Probabilistic navigation in dynamic environment using Rapidly-exploring Random Trees and Gaussian Processes

Chiara Fulgenzi, Christopher Tay, Anne Spalanzani, Christian Laugier

Probabilistic navigation in dynamic environment using Rapidly-exploring Random Trees and Gaussian Processes

Chiara Fulgenzi, Christopher Tay, Anne Spalanzani, and Christian Laugier
INRIA Rhône-Alpes, LIG, France
Email: firstname.lastname@inrialpes.fr

***Abstract*— The paper describes a navigation algorithm for dynamic, uncertain environment. Moving obstacles are supposed to move on typical patterns which are pre-learned and are represented by Gaussian processes. The planning algorithm is based on an extension of the Rapidly-exploring Random Tree algorithm, where the likelihood of the obstacles 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 of the navigation algorithm for a car-like robot moving among dynamic obstacles with probabilistic trajectory prediction.**

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 (people, vehicles and the robot itself) moving in the space. 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 all 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 not taken 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 point of view, and are limited to low dimensional problems and to off-line planning. Some recent work proposes to integrate uncertainty in randomised techniques, such as Probabilistic Road Maps [12] and Rapidly-exploring Random Trees (RRT) [13] [14]. In this paper we address the problem of taking explicitly into account the behaviour of the obstacles, and the uncertainty in sensing and in prediction. We want our navigation algorithm to integrate new information coming from the acquisition system and to be able to react to the changes of the environment. We propose then a novel extension of the RRT algorithm to handle a probabilistic representation of the static environment and of the moving obstacles prediction. The search algorithm is then 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, [15]), where execution and local planning work in parallel to assure safe behaviour. In particular, we consider the case of a robot equipped with a distance sensor which models the static environment in an occupancy grid. The moving obstacles follow typical patterns with some amount of uncertainty: these patterns are a priori known and represented with Gaussian Processes (GP). The moving obstacles are detected and tracked on-line and the prediction of their future position is computed on the base of the known typical paths.

The reminder of this paper is structured as follows: first the way the trajectories are represented and predicted by Gaussian Processes is described in Section II. Section III recalls the RRT basic algorithm and details the new proposed approach. Section IV recalls the PMP method and describes the planning and navigation algorithm developed. Results are presented in Section V: an experiment with a laser scan

Fulgenzi Chiara is supported by a grant from the European Community under the Marie-Curie project VISITOR MEST-CT-2004-008270

dataset with moving pedestrians is presented in V-A and results in a simulated environment are shown in V-B. Section VI ends the paper with remarks and ideas for future work.

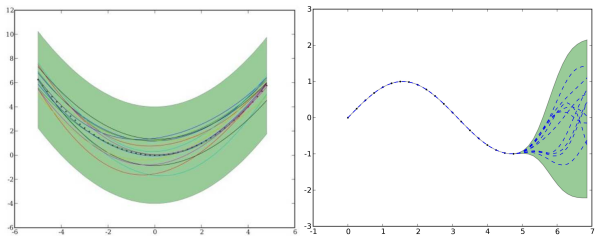
II. Gaussian Processes for pattern modelling

We consider an environment in which moving objects do not move around randomly, but follow typical patterns. This is a well suited hypothesis in many typical scenarios, as cars in a traffic cross junction or people in a transit area.

Most existing models for modelling motion patterns require the discretization of the state space [16]–[18]. These approaches mostly model motion trajectories as transitions between discretized states. The main disadvantage of discretization is the need to determine the discretization of the state spaces and the association of observations to these discretized state spaces.

In this paper, a path modelling approach based on Gaussian processes [19] is used. Gaussian processes make it possible to represent paths as continuous functions in a probabilistic manner. The problem of discretization is conveniently side stepped. Prediction on the future path can be performed in a theoretically proper probabilistic framework.

The advantage of using a Gaussian process to model a paths is its ability to give a Gaussian probability distribution over paths. Figure 1(a) shows an example of a hypothetical path exemplar. We call a path exemplar a certain motion path pattern in workspace. The lines represent sample paths from the Gaussian process distribution. A Gaussian process for an exemplar path incorporates the representation of variations around its mean path. We suppose that path exemplars can be represented as a Gaussian.



(a) (b)

Fig. 1. (a) Variance and mean of a typical path exemplar. Lines are sample paths from the Gaussian process. (b) Prediction on future paths with its mean and variances given a partial path. Dotted lines represents samples from distribution.

With a Gaussian distribution over a path exemplar, it is possible to obtain a mean and variance for possible future path motions. Figure 1(b) illustrates an example of a partially observed path and the prediction on future path with its variance indicated by the shaded area. Dotted lines in the figure are sample paths from this distribution.

A. Pattern Learning and Prediction

We consider the problem of learning a set of typical paths in a certain scene. In our approach, a training data set consists of motions executed by moving objects in the scene. An unsupervised learning approach is then applied to discover automatically the number of typical exemplar paths, along with a probabilistic description of the variations within each exemplar path.

As mentioned previously, an exemplar path can be represented using a Gaussian process and variations of paths belonging to the sample exemplar are Gaussian distributed. However, in realistic environments, there are often several exemplar paths. We model environments with several exemplars using a mixture of Gaussian processes. The training data can be viewed as realizations of the mixture of Gaussian process generative model and the aim of learning will be to recover the parameters of the mixture model.

Section II-B will give a short introduction to Gaussian processes, followed by a description of the probabilistic model for representing a single exemplar path (sect. II-C) and for situations involving several exemplar paths (sect. II-D). More information can be found in [20].

B. Gaussian Process

A gaussian process is a generalization of the gaussian probability distribution in function space. Given the set of gaussian distributed random variables $\{f(x_1), f(x_2), ..., f(x_N)\}$, it can be represented mathematically using the mean function and covariance function [19]:

$$f(x) \sim G(m(x), k(x, x')) \quad (1)$$

$$m(x) = E[f(x)] \quad (2)$$

$$k(x, x') = E[(f(x) - m(x))(f(x') - m(x'))] \quad (3)$$

Where $G(\mu, \Sigma)$ represents a Gaussian distribution with mean μ and covariance Σ. $k(x, x')$ is the covariance function with domains from the input space. A pertinent property of the covariance function is that it has to be positive semidefinite. In this paper, the squared exponential covariance function is adopted:

$$k(x, x') = \theta_0^2 exp - \frac{(x - x')^2}{\theta_1^2} + \theta_2^2 \delta(x - x') \quad (4)$$

Where $\delta()$ is the Dirac delta function. The chosen covariance function is stationary since it is a function of the difference, $x - x'$. Thus, covariance of paths are assumed to be the same throughout the input space. $\Theta = \{\theta_0, \theta_1, \theta_2\}$ are parameters for the covariance function, also known as the hyperparameters for the Gaussian process. The hyperparameters influence the form of the Gaussian process. Intuitively, the hyperparameters set a characteristic length scale for the Gaussian process covariance function and this length scale can be viewed as how much the difference $x - x'$ in input space has to be before there is a "significant" change in $f(x) - f(x')$. A short characteristic length scale gives a highly fluctuating Gaussian process whereas a longer length scale gives lower fluctuations.

C. Representing a typical path using Gaussian Process

We assume that the observation of paths belonging to an exemplar path are generated by a Gaussian process. Each path corresponds to a typical path that an object takes in

a given environment and is represented with 2 Gaussian processes, one each to represent the path in x and y axes as we assume the movements in the x and y axes to be independent. The mean of these Gaussian processes is the mean function (in the x and y axes each) representing the path.

A single observation of a path is represented as two vectors of dimension D, where $D$is the number of positions observed along the path. One vector represents the sequence of positions along the cartesian x axis and the other for the corresponding sequence in the y axis. The likelihood based on the N training data where each data is a sequence (x_n, y_n) is then:

$$L_x = \prod_{n=1}^{N} G(x_n|\mu_x, \Sigma_x) \quad (5)$$

$$L_y = \prod_{n=1}^{N} G(y_n|\mu_y, \Sigma_y) \quad (6)$$

Where x_n and y_n are vectors of x and y positions for the n^{th} observation. μ_x, μ_y, Σ_x and Σ_y are the mean vectors and covariances of the x and y positions for the typical path. In almost all cases, the sequence of observations of positions are of different length for different path observations. In this case, a fixed dimension D can be chosen and the D positions can be obtained by choosing D points uniformly distributed along the interpolated path.

D. Multiple Typical Trajectories using Gaussian Mixture Model

Since each observed path corresponds to two D dimensional vectors and that a typical motion path is Gaussian distributed, the observed paths are Gaussian distributed from the generative point of view. A single typical motion path is a D dimensional Gaussian distribution and this can be easily extended to the case of representing several typical motion paths using a mixture model. Considering K components, the likelihood based on the N training data is then:

$$P(x|z, \mu_x, \theta) = \prod_{n=1}^{N} \prod_{k=1}^{K} G(x_n|\mu_{x,k}, C(\theta_k))^{Z_{nk}} \quad (7)$$

$$P(y|z, \mu_x, \theta) = \prod_{n=1}^{N} \prod_{k=1}^{K} G(x_n|\mu_{y,k}, C(\theta_k))^{Z_{nk}} \quad (8)$$

Where z is the vector of component weights, θ the Gaussian process hyperparameters, μ the mean function of the Gaussian process, and $C(\theta_k)$ the covariance matrix of the Gaussian process parameterized by θ. The complete model used is a hierarchical probabilistic model where random variables include cluster component weights and cluster means unlike the standard Gaussian mixture model. As the hierarchical model is intractable, approximate learning methods are used. For practical purposes, it is desirable to have point estimates for the cluster component weights and the hyperparameters of the Gaussian processes. This leads to a learning approach closely related to Expectation-Maximization (EM) where in the E-step, variational mean field type learning is used. And in the M-step, maximization of the log likelihood over the space of parameters is performed. To determine the number of cluster components, training is performed by assuming a large number of cluster components. The training result gives small weights for certain clusters, and the clusters with small weights are be removed. For details refer to [20].

E. Prediction

When performing path prediction, the input is a partially observed path of dimension $M < D$. For the case of a D dimensional Gaussian with x_1 of dimension M and x_2 of dimension $D - M$:

$$P'(x_1, x_2) \sim G\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\right) \quad (9)$$

The probability of a partial path observation of dimension M belonging to a Gaussian of dimension D is evaluated by integrating over the $D - M$ dimensions of the Gaussian distribution to yield the marginal Gaussian distribution:

$$P'(x_1) \sim G(\mu_1, \Sigma_{11}) \quad (10)$$

The prediction of a path x_2 given observation x_1 can be obtained by the Gaussian conditional distribution for each cluster k:

$$P'_k(x_2|x_1) \sim G\left(\mu_2 + \Sigma_{21}\Sigma_{11}(x_1 - \mu_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{21}^{T}\right) \quad (11)$$

In order to choose the suitable clusters that corresponds to the observations made so far, the Mahalanobis distance can be calculated and then gated based on the appropriate chi-square values. Figure 2 shows an example of the prediction

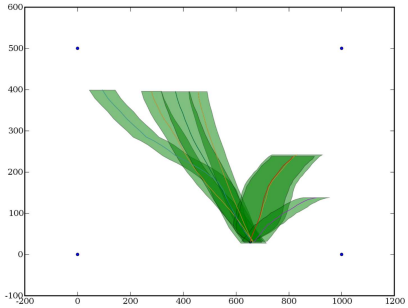


Fig. 2. Prediction paths and path variance to 2 standard deviations

where the predicted path mean and variance are represented by the 'bars'. Clusters for prediction were selected according to the chi-square statistic corresponding to the 95% confidence interval. For each cluster, the Gaussian distribution of the predicted path can be obtained using eq. 11.

III. PROBABILISTIC RRTS

Consider a car-like robot moving in an unknown environment among static and moving obstacles. The task of the robot is to reach a given goal state avoiding collisions. The robot perceives its surroundings with a distance sensor (laser range finder) and is able to detect and track the moving obstacles in its view range. At each instant the robot

knowledge about the world is incomplete and uncertain: incomplete in space because of the sensor range and the hidden areas and in time because of the limited validity of the motion models of dynamic obstacles and the unpredictability of new obstacles entering the scene. The uncertainty instead comes from the sensor error and accuracy and from the error of the motion models, the detection and tracking algorithms. Spatial uncertainty and incompleteness are represented by a probabilistic occupancy grid. In [13] the RRT algorithm is integrated in a *worst case* approach: obstacles are tracked in a dynamic environment and their kinematic model is used to bound the prediction of their future position. Also, partial planning is performed taking into account real-time constraints and the reliability of prediction. In [14] a particle filter is combined with an RRT method to perform path planning in environment with uncertain slipping: an a priori model of uncertainty is defined and planning is performed in a static environment.

In our paper we propose an algorithm to take into account a probabilistic representation of sensing and prediction uncertainty which can be updated by new incoming information. Spatial uncertainty and incompleteness are represented by a probabilistic occupancy grid. Temporal uncertainty is represented using Gaussian mixture predictions.

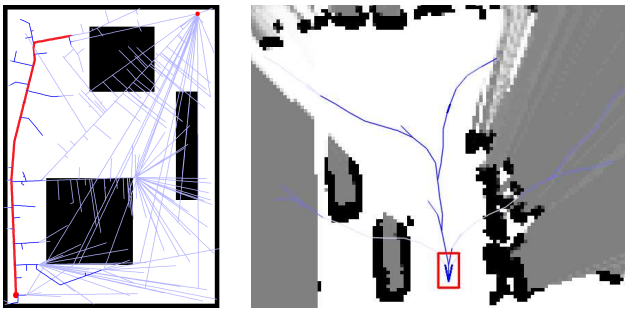
A. Probability of Collision

At a given instant, the robot knowledge about the state of the environment is represented by:

1) a list of pre-learned GPs which represent the typical patterns of the moving obstacles;
2) an occupancy grid, which represents the structure of the static environment around the robot, according to the previous observations;
3) a list of moving objects, their estimated position, velocity and previous observations;
4) an estimation of the state of the robot itself;
5) a goal state.

The configuration of the robot $q = (x, y, \theta, v, \omega, t)$ at a time instant t is given by the position (x, y) and orientation θ of the robot in the plane, the linear velocity v, the angular velocity ω. For each configuration q, a probability of collision $P_c(q)$ can be computed considering the static and moving obstacles and the perception limits.

The probability of collision with the static environment P_{cs} is computed considering the maximum probability of occupation among the cells of the occupancy grid touched by the robot in that configuration. Let's consider M moving objects and K the number of GPs associated to the typical patterns learned. For an obstacle O_m, the predicted position at time t is estimated by a gaussian mixture of K components. Considering each component k separately, the associated probability of collision $P_{cd}(k, m)$ is calculated considering the overlapping of the robot and the predicted Gaussian. In practice, the probability of collision is computed in a set of points in the area occupied by the robot and the maximum is considered.



(a) (b)

Fig. 3. Probabilistic RRT in static environment. Figures show the search tree and the likelihood of the nodes (lighter colour is for lower likelihood). Observations are taken with a distance sensors, and there are occluded zones. (a) A point holonome robot in a simulated environment; (b) A car-like robot in an occupancy grid.

B. The RRTs

The Rapid-exploring Random Tree (RRT) [21] is a randomized algorithm to explore large state space in a relatively short time. The algorithm chooses a point P in the configuration space and tries to extend the current search tree toward that point. P is chosen randomly, but generally in single-query planning, some bias toward the goal is applied in order to speed up the exploration. The nearest node neighbour of P within the nodes of the search tree T, is chosen for extension. A new node is obtained applying an admissible control from the chosen node s toward P. If the node is collision free the new node is added to the tree. The algorithm can be stopped once the goal is found, or it can keep on running to find a better path. Once the goal state is reached, the path from the initial state to the goal is retrieved.

C. Probabilistic Extension

The algorithm described above lies on a deterministic representation of the environment, i.e. the algorithm knows a priori if a node is collision free or not. If the environment is not completely known, or if there are moving obstacles with unknown trajectories, we would need to make recursion to a *worst case* approach. This could limit the search and prevent the algorithm from finding any solution or probable solution in most cases. In this paper, we propose an extension of the basic RRT algorithm to take into account the probability of collision both during the exploration of the space and at the choice of the path, so that the probability of non-collision of a path becomes a measure of its feasibility. All the explored configurations are maintained in the tree and their probability of collision is updated by more recently acquired information.

Given an initial configuration s_0, and a search tree T, the probability of collision of a path $\pi(s_N) = \{s_0 ... s_N\}$ is given by the probability to cross the tree from the root s_0 to the considered node s_N. This is computed as the probability of *not* having collision in each of the traversed nodes. For a

single obstacle m and a Gaussian process k :

$$P_\pi(s_N, m, k) = 1 - \prod_{n=0}^{N} (1 - P_{cd}(s_n, m, k)) \quad (12)$$

The probability falls exponentially with the length of the path. This is a sign that longer paths are more dangerous, as the uncertainty accumulates over subsequent steps. Considering first all the GPs for one moving object and then integrating on all the objects we have:

$$P_\pi(s_N, m) = \sum_{k=1}^{K} l_{k,m} \cdot P_\pi(s_N, m, k) \quad (13)$$

$$L_\pi(s_N) = \prod_{m=1}^{M} (1 - P_\pi(s_N, m)) \quad (14)$$

where we considered independent collision events for each obstacle. L_{π_N} is the probability that the robot traverses the path without entering in collision and we call it the probability of success or likelihood of the path.

Once a point P is chosen in the configuration space, we have to decide which node to grow next. For each node s, a weight w_S is computed considering the expected length of the path $dist(s_0, s, P)$ and the probability of success. More precisely, $L_\pi(s)$ is normalized by the length of the path and multiplied by the inverse of the distance. This normalization is taken out so that the probability of success does not depend on the length of the path, that is taken into account by the distance term:

$$\tilde{w}_{s_N} = \frac{1}{dist(s_0, s_N, P)} \sqrt[N]{L_\pi(s_N)} \quad (15)$$

$$w_s = \frac{\tilde{w}_s}{\sum_s \tilde{w}_s} \quad (16)$$

The function $dist(s_0, s_N, P)$ is a sum of the length of the path from the root s_0 to s_N and of the shortest path from s_N 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 (16). The node to grow next is then chosen taking the maximum over the weights or drawing a random node proportionally to the weight. In our implementation we choose the second approach which appeared to be more robust to local minima. Even if a path to the goal is found, the algorithm continues to search for a better/safer path until the available time is out. All nodes are added to the tree, or eventually a threshold P_{safe} can be chosen to avoid the exploration to move toward too unlikely zones.

Figure 3(a) shows of the described algorithm for an holonome robot within a static simulated environment. The initial position of the robot is in the bottom left corner, while the goal is in the upper right corner. Black rectangles are obstacles. The robot perceives the environment with a distance sensor, so that areas behind the obstacles are unknown ($P_c \simeq 0.5$). The colour of the edges of the tree depends on the probability of success of the associated path: the lighter the colour the lower the probability. In red, the path chosen. Figure 3(b) shows the described algorithm for a car-like robot within an observed occupancy grid.

IV. On-line navigation

A. 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 behaviour or some new obstacle could have entered the scene. The idea of Partial Motion Planning [15] 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. The developed solution

At the beginning of navigation, we suppose that the robot has some time to observe the environment and to plan before moving. From the initial state s_0, the tree is grown. At the end of the available time, the tree have reached deep N. At each of the explored partial paths a weight is associated using equations 15 and 16. The best path (with the higher weight) is picked out. To guarantee that the robot does not choose a dangerous path, we can specify the maximum allowed risk using a safety threshold. Also, from the last state of the path, a safe braking trajectory is searched. If it exists, the path is considered safe and passed for execution, otherwise it is discarded and another one is chosen from the tree. The chosen path is a sequence of N_2 configurations to be attended, where $0 \leq N_2 \leq N$. While the robot executes the path, the tree is updated and grown. At each timestep, the nodes of the tree that are not reachable (in the past) are pruned out and the probability of success of each path is updated with the new estimation of the likelihood of each GP, given by observation following equation 13 and 14. In the remaining available time, the tree is grown. At the same time, the robot acquires new observations and compares the planned path with the new estimation: if the path is still safe, the robot continues to follow it; if an unexpected event is detected, a new safe path is retrieved from the search tree. If no safe path is found within the available time, the robot proceeds to a braking manoeuvre. In a very unpredictable environment, the time horizon will be shortned by the unexpected events and the algorithm will act like a

purely reactive algorithm. On the opposite, in the case of static known environment, the algorithm will allow longer and longer time to grow the tree, and it the result would be nearer to the solutions of a global planner.

V. Experimental Results

A. Laser Data Set

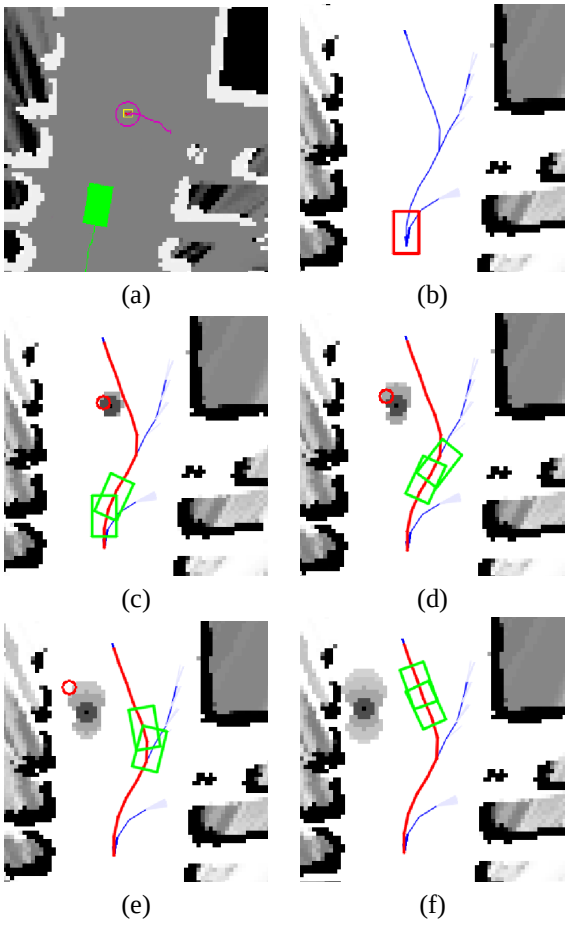


Fig. 4. Results with a laser dataset. (a) The static environment is mapped and the moving obstacles are tracked. (b) The algorithm explores the state space and chooses a path. (c-e) the path is compared with the real observations acquired.

The algorithm has been tested with real data acquired on a car-like vehicle equipped with a laser range finder (Cycab [23]). During the experiment, the robot is manually driven in an outdoor environment and perceives static obstacles and moving pedestrians. Using the algorithm developed in [24], [25] the robot localizes itself, builds an occupancy grid map of the static environment and tracks the moving obstacles. The probabilistic predictions of the future state of the obstacles are computed using a constant velocity motion model and the position and velocity estimated, with their associated covariance. To test the planning algorithm we define a goal 20 meters ahead the robot at each observation cycle and let the algorithm run in parallel with the on-line mapping and tracking at 2Hz (fig. 4(b)). Each sequence is then tested with the real data, letting a virtual robot move through the map (fig. 4(c-e). Fig.4(a) shows the observed occupancy grid: free space is grey, occupied space white, non-explored space is black. The robot is the green rectangle and the circle is the tracked pedestrian with his trajectory. Fig.4(b) shows the tree of states explored in the available time: lighter blue is for higher probability of collision. The red line is the chosen path. Fig.4(c-f) shows subsequent positions of the virtual robot; on the background the occupancy grid predicted for planning; red circles represent instead the position of the moving obstacles as estimated in real-time by the tracking algorithm. Results prove that the algorithm is able to compute safe trajectories in real time taking into account the static and moving obstacles perceived and the uncertainty in prediction of a real data set.

B. Simulation Results

Tests have also been conducted in a simulated environment. A set of 3000 trajectories has been simulated in a rectangular environment. A subset of 1000 trajectories have been used as training data set and, as a result, 26 Gaussian processes have been learned.
To test the probabilistic planning, we simulate the robot navigating among circular obstacles with trajectories that are chosen randomly from the simulated set. The static environment is supposed to be free and the perception of the robot is simulated. The timestep chosen is of $0.5s$. Planning and execution run in parallel. Figure 5 shows some snapshot from the obtained results. The robot is the red rectangle and perceives the circular obstacle (red full point). The goal of the robot is at the bottom of the image (black circle). Green paths represent the mean of the Gaussian Processes: the likelihood of each GP is estimated at each time step on the basis of the previous observations; lighter colour is for lower likelihood. The tree explored by the robot is given by the blue lines. Again, lighter blue means lower likelihood. Circles represent the prediction for the obstacle for each associated Gaussian process and at subsequent timesteps. Fig. 5(a) shows the planning at the first timesteps; in figure 5(b) the robot moves toward the goal, while the obstacle moves toward the upper left corner and the prediction gets better. around one GP only. In figure 5(c) the moving obstacle has disappeared behind the robot, while another one appears near the goal. Fig. 5(c), shows how the search tree is grown and the path is adapted to the new situation.

VI. Conclusions and Future Work

The paper presents a navigation algorithm which integrates perception uncertainty and incompleteness in the planning strategy using a probabilistic framework. The use of pre-learned typical patterns allows to have a medium-term prediction that is more reliable than what can be expected from a simple target tracking algorithm and that can easily handle non-linear motion. The integration of the likelihood of obstacle paths and of the probability of collision makes it possible to update previously explored states with the on-line estimation; this gives in result less dangerous and more intelligent paths for the robot. The use of an adaptable time

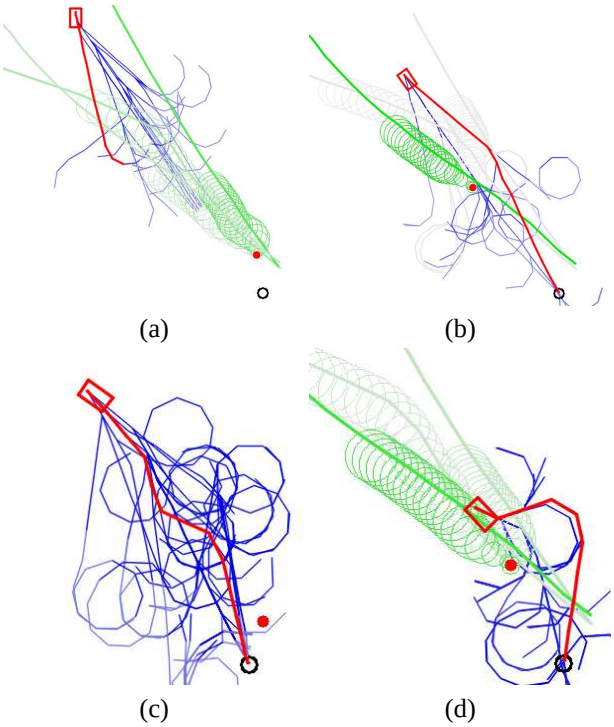


Fig. 5. The robot moves in a simulated environment with a moving obstacle. The prediction of the obstacle is given by a Gaussian mixture based on the pre learned Gaussian processes (green). The exploring tree maintains an estimation of the likelihood of the path that adapts to the incoming observation.

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. Also we will focus our study to reduce the complexity of the algorithm discarding unlikely Gaussian processes and obstacles that cannot interfere with the trajectory of the robot. Ideas for future work are directed to integrate in one framework the prediction coming from target tracking, which is more accurate in short term and the prediction coming from the pre-learned Gaussian processes, which is more reliable at medium term.

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*, April 1996, pp. 2833–2838.

[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, vol. 2*, Atlanta, 1993, pp. 802–807.

[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*, Lausanne, October 2002, pp. 508–513.

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

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

[14] N. Melchior and R. Simmons, "Particle rrt for path planning with uncertainty," in *2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 1617–1624.

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

[16] V. D, F. Th., A. O., and L. C., "Intentional motion on-line learning and prediction," in *Proc. of the Int. Conf. on Field and Service Robotics*, Port Douglas (AU), July 2005. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2005/VFAL05

[17] 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.

[18] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, "A system for learning statistical motion patterns," *IEEE PAMI*, vol. 28, no. 9, September 2006.

[19] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[20] C. Tay and C. Laugier, "Modelling paths using gaussian processes," in *Proc. of the Int. Conf. on Field and Service Robotics*, 2007. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2007/TL07

[21] S. LaValle and J. Kuffner, J.J., "Randomized kinodynamic planning," vol. 1, 1999, pp. 473–479 vol.1.

[22] T. Fraichard and H. Asama, "Inevitable collision states. a step towards safer robots?" in *IEEE IROS*, vol. 1, Oct. 2003, pp. 388–393.

[23] C. Pradalier, J. Hermosillo, C. Koike, C. Braillon, P. Bessière, and C. Laugier, "The cycab: a car-like robot navigating autonomously and safely among pedestrians," *Robotics and Autonomous Systems*, vol. 50, no. 1, pp. 51–68, 2005.

[24] N. A. Trung-Dung Vu, Olivier Aycard, "Online localization and mapping with moving object tracking in dynamic outdoor environments," in *IEEE Intelligent Vehicles Symposium*, Istanbul, 2007.

[25] J. Burlet, "Suivi multi-objets adaptatif : Application la classification de comportements de mobiles," Ph.D. dissertation, Institut National Polytechnique de Grenoble, France, December 2007.