(X^{t+k}|X^{t+k-1})P(X^{t+k-1}|Z^t)$$

where the first term in the sum is the probability to pass from state X^{t+k-1} to state X^{t+k} specified by the edges in the Markov model and the second is given by the observation model. Considering a state of the robot X_r^t and the moving obstacle O_i, the probability of collision is given by the integral of the probability distribution over the area $S(X_r^t)$ covered by the robot and enlarged by the radius of the obstacles:

$$P(O_i \in S(X_r^t)) = \iint_{S(X_r^t)} P(O_i^t) = \sum_{k=1}^{K} P(O_i \in s_k) \quad (3)$$

The integral is obtained summing the probability that the obstacle is in one of the regions s_k for which $s_k \cap S(X_r^t) \neq \emptyset$. Considering multiple moving obstacles, the total probability of collision is given by the probability of colliding with one or another obstacle. Under the assumption that the collision with each obstacle is conditionally independent of all others, the following equation is obtained:

$$P_c(X_r, O) = 1 - \prod_i (1 - P(O_i \in S(X_r^t))) \quad (4)$$

The probability of collision considering both the static environment and the moving obstacles is obtained in the same way:

$$P_c(X_r, O, \mathcal{G}) = 1 - (1 - P_c(X_r, \mathcal{G})) \cdot (1 - P_c(X_r, O)) \quad (5)$$

C. New obstacles entering the scene

In dynamic environments, obstacles can enter or exit the workspace during the navigation task. Also if partial planning is used, it should be taken into account that new obstacles can enter the the workspace and interfere with the next motions of the robot. If it is possible to predict from where and when some obstacle may enter the scene, a more robust planning can be performed. The robot must:

- Distinguish from where a new obstacle may come.
- Apply a probability to the fact that an obstacle may enter and a motion model.

For the first problem the robot searches for specific areas from where an obstacle may enter (*doors*). This technique is based on some assumptions about the observed space and the size, shape and behaviour of the obstacles. In the general case, the robot must be able to recognize on-line the *doors* with its perception only. In Fig. 2, a local occupancy grid obtained with a laser range finder in a car park is shown. We assumed that obstacles may enter only traversing hidden areas: i.e. they cannot pass through static obstacles. The distance between the points on the scan is studied and intervals bigger than the minimal size of an obstacles are kept as possible doors (green lines). The red circles are hypotheses of new entering pedestrians. In the case where the obstacles follow typical patterns, they are supposed to enter from points along or around the pattern prototypes. In Fig. 3, given the point of view of the robot and the pre-learned patterns, new entering obstacles hypotheses are initialized on th nearest hidden points of the patterns.
The probability of a new obstacle entering in the workspace during a certain time interval can be modeled as an homogeneous Poisson process. The probability that at least one obstacle enters the scene, is given by the following equation:

$$P[N(t+\tau) - N(t) \geq 1] = 1 - e^{-\lambda\tau} \quad (6)$$

The rate parameter λ, is the expected number of arrivals per unit time. This parameter is learned from the observation dataset, at the same time than the typical patterns. The probability of occupation correspondent to the obstacle grows with the length of the time period of prediction according to equation 6.

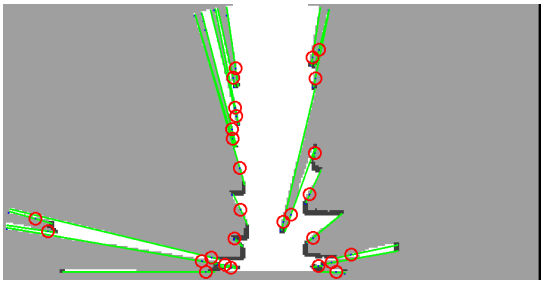


Fig. 2. A partial grid map, the extracted *doors*(green) and the supposed new entering obstacles (red) .

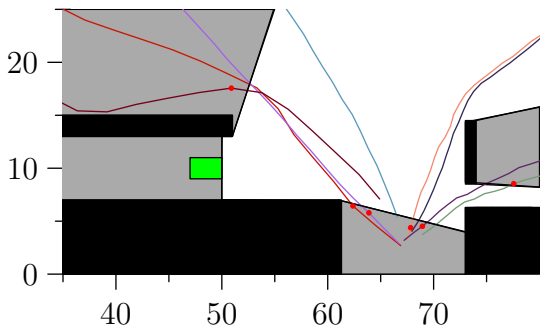
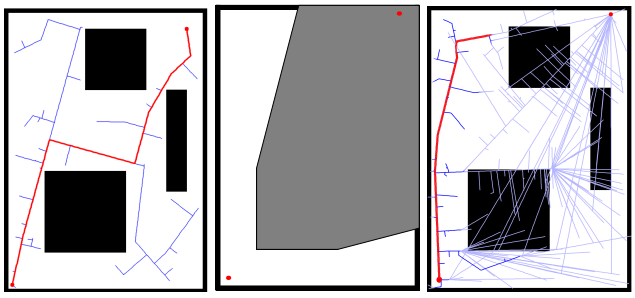


Fig. 3. A partial grid map, the extracted *doors* and the supposed new entering obstacles.

IV. Probabilistic RRTs

A. Basic Algorithm for RRTs

The Rapidly-exploring Random Tree (RRT) is a well known randomized algorithm to explore large state space in a relatively short time. The pseudocode of the algorithm is given in Algorithm 1. The algorithm chooses a point p



(a) (b) (c)

Fig. 4. (a) RRT basic algorithm applied to a point holonome robot in a known static environment; (b) Perception given by a distance sensor at the initial position: white, black and grey represent respectively free, occupied and occluded zones; (c) Probabilistic RRT built in limited time: the search tree and the likelihood of the nodes in blue (lighter colour is for lower likelihood) and the chosen partial path in red.

Algorithm 1: basic RRT

```
   Data: T
1  while q_goal ∉ T do
2      p = ChoosePoint(q_goal);
3      q = T.NearestNeigbohr(p);
4      q_new = extend(q, p);
5      if q_new ∈ C_free then
6          T.addSon(q, q_new);
7  end
8  q = q_goal;
9  path = add(q);
10 while q ≠ T.root do
11     q = T.parentNode(q);
12     path = add(q);
13 end
```

in the state space and tries to extend the current search tree toward that point. p is chosen randomly, but in single-query planning, some bias toward the goal is generally applied in order to speed up the exploration. p is chosen in the limited $\mathcal{C}_{free}$ (line 2). The nearest neighbour q of p within the nodes of the search tree is chosen for extension. A new node is obtained applying an admissible control from the chosen node q toward p (line 3). If q is collision-free, it is added to the tree. The algorithm can be stopped once the goal is found (line1) or it can continue to run to find a better path. The algorithm lies on a deterministic representation of the environment, so that both in the static and dynamic case we have a priori information on if a node is collision free or not and add it or not to the search tree. Once the goal state is reached, the path from the initial state to the goal is retrieved. Figure 4(a) shows a point holonome robot in a known environment with static obstacles. The initial position of the robot is in the left corner at the bottom while the goal is in the upper right corner. An example of the search tree (blue lines) and the found path (red line) is shown; different running of the algorithm would give different results. In this case, the robot is supposed to move along straight lines, so that the Euclidean distance can be used to determine the nearest neighbour in the current tree. The algorithm can be generalized for car-like robots setting a different $\mathrm{NearestNeighbor}(.)$ function. and limiting the set of possible actions to the admissible controls of the robot from the node configuration.

B. Introducing probabilistic uncertainty

As stated in previous sections, the robot knowledge about the environment is incomplete in both space and time (sensor range, occlusions, new moving obstacles) and uncertain (sensor accuracy, motion model of the moving obstacles). On the basis of the RRT algorithm we developed an exploring algorithm which takes into account probabilistic uncertainty. For each configuration q of the space, a probability of collision $P_c(q)$ is computed considering the static and moving obstacles and the perception limits as in equation 5. The probability of reaching a particular configuration q_N is then given by the probability to cross the tree from the root q_0 to the considered node, i.e. the probability of *not* having collision in each of the traversed nodes:

$$\begin{aligned} P_s(\pi(q_N)) &= P_s(q_0 ... q_N) \\ P_s(q_0 ... q_N) &= (1 - P_c(q_N)) \cdot P_s(q_0 ... q_{N-1}) \\ &= \prod_{n=0}^{N} (1 - P_c(q_n)) \end{aligned} \tag{7}$$

where we have considered that collision in subsequent nodes is statistically independent. We call this probability the probability of *success* P_s of the path. The probability falls exponentially with the length of the path. This is a sign that longer path are more dangerous, as the uncertainty accumulates over subsequent steps. All nodes than can be added to the tree, or a minimum threshold P_{smin1} can be chosen in order to avoid keeping in the tree very unlikely paths. Once a point p is chosen in the configuration space, the node to grow next q is chosen in dependence both on a measure of the expected length of the path $dist(q_0, q, p)$ and on the probability of success of the path. More precisely, $P_s(q_N)$ is normalized by the length N of the path and multiplied by the inverse of the distance to the chosen point to obtain a weight for each node. This normalization is taken out so that the probability of success doesn't depend on the length of the path, which is taken instead into account by the distance term, as in the following equation:

$$\tilde{w}_q = \frac{1}{dist(q0, q, p)} \sqrt[N]{P_s(q)} \tag{8}$$

The function $dist(q_0, q, p)$ is a sum of the length of the path from the root q_0 to the considered node and of the shortest path from q to p, which is a lower limit for the length of the eventual path to p. The weights are normalized over the set of nodes in the tree, and the result is a distribution over the nodes. The node to grow next is than chosen taking the maximum or drawing a random node proportionally to the probability. In our implementation we choose the second case which appeared to be more robust to local minima. Even if a path to the goal is found, the algorithm can continue to search for a better/safer path, until a path is asked for execution. However, is not guaranteed that a path that could be considered *safe enough* can be found even in infinite time, because of the environment uncertainty. The chosen path is then the best path that is safe enough, i.e. for which $P_s(q_N) \geq P_{smin2}$. Note that this threshold can be defined

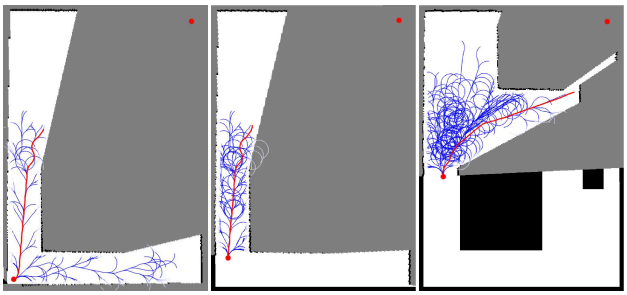


Fig. 5. Updating and growing the tree during environment exploration

only for the choice of the path or can be different from P_{smin2} if the same tree is updated and grown after different observation as we will explain in the next section, Figure 4(b) shows the perception given by a distance sensor in a static environment: areas behind the obstacles are unknown to the robot ($P_c \simeq 0.5$). Figure 4(c) shows the tree grown by the described algorithm for an holonome point robot. The colour of the edges of the tree depends on the likelihood of the associated path: the lighter the colour the lower the likelihood. In red, the best path chosen.

V. On-line navigation

A. Related work: the Partial Motion Planning

In a dynamic environment the robot has a limited time to perform planning which depends on the time-validity of the models used and on the moving objects in the environment. The conditions used for planning could be invalidated at execution time: for example an obstacle could have changed its velocity or some new obstacle could have entered the scene. The idea of Partial Motion Planning [18] is to take explicitly into account the real-time constraint and to limit the time available for planning to a fixed interval. After each planning cycle, the planned trajectory is generally just a partial trajectory. The exploring tree is updated with the new model of the world and the final state of the previous trajectory becomes the root of the new exploring tree. The planning algorithm works in parallel with execution. Each node of the tree is guaranteed to be not an Inevitable Collision State (ICS, [22]) by checking if it exists a collision free braking trajectory from the node. This is a conservative approximation that doesn't allow the robot to pass an intersection before an approaching moving obstacle. Our approach presents an adaptable time horizon for planning. The time for the planning iterations depends on the length of the previous computed trajectory and on the on-line observations. Safety of a path is guaranteed studying braking trajectories only for the last state of the path.

B. Developed Algorithm

When the robot moves, it observes the environment and updates its estimation with the incoming observations. The cost of crossing the tree changes and the tree needs to be updated. The update consists in three steps:

1) Prune the tree: the new root is the position of the robot and nodes that are in the past are deleted; the probability of reaching the nodes is updated, taking into account that the robot has already crossed part of the tree.
2) Update the weight of the nodes: when a change in the probability of collision is detected, the weight of the correspondent nodes (and of their subtree) must be updated.
3) Retrieve the best path.

If the considered environment is dynamic we need the robot to do these operations in real-time. In better words we need to know how much time is available for updating and how to allocate it. In the first step, the present state of the robot is considered. The tree is pruned so that only the subtree attached to the state of the robot is maintained. When the probability to pass from a configuration q_0 to q_i changes, the weight of the subtree attached to q_i is updated using the following equations:

$$P(q_N|q_i) = (P(q_N|q_0) - \hat{P}(q_i|q_0))\frac{1}{1-\hat{P}(q_i|q_0)} \quad (9)$$

$$P(q_N|q_0) = P(q_i|q_0) + (1 - P(q_i|q_0))P(q_N|q_i) \quad (10)$$

The first equation gives the probability of traversing the tree from q_i to q_N, assuming that the probability of reaching q_i changed from $\hat{P}(q_i|q_0))$ to 1. This equation is used once when the tree is pruned. This first update is due to the fact that the robot has already moved from q_0 to q_i, so that the new $P(q_i|q_0)$ is 1. In the equation q_0 is the old root, q_i is the new root and q_N is one node in the family of q_i. The second equation gives the probability to traverse the tree from q_0 to q_N when the probability to pass from q_0 to q_i changes from 1 to $P(q_i|q_0)$. Equation 10 is used after equation 9 when the observations revealed some difference with the prediction. The zones in which some difference have been detected are considered and the affected nodes are updated. In this case q_0 and q_i are respectively the start and ending configuration in which a change in the probability of collision has been detected.

In Fig. 5, the on-line updating of the tree is shown at 3 instants during navigation. At the beginning, the most likely paths are explored in the two possible directions and the most promising one is chosen. Fig.5(b) shows the tree after some steps: the tree has been updated: the branch in the right direction has been cut has is not reachable anymore and the tree has been grown. Fig.5(c) shows the tree and the new partial path found when a bigger portion of the space is visible

VI. Experimental Results

The planning algorithm has been tested with real data acquired on the car-like vehicle (Cycab) shown in Figure 1(a). To test the algorithm we define a goal 20 meters ahead the robot at each observation cycle and let the algorithm run in parallel with the online mapping and tracking (fig. 1(b)). The planning algorithm runs at 2Hz. The prediction used is the linear prediction given by the tracking algorithm. An

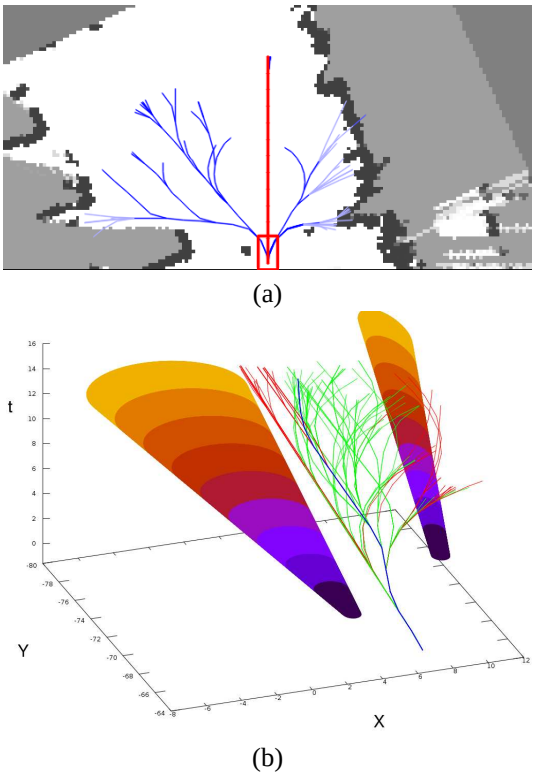


Fig. 6. (a) The RRT grow in a static occupancy grid: lighter blue is for lower likelihood. (b)The prediction of the moving obstacles and the explored tree in (x, y, t) space.

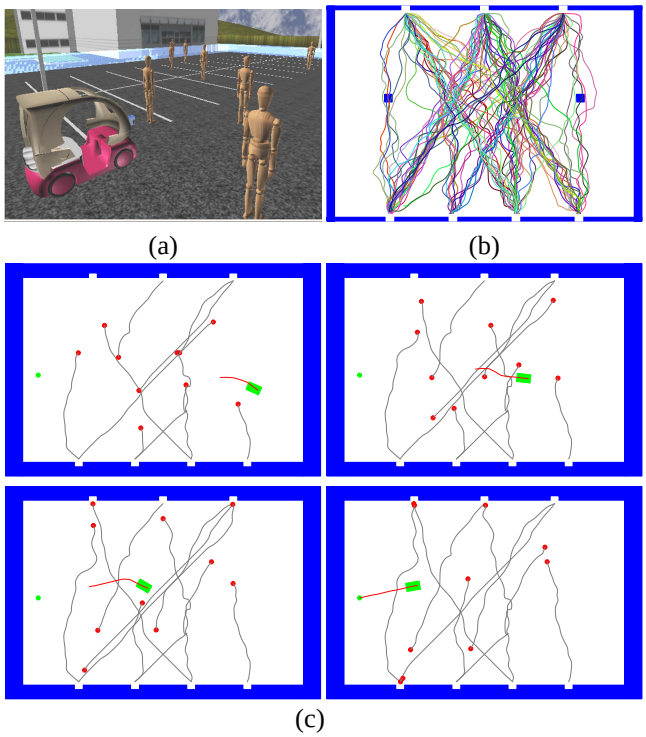


Fig. 7. Navigation results in simulated environment. (a) The Cycab simulator (b) The trajectory dataset. (c) Avoidance sequence based on HMM prediction.

example of the grown tree and the chosen path is shown in fig. 6. The occupancy grid correspondent to the figure is the one in fig. 1(b). The two cones represent the prediction of the two moving pedestrians considering one linear motion model and ellipses of axes correspondent to one covariance interval. A threshold has been applied to show different colours for safer (green) and dangerous (red) paths. The best path is shown in blue. Each sequence is then tested with the real data, letting a virtual robot move through the estimated map. Results have proven that the algorithm is able to compute safe trajectories in real time taking into account the static environment, the moving obstacles perceived and their velocity and the uncertainty which arise from a real dataset. However, the reliability of linear prediction is limited to a short time range, expecially for moving obstacles as pedestrians. The computed trajectories are safe only in the short period in which the prediction is reliable; the probabilities of the tree and the chosen path changes often as the obstacles change their directions. In the next simulated experiments we show that the use of typical patterns allow the robot for most robust planning and more intelligent decisions. The navigation algorithm has been tested in the Cycab simulator (7(a)). A rectangular environment has been simulated. A certain number of doors is simulated for the two long sides of the rectangle. Obstacles are supposed to enter from a door and to exit by another door in the opposite side. The space has been discretized in a uniform cell grid of step $0.5m$ An 4-connected HMM graph has been built on the grid for each goal: the probability to pass from a state to another depends on the decrease of the distance to the goal between the origin state end the destination one. A certain amount of noise is applied so that states that present nearly the same decrease in distance are given the same probability. The probability is then normalized over the set of edges coming out from the origin node. A set of trajectories has been randomly simulated on the basis of the graph: for each trajectory the enter door and the exit door are chosen (Figure 7(b)). Given a state of the obstacle, the next state is drawn proportionally with the edges probability. The position of the obstacle inside the cell is chosen by a smoothing filter.

The simulated robot has the same dimensions and kinematic and dynamic constraints of the Cycab. Perception is assumed perfect: the obstacles are represented by circles of $0.30m$ radius whose position is always known. The robot has to cross the environment and successively reach goals which are positioned randomly in the environment, with some bounds near the walls. The robot knows the Markov graph correspondent to the simulated trajectories and performs prediction on the basis of HMMs. The robot reached 1000 goals with various numbers of pedestrians simulated in the space. Figure 7(c) show the robot (green rectangle) traversing the environment to reach the goal: the red line is the partial path computed at the time-step in the shot, while red circles represent the moving obstacles with their previous trajectory attached. The robot reached 1000 goals with various numbers of pedestrians simulated in the space. No collision with the robot in motion was detected during the experiment, while the number of collisions as 0 velocity grows with the

number of objects in the space. To understand these results, we must notice that the simulated obstacles don't have any knowledge of the robot and that its kinematic possibilities are strongly limited if compared to those of the obstacles: as the robot cannot go backward, it tends to avoid obstacles and get stacked with the walls of the environment, while the obstacles continue to move around it. The two columns in Figure **??** show respectively the robot stopping to let an obstacle pass and the robot moving out from the possible paths of an obstacle before reaching the goal.

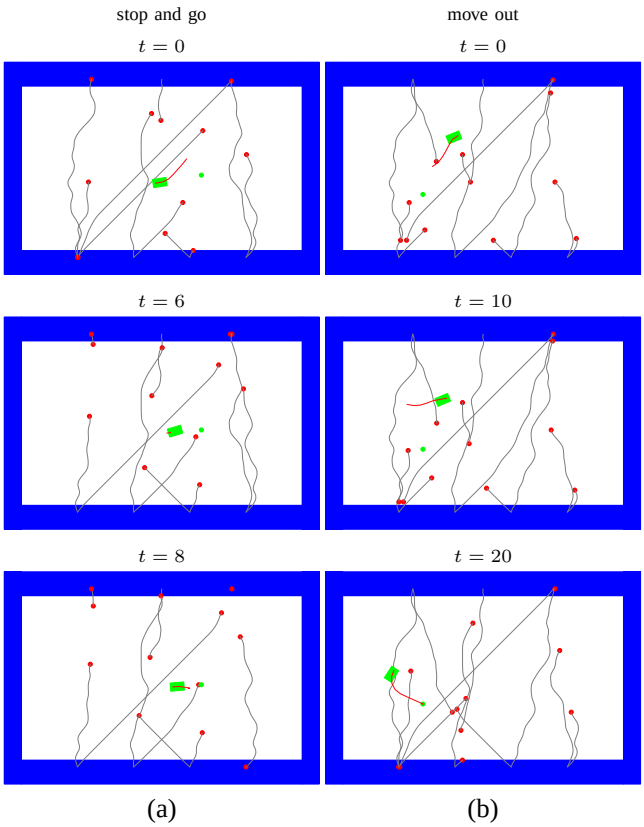


Fig. 8. (a) The robot stops to let an obstacle pass. (b) The robot moves out from the way of the obstacle before reaching the goal.

VII. Conclusion and future work

The paper presents a navigation algorithm which integrates perception uncertainty and incompleteness in the planning strategy using a probabilistic framework. The tests prove that the robot is able to navigate in real-time reacting properly to unexpected changes of the environment and reaching the given goal positions. The use of an adaptable time horizon for planning makes the algorithm both reactive to unexpected changes of the environment and *forward looking* when previously planned trajectories are not invalidated by observation.

Immediate work will deal with testing the navigation algorithm to have a measure of its performance in more complex and realistic scenarios. Future work will deal with the integration of the localization and execution uncertainty in the planning algorithm and with testing the navigation with the real robot.

References

[1] J. C. Latombe, *Robot Motion Planning*. Dordrecht, The Netherlands: Kluwer, 1991, vol. SECS 0124.

[2] S. M. LaValle, *Planning Algorithms*. Cambridge University Press (also available at http://msl.cs.uiuc.edu/planning/), 2006.

[3] J. Borenstein and Y. Koren, "The vector field histogram - fastobstacle avoidance for mobile robot," *IEEE Transaction on Robotics and Automation*, vol. 7, no. 3, June 1991.

[4] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *IEEE International Conference on Robotics and Automation, ICRA*, 1996.

[5] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, Mar. 1997.

[6] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *IEEE International Conference on Robotics and Automation, ICRA*, 1993.

[7] C. Stachniss and W. Burgard, "An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments," in *IEEE International Conference on Intelligent Robots and Systems, IROS*, 2002.

[8] F. Large, "Navigation autonome d'un robot mobile en environnement dynamique et incertain," Ph.D. dissertation, Université de Savoie, 2003.

[9] R. Simmons and S. Koenig, "Probabilistic robot navigation in partially observable environments," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI '95)*, July 1995, pp. 1080 – 1087.

[10] S. LaValle and R. Sharma, "On motion planning in changing, partially-predictable environments," *Int'l J. Robotics Research*, vol. 16, pp. 775–805, 1997.

[11] A. Foka and P. Trahanias, "Real-time hierarchical pomdps for autonomous robot navigation," *Robot. Auton. Syst.*, vol. 55, no. 7, pp. 561–571, 2007.

[12] M. Bennewitz and W. Burgard, "Adapting navigation strategies using motion patterns of people," in *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA*. IEEE, 2003, pp. 2000–2005.

[13] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun, "Learning motion patterns of people for compliant robot motion," *The International Journal of Robotics Research (IJRR)*, vol. 24, no. 1, 2005.

[14] P. Missiuro and N. Roy, "Adapting probabilistic roadmaps to handle uncertain maps," in *Proc. of IEEE International Conference on Robotics and Automation*, 2006, pp. 1261–1267.

[15] R. Benenson, S. Petti, M. Parent, and T. Fraichard, "Integrating perception and planning for autonomous navigation of urban vehicles," in *IEEE IROS*, 2006.

[16] N. Melchior and R. Simmons, "Particle rrt for path planning with uncertainty," in *IEEE ICRA*, April 2007.

[17] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, "Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes," *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 1056–1062, Sept. 2008.

[18] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *IEEE IROS*, 2005.

[19] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, pp. 46–57, June 1989.

[20] J. Burlet, "Adaptive interactive multiple models applied on pedestrian tracking in car parks," 2008, phD thesis.

[21] D. Vasquez, "Incremental learning for motion prediction of pedestrians and vehicles," Ph.D. dissertation, INP de Grenoble, February 2007. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2007/Vas07

[22] T. Fraichard and H. Asama, "Inevitable collision states. a step towards safer robots?" in *IEEE IROS*, 2003.