Risk based motion planning and navigation in uncertain dynamic environment

Chiara Fulgenzi, Anne Spalanzani, Christian Laugier, Christopher Tay

Risk based motion planning and navigation in uncertain dynamic environment

Chiara Fulgenzi, Anne Spalanzani, Christian Laugier and Christopher Tay

***Abstract*—Navigation in large dynamic spaces has been often adressed using deterministic representations, fast updating and reactive avoidance strategies. However, probabilistic representations are much more informative and their use in mapping and prediction methods improves the quality of obtained results. The paper proposes a new concept to integrate a probabilist collision risk function linking planning and navigation methods with the perception and the prediction of the dynamic environments. Moving obstacles are supposed to move along typical motion patterns represented by Gaussian Processes. The likelihood of the obstacles' future trajectory and the probability of occupation are used to compute the risk of collision. The proposed planning algorithm is a sampling-based partial planner guided by the risk of collision. The perception and prediction information are updated on-line and reused by the planner. The decision takes into account the most recent estimation. Results show the performance for a robotic wheelchair in a simulated environment among multiple dynamic obstacles.**

***Index Terms*—autonomous navigation, dynamic environment, probabilistic planning, typical patterns, Gaussian Processes.**

I. Introduction

AUTONOMOUS navigation in populated environments is one of the most challenging topic in robotics research. Unlike static or controlled environments where global planning is suitable, dealing with highly dynamic and uncertain environments requires to address 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 must take into account all the sources of uncertainty involved: perception accuracy, modeling of the world, configuration of the robot.... In the last few years, the problem of incomplete, uncertain and changing information in the navigation problem domain has gained even more interest in the robotic community and probabilistic frameworks aiming to integrate and elaborate properly such information have been developed.

The problem addressed in this paper is the autonomous navigation in an uncertain and dynamic environment. The purpose is to develop techniques allowing a robot to move autonomously and safely in an environment which is not perfectly known *a priori* and in which static and moving obstacles are present. The task of the robot is to find and execute a sequence of actions to reach a given position, avoiding collisions with the obstacles. We aim to give a robot the possibility to exploit the fact that pedestrians and vehicles usually do not move at random in the given environment but often engage in typical behaviors or motion patterns. The robot may use this information to better predict the future position of these moving obstacles and adapt its behavior accordingly.

A. Realistic Perception

We address particularly the problem of putting in relation the decision and action process with a realistic perception input, assuming that the robot has little *a priori* knowledge about the static environment and about the surrounding moving obstacles. We investigate the problems and limits inherent to sensor perception and future prediction and focus on environmental models that best express the changing information and uncertainty coming from perception. We focus on the following considerations:

- The *uncertainty* and *incompleteness* of the information perceived by the robot is not negligible and some mean to take it into account into the decision process should be introduced;
- The fact that the environment is *dynamic* cannot be ignored: the robot performance is influenced by the fact that obstacles move in the environment and the robot should be able to take safe and good decisions at anytime and act promptly in the dynamic environment.

The unknown dynamic environment is perceived and mapped by the robot during navigation. To properly map the uncertainty and incompleteness of the sensing information regarding both static and dynamic environments probabilistic representations are chosen. The static environment is mapped by an occupancy grid [1]; moving obstacles are detected and their dimension, position and velocity are estimated thanks to a target tracking method

B. Probabilistic prediction: Gaussian processes for typical patterns representation

In a given environment, pedestrians and vehicles often engage in typical behaviors or motion patterns. Supposing that the environment has been observed for enough time and that the typical patterns have been learned, the information gathered provides a more reliable prediction in the medium and long term with respect to a simple linear and conservative model [2], and an hint of the zones from where new obstacles are likely to enter the scene. In our planner the typical trajectories are represented by Gaussian Processes [3]. This representation provides a prediction which is continuous in the space and time dimensions. These issues, together with the meaningful probabilistic framework with which the prediction is generated, enhance the performance of the planner over kinematics based prediction.

C. Risk-RRT: motion planning based on the risk of collision

Risk-RRT (Risk Rapidly exploring Random Tree) is the new approach we propose for planning and navigation based on the following concepts:

1) **Probabilistic risk of collision**: we consider unknown dynamic environments. The probabilistic risk of collision is computed on the basis of the probabilistic models which map the static and the dynamic obstacles.
2) **Risk guided search**: the search of safe paths is guided by the risk of collision. The search method is based on the well known RRT framework [4].
3) **Real-time decision update**: the search algorithm is integrated in an anytime planning and re-planning approach: the probabilities of collision and the decisions of the robot are updated on-line with the most recent observations.

Results show how the navigation algorithm deals with unknown static environment and multiple moving obstacles with uncertain trajectories.

This article is structured as follows : in Section II, related works on navigation and planning are presented. Section III describes the perception and prediction models used and how the probabilistic risk of collision is evaluated (see point 1) Section IV describes the planning algorithm developed (point 2) and its integration in real-time navigation strategy (point 3). Section V shows results for a two wheeled robot among simulated pedestrians. Section VI discusses conclusions about the work and Section VII present future work and perspectives.

II. Related works

The aim of this work is to propose a method to navigate autonomously among static and moving obstacles while handling the uncertainty due to the robot's perception of the environment and the real time constraints due to the dynamic and evolutive nature of the information. Let's consider the following classification of sources of uncertainty [5]: Configuration Sensing (CS), Configuration Predictability (CP), Environment Sensing (ES) and Environment Predictability (EP) uncertainty. A real-world system is usually affected by all four of these sources, but depending on the quality of the mechanical system (sensors, actuators) and of the complexity of the environment and the corresponding models, one or the other source affects the navigation problem with more or less importance.

The problem presents the following characteristics:

1) The state of the robot is known.
2) The future configuration of the robot deterministically depends on the current configuration and the applied controls.
3) The static environment is unknown. The occupation of observed parts of the environment is probabilistically estimated.
4) There are obstacles that move in the workspace and that can appear during navigation.
5) The present state of moving obstacles must be estimated.
6) The future position of moving obstacles can only be predicted according to an estimated motion model and known typical environmental patterns.

We address the environmental sensing uncertainty ES (see points 3-4) and environmental prediction uncertainty EP (see points 5, 6). We will discard the configuration sensing CS and configuration prediction CP uncertainty (respectively points 1 and 2). As the environment is dynamic, we need to address also the time-constraint problem, making use of an anytime algorithm. Table I synthesizes the properties needed (+) by our navigation algorithm.

	ES	EP	CS	CP	Dynamic
Needed Properties	+	+	-	-	+

TABLE I
PROPERTIES NEEDED BY THE NAVIGATION ALGORITHM IN UNKNOWN DYNAMIC ENVIRONMENTS.

The next paragraphs revise the main approaches developed to face the various issues of the problem. Paragraph II-A summarizes some of the main approaches used to represent typical motion patterns; paragraph II-B presents some literature on navigation among obstacles presenting typical motion; paragraph II-C synthesizes the state of the art in navigation in dynamic environment; paragraph II-D presents the approaches where probabilistic uncertainty has been taken into account. Finally, paragraph II-E discusses the different methods and introduces the contribution of this paper.

A. Typical motion models

The learning of typical patterns and the representation of pattern based motion models has been the subject of extensive study and many different approaches have emerged. Typical trajectories are usually represented as a sequence of points in the continuous state-space. Most approaches do not model the time variable explicitly and assume that the points in the sequence are regularly spaced in time. Sometimes, a measure of the "width" of the cluster is also included in the representation [6], [7]. In [8] a probabilistic model is proposed in which the width of the trajectory is represented as the variance of the distance between the trajectories that belong to the same cluster. Another probabilistic model of width has been proposed by [9]: every point of the trajectory prototype is modeled as a Gaussian and it is assumed that all such Gaussians have the same covariance matrix. A novel approach has been proposed by Tay in [10] where trajectories are represented by Gaussian Processes. In this case both the typical trajectory and its "width" (mean and covariance) are probabilistically estimated by a proper Bayesian framework. The advantages of the Gaussian Processes representation is that they present a solid probabilistic theory for the representation of the mean and covariance of the different paths and the future prediction. Also, trajectories are represented by continuous functions, which allows to use different time steps for prediction and for observation, thereby limiting the necessary interpolations at the learning phase. Finally, the

Gaussian representation allows the prediction to be very fast and computationally cheap.

B. Navigation using pattern based motion models

Literature on navigation with pattern based motion models is quite poor. An early work on navigation in changing environments and in presence of typical motion suggests to divide the state space in *hazardous* and *shelter* regions [5]. A shelter designates an area in which the robot is guaranteed to avoid collision, while an hazardous region designates an area in which other obstacles can move. The cost of traversing an hazardous or dynamic region directly corresponds to the risk of encountering a moving obstacle.

In more complex environments however, this representation may reveal too simplistic: there may be no shelter areas at all, or they can be interleaved with the hazardous ones, so that having a spatial and temporal hint of where moving object actually are becomes a necessity.

In [11], the robot applies an A^* algorithm to find a path on a 2D space cost grid: the cost of passing through a cell at time t is given by the probability of collision plus the probability that a person covers the cell at that time. The algorithm is applied in an office-like environment where each typical pattern is represented by a fixed number of Gaussians that specify the probability of a position at a stage of the trajectory. Re-planning is performed whenever information changes. In [12], the use of these motion models is shown to improve the navigation performance in comparison with the case of a model based only on target tracking.

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 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.

C. Motion Planning in dynamic environments

Navigation methods are usually classified in path planning approaches, which look for a complete path from the initial configuration to the goal and reactive approaches, which just look for the next action of the robot. A global planner is usually combined with a reactive obstacle avoidance or trajectory deformation techniques [13], [14] when operating in partially known and dynamic environments. Most of the navigation methods described in literature rely on a deterministic representation of the world. Pure globall planning methods assume that the trajectory of the moving obstacle is known *a priori* and skip the problem of uncertainty: either the information gathered is assumed as *true* or a worst-case approach is used. Under these assumptions, the global path planning methods used in static environments are extensible to the dynamic case just by adding the time dimension. This is the case of optimal combinatorial algorithms, complete or discretized ([15], [16], [13]) and sampling based algorithms ([17], [18], [19], [20], [4]).

In reactive techniques, either the obstacles are assumed to be static and the algorithm relies on the high frequency of iteration to face dynamic obstacles [21] or it is assumed that the velocity is known and constant [22].

As a hybrid method, we can cite *Anytime RRTs*, proposed by [23]: the algorithm plans a complete path at the beginning using the classical RRT method [4]. If the path is invalidated by the observations at execution time, the tree (which is maintained in memory) is locally "repaired": the invalid nodes are deleted, while new configurations are searched to repair the tree from the root to the branches that have been isolated by the unexpected obstacles. In *Partial Motion Planning (PMP)* [24] the time constraints of the dynamic environment are explicitly taken into account: a search tree is grown using a sampling based algorithm and a state is added to the tree only if it is safe with respect to the known trajectories of moving obstacles. During execution of the partial path selected, another partial path is elaborated starting from the end of the previous elaborated path. In this way, PMP is able to give a safe partial path at anytime.

D. Motion Planning with probabilistic uncertainty

While a probabilistic model requires much more information than a deterministic one (since it is based on statistics), it is also much more expressive and it is able to quantify the risk of the robot in each configuration. The probabilistic representation has been exploited so far only for low dimensional problems.

The state of the system and the control space are discretized and uncertainty is modeled as an action taken by a particular agent called *nature*. This action influences the next state of the system and is unknown. If the state of the system is perfectly observable at each instant, the problem is referred as a Markov Decision Process (MDP) Solving the problem means finding the optimal strategy through a weighted graph where nodes represent configurations and branches represent actions. Each action is given a cost, which is function of the action of *nature* and represents the risk that the action is not successful. The problem can be solved applying graph search algorithms, like A^* and value iteration. If also the state of the system is not perfectly observable, the problem can be formulated as a Partially Observable MDP (POMDP). This new class of problems is however much complex and harder to solve.

When the *a priori* knowledge on the workspace and on the obstacles motion model is too poor, the complexity of the problem is too high to be addressed with a complete algorithm in reasonable time. If the world is static, the robot may interleave planning and execution phases, as it can stop and has time to plan a new path when something different or unexpected is perceived. In order to slim the re-planning phase, some algorithms have been proposed to take advantage of the previous planning phases. D^*, proposed in [25], is a generalization of the A^* in the case of partially known environments. The optimal path is found with the initial information; if a change is detected, only the affected configurations are considered and the optimal path is updated in a reduced time.

Only in more recent works sampling-based methods, which are more suited to high dimensional problems, have been ex-

tended to take into account probabilistic uncertainty: *Particle-RRTs* [26], for instance, extend the RRT algorithm to an environment with uncertain slipping conditions. A probabilistic motion model is considered, dependent of the unknown friction of the terrain. Each vertex of the search tree is a cluster of the possible configurations of the robot sampled from motion model, as in particle filters.

E. Contribution

Table II shows some of the main approaches in state of the art, if they are able to represent (+) the various kind of uncertainty or not (-) and their answer in terms of time constraint, i.e. if they are adapted not only to unknown environments but especially to navigation among dynamic obstacles. Optimal methods take into account probabilistic uncertainty, but their complexity scales badly with the dimensions of the environment. This can prevents them to be used among unknown moving obstacles. Also, in our case, the dimension of the space is initially unknown and changes during the task. Finally, the discretization required, especially in the action space, is not adapted to maneuver in cluttered environments. In particular the POMDP formulation, which presents the largest range of uncertainty representation, also requires the discretization of the observation space, and exact or approximate algorithms (see, for instance [27]) have been developed only for very low dimensional scenarios.

		ES	EP	CS	CP	Dynamic
Optimal	D*	+	+	-	-	-
	MDP	-	+	-	+	-
	POMDP	+	+	+	+	-
Sampling Based	AnytimeRRTs	-	+-	-	-	-
	ParticleRRTs	+	-	-	+	-
	PMP	-	+-	-	-	+

TABLE II
COMPARISON BETWEEN PLANNING METHODS.

Sampling-based methods operate in a continuous state space, which is a more natural approach in real world navigation. Also, they seem more adapted to the dynamic environment and to handle time-constraints, as they are expressly dedicated to high dimensional spaces. However while *AnytimeRRTs* and *PMP* both lack an explicit representation of uncertainty, *Particle-RRT* is not easily extensible to dynamic environments. The complexity of sampling is in fact exponential in the number of dimensions and in the dynamic case, the dimension of the space to sample would be proportional the number of possible states of the obstacles.

We need then to develop a new algorithm which is able to represent and update uncertainty simultaneously, in real-time, with respect to the changes of the environment. Among the sampling-based path planning algorithm presented, the RRT algorithm presents some important advantages:

- it easily handles non-holonomic constraints;
- it is incremental, which seems a natural requirement when exploring an unknown environment.

We propose to use this algorithm as basis for the exploration of the configuration-time space and introduce two basic novelties:

- the probabilistic representation of the configuration-time space;
- the real-time updating of such a representation and the associated decision process.

We propose to represent the uncertainty in the static environment by mean of an occupancy grid; the uncertainty in the dynamic environment is instead given by the probabilistic prediction of the moving obstacles, represented by a mixture of Gaussians at each timestep.

III. PROBABILISTIC RISK OF COLLISION ASSESSMENT

The contribution of this paper can be summarized in the three main points:

1) **Probabilistic risk of collision**
2) **Risk guided search**
3) **Real-time decision update**

This Section presents how the probabilistic risk of collision is evaluated on the basis of the perception and prediction model used (point 1). The following paragraph (§III-A) describes how prediction is performed on the basis of the work developed in [10] to use Gaussian Processes for representing typical patterns. Paragraph III-B describes how the probability of new obstacles entering the workspace is taken into account. Paragraph III-C details the valuation of the risk of collision associated to each configuration of the robot.
Points 2 and 3 will be detailed in Section IV.

A. Probabilistic Prediction using Gaussian Processes

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 [3]:

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

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

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

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. $E[.]$ stands for expected value. It is assumed that the observation of paths belonging to an exemplar path are generated by a Gaussian process. The movements in the x and y axes are assumed to be independent, so each prototype is represented with 2 Gaussian processes, one for each axis respectively. The mean of these Gaussian processes is the mean function representing the path.
A single observation of a path is represented as two vectors (x, y) 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 sequences is then:

$$L_x = \prod_{n=1}^{N} G(x_n|\mu_x, \Sigma_x) \quad (4)$$

$$L_y = \prod_{n=1}^{N} G(y_n|\mu_y, \Sigma_y) \quad (5)$$

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. The typical velocity is learned and noted in the trajectory prototype. A single typical motion path is a D dimensional Gaussian distribution. To represent several typical motion paths, a mixture model is used. Considering K components, the likelihood based on the N training data sequences is then:

$$P(x|z,\mu_x,\theta) = \prod_{n=1}^{N}\prod_{k=1}^{K} G(x_n|\mu_{x,k},C(\theta_k))^{Z_{nk}} \quad (6)$$

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

In this equation, z is the vector of component weights, μ the mean function of the Gaussian process, and $C(\theta_k)$ the covariance matrix of the Gaussian process parameterized by θ, the Gaussian process hyperparameters.

1) Prediction: The proposed model enables one to pose questions in a probabilistic manner such as the distribution of predicted trajectories. 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 $D - M$:

$$P'(x_1,x_2) \sim G\left(\begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}, \begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}\right) \quad (8)$$

The probability of a partial path observation x_1 is evaluated by integrating over the $D - M$ dimensions of the Gaussian distribution to yield the marginal Gaussian distribution:

$$P'(x_1) \sim G(\mu_1,\Sigma_{11}) \quad (9)$$

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

$$P'_k(x_2|x_1) \sim G\left(\mu_2 + \Sigma_{21}\Sigma_{11}(x_1-\mu_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{21}^{T}\right) \quad (10)$$

The prediction obtained considering all the clusters is then a Gaussian mixture: the weight w_k of each cluster k is given by the likelihood of the observation sequence x_1 given the considered Gaussian process normalized over the set of clusters:

$$P(x_1|G_k) = G(x_1,\mu_{k,1},\Sigma_{k,11}) \quad (11)$$

$$w_k = \frac{P(x_1|G_k)}{\sum_{k'\neq k} P(x_1|G'_k)} \quad (12)$$

where $G(x_1,\mu_{k,1},\Sigma_{k,11})$ is the Gaussian Process k, $G(\mu_k,\Sigma_k)$ evaluated for sequence x_1 (see also equation 9). In order to reduce the number of considered clusters, an appropriate threshold is defined according to the Mahalanobis distance between the observation sequence and each GP.

B. New obstacles entering the scene

As the obstacles are supposed to move along typical patterns, they can enter the field of view only from points along those patterns. Also, their motion model is defined by the motion pattern itself. 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:

$$P[N_k(t_2) - N_k(t_1) \geq 1] = 1 - e^{-\lambda_k \tau} \quad (13)$$

where $N_k(t_2) - N_k(t_1)$ is the number of obstacles entering from typical pattern k in time interval $(t_1, t_2]$ where $t_2 - t_1 = \tau$. The rate parameter λ, is the expected number of arrivals per unit time. A different frequency coefficient λ_k is chosen for each typical pattern k. These coefficients can effectively be defined on the basis of the observations gathered in the learning phase, and can change in time: for example it is possible to consider different coefficients for different hours during the day or different days in a week etc... The obstacles are then supposed to move along one pattern or another with probability proportional to the learned λ_k.

C. Probabilistic Risk of Collision

When searching for a safe path, the algorithm must determine how much is the risk of collision of taking an action $u \in U$ when in configuration $q(t_1)$. This risk can be written as $P(coll(q(t_1),u) = 1)$, the probability of collision and will be referred as P_c in the rest of the paper. The risk is computed on the basis of the probability of occupation of the surface swept by the robot moving from $q(t_1)$ under control u in the interval to $[t_1, t_2[$:

$$q(t_2) = f(q(t_1),u,\tau) \quad (14)$$

$$A = \iint_{t_1}^{t_2} q(t)dt \quad (15)$$

where $f(.)$ is the motion model of the robot and $\tau = t_2 - t_1$ is the timestep. The risk of collision must take into account both the static and the moving obstacles. We make the hypothesis that moving obstacles and static obstacles cannot overlap, and consequently that collision with a static obstacle and collision with each one of the moving obstacles are mutually exclusive events, which yields:

$$P_c = P_{cs} + (1 - P_{cs}) \cdot P_{cd} \quad (16)$$

$$P_{cd} = 1 - \prod_{m=1}^{M} 1 - P_{cd}(o_m) \quad (17)$$

where P_{cs} is the probability of collision due to the static obstacles, $P_{cd}(o_m)$ is the probability of collision due to the dynamic obstacle o_m and P_{cd} is the probability of collision due to all the dynamic obstacles.

The static obstacles are represented in the occupancy grid which is assumed to be stationary. Given $\mathcal{M}(t_0)$ with $t_0 \leq t_1$ the most recent estimation of the static map, the risk of collision with a static obstacle is given by the max probability over the subset $S \subset M(t_0)$ of cells which is the minimal approximation of surface A:

$$S = \{(\mathbf{x},\mathbf{y}) \in \mathcal{M}(t_0), (\mathbf{x},\mathbf{y}) \cap A \neq \emptyset\} \quad (18)$$

$$P_{cs} = \max_{S}(P(Occ(\mathbf{x},\mathbf{y}) = 1)) \quad (19)$$

The risk of collision with a moving obstacle o_m is approximated by the probability that the area swept by the robot intercepts the one swept by the obstacle in the considered interval:

$$P_{cd}(m) = P(o_m(t) \cap A_e \neq \emptyset, \forall t \in [t_1, t_2[) \quad (20)$$

The prediction $o_m(t)$ is given by a weighted sum (mixture) of Gaussian Processes, as explained in §III-A1. To do the computation we consider the area A_e which is A enlarged by the estimated radius of the obstacle. First, each Gaussian component k is considered separately, then all the Gaussian components are summed:

$$P_{cd}(m,k) = \int_A G(o_m(t), \mu_k, \Sigma_k) \quad (21)$$

$$P_{cd}(m) = \sum_{k=1}^{K} w_{mk} \cdot P_{cd}(m,k) \quad (22)$$

where $P_{cd}(m,k)$ is the probability of collision with the obstacle m moving along pattern k; $G(o_m(t), \mu_k, \Sigma_k)$ is the Gaussian Process representing pattern k, given the observation history of object o_m (see also eq 9). The probability is marginalized over the set of possible patterns to yield $P_{cd}(m)$, where w_{mk} is the weight of the k component for object m (see equation 12).

The probability of collision with new entering obstacles is computed in the same way and then multiplied by the probability that the obstacle enters the scene (see equation 13) before being used in equation 17.

In order to choose an appropriate path, the Risk-RRT algorithm makes pass from the risk of collision of an action to the risk of partial paths:

$$\pi(q_N) = \{q_n\}_{n=0\ldots N} \,|\, q_{n+1} = f(q_n, u_n, \tau) \quad (23)$$

$$P_{c,\pi}(q_N) = 1 - \prod_{n=1}^{N} (1 - P_c(q_n)) \quad (24)$$

$$L_\pi(q_N) = 1 - P_{c,\pi}(q_N) = \prod_{n=1}^{N} (1 - P_c(q_n)) \quad (25)$$

where q_0 is the position of the robot at the current time t_0, and $\pi(q_N)$ represents the path from q_0 to q_N. $L_\pi(q_N)$ is the "likelihood" or "probability of success" of the path, and is given by the probability of *not* encountering a collision along the path.

IV. Risk Rapidly-exploring Random Trees

In this section, we present our partial motion planning algorithm *Risk-RRT* and the collision risk assessment taken into consideration by the algorithms.

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

1) An estimation of the state of the robot:
$$[x, y, \theta, v, \omega]^T$$
where (x, y, θ) is the position and orientation of the robot in the plane and (v, ω) are its linear and angular velocity respectively.
2) A set of Gaussian Processes which represent the typical patterns of the obstacles:
$$\mathcal{G} = \{G_k\}_{k=1\ldots K}$$
with K the total number of known typical patterns.
3) A goal position:
$$g = [x, y]^T$$
4) An occupancy grid which represents the structure of the static environment around the robot according to the previous observations:
$$\mathcal{M}(t) = \{p_{occ}(x, y)\}_{x \in X, y \in Y}$$
where $pocc$ is the probability of occupation, X and Y are finite sets representing the discrete coordinates of the cells of the grid.
5) A list of moving objects their estimated position, velocity and previous observations:
$$O = \{o_m\}_{m=1\ldots M} = \{[x, y, vx, vy]_m^T, H_m\}_{m=1\ldots M}$$
where H_m represents the history of observation related to obstacle o_m.

The occupancy grid and the state of the robot are estimated by a Simultaneous Localization And Mapping algorithm based on scan matching; the position, velocity and track of the obstacles is estimated thanks to a Multiple Target Tracking algorithm [28]. The typical patterns are supposed to have been learned by an off-board platform before navigation and to be known by the robot.

A. The Risk-RRT algorithm

The motion planning algorithm proposed is described in Algorithm IV-A. It combines a task dedicated to perception (of static and moving obstacles), a task for planning partial but safe trajectories and another one for navigating along planned safe trajectories. In practical, navigation and planning are done in parallel. The prediction done for forecasting the position of moving obstacles in the near future can be done by different ways, depending on the knowledge the robot has on the environment. If the robot does not have any model of the future, it can base its planning a short-term prediction based on conservative behavior [29]. If the robot has models of the future, it can anticipate the behavior of the obstacles by a long-term prediction. Section III-A describes Gaussian Processes based method for predicting moving obstacles trajectories. The selection of the best trajectories is done by computing a probability of collision between the robot following these trajectories and the moving obstacles following the predicted trajectories.

In Fig. 1, the tree of trajectories generated by the algorithm is shown at 4 instants during navigation. The initial position of the robot is at the left bottom corner, while the goal is at the right top corner. At the beginning, the most likely paths are explored in the two possible directions and the most promising one is chosen: the more promising path is drawn in red in Fig.1(a). Fig.1(b) shows the tree after some steps: the tree has been updated: the branch in the right direction has been cut

Risk-RRT

1: **procedure** RISK-RRT
2: trajectory = empty
3: Tree = empty
4: Goal = read()
5: t= clock()
6: **while** Goal not reached **do**
7: **if** trajectory is empty **then**
8: brake
9: **else**
10: move along trajectory for one step
11: **end if**
12: observe (X);
13: delete unreachable trajectories(T, X)
14: observe$(Map, movingobstacles)$
15: t= clock()
16: predict moving obstacles at time $t, ..., t + N\tau$
17: **if** environment different **then**
18: update trajectories(T,Map,moving obstacles)
19: **end if**
20: **while** clock()$< t + \tau$ **do**
21: grow trajectories with depth$<= N$ in T
22: **end while**
23: trajectory = Choose best trajectory in T
24: t = clock()
25: **end while**
26: brake
27: **end procedure**

as it became unreachable and the tree has been grown a little toward the promising direction. Fig.1(c) and (d) show the tree and the new partial path found when a bigger portion of the space is visible.

B. Risk guided search

This paragraph explains how the configuration-time space is searched and how a path is chosen. These operations correspond respectively to line 21 and line 23 in Algorithm IV-A.

The search algorithm has his basis in the well known Rapidly-exploring Random Tree algorithm [4]. The configuration-time space is searched randomly and a tree T is grown from the initial configuration all over the configuration space. The algorithm chooses a point P in the configuration space and tries to extend the current search tree toward that point. In the classical RRT algorithm, P is chosen randomly in the free configuration space. In our problem there is little or no knowledge on the structure of the environment: P is sampled from the rectangular region between the robot and the goal enlarged by some amount to take into account for possible local minima. P can be in an occupied or in an unknown zone. At the beginning, and then once on 100 times, the goal is chosen; this bias, which has been empirically set, speeds up the exploration toward the goal. The node chosen for extension is the most *promising* node: all the nodes in T are weighted taking into account the risk of collision and the estimated

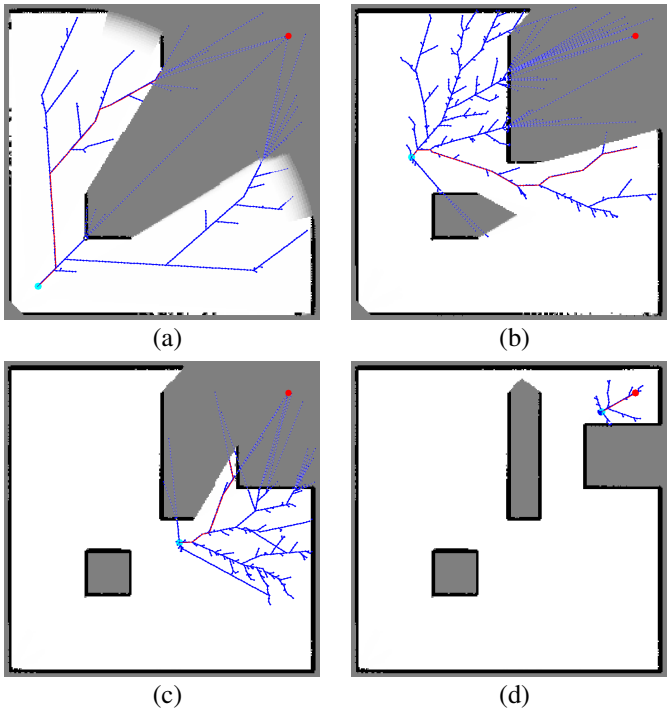


Fig. 1. Risk-RRT: Growing tree evolving in time.

length of the total path:

$$\tilde{w}(q_N) = \frac{\sqrt[N]{L_\pi(q_N)}}{dist(q_0, q_N, P)} \quad (26)$$

$$w(q_N) = \frac{\tilde{w}_(q_N)}{\sum_q \tilde{w}_q} \quad (27)$$

At numerator, the likelihood of $\pi(q_N)$ is normalized with respect to the length of the path N; at denominator, $dist(.)$ is the sum between the length of the path from the root q_0 to the node q_N (which is known) and an estimation of the length of the path to P: this estimation is an underestimate; in case of car-like robot, we have chosen a maximum steering plus straight line trajectory. The weights are normalized over the set of nodes in the tree (27). 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. The new node q_+ is obtained applying an admissible control from the chosen node q toward P. The weight of q_+ is computed. If $w(q_+) \geq w(q)$ the tree is grown again from q_+ toward P otherwise another point is sampled from the space. When the available time for planning is over, the best partial path with the highest weight for the goal is retrieved and passed to execution.

C. Real-time decisions update

This paragraph explains how the search tree is updated and how the information coming from perception is integrated in it (see respectively lines 13 and 18 of Algorithm IV-A).

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 behavior or some new obstacle could have entered the scene. The idea of Partial Motion Planning [24] 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. Execution and planning are done in parallel: while the robot moves a step along the planned partial path, the tree is updated with the information coming from the perception algorithm, the tree is grown and the new partial path is passed for execution when the timestep is over. In order to do this, the expected state of the robot at next step becomes the root of the new search tree. If there is no error in the execution of the robot, the robot expected state is a node along the best partial path already explored. In this case, the subtree of this node becomes the current search tree. If there is some error and the robot is far away of the configuration it is expected to reach, the new tree is constituted only by the new expected position and search must begin from scratch.

The updating step is done in two phases. The likelihood of each partial path (see equation 25) can also be expressed as the multiplication of the independent static and dynamic components:

$$L_\pi(q_N) = \prod_{n=0}^{N}(1 - P_{cs}(q_n)) \cdot \prod_{n=0}^{N}(1 - P_{cd}(q_n)) \qquad (28)$$

this two values are both stored in the nodes, so that they can be updated separately and only when needed. When an observation comes in from the perception algorithm, the planner checks for differences in the observed grid at first and in the tracked obstacles then. The incoming grid and the old one are subtracted; where a difference is found, the P_{cs} of the corresponding node is updated. For the moving obstacles, the algorithm checks for difference in the weights w_m, k, and the P_{cd} of the affected nodes is correspondingly updated. The tree is grown in the rest of available time.

When the static environment is known and free of moving obstacles, the algorithm degenerates to a classical RRT approach: the likelihood of the nodes is either 0, when the corresponding space is occupied, either it depends only on the distance to the goal, which is the case for the deterministic RRT algorithm. Also, the new observations do not give any new information, so there is no need to update the tree and the search can continue till the goal is reached.

V. Experiments and Results

Methods presented above have been validated in an indoor environment using described Section V-A. Section V-B illustrates the way the GP predicts the behavior of the obstacles in the near future, Sections V-C and V-D give some results based on real data perception.

A. Experimental Setup

Methods presented above have been validated in simulation using real pedestrian trajectories. A dataset of people trajectories in the entrance hall at INRIA Rhône-Alpes has been recorded using a fixed camera (see Figure 4(b)). Trajectories have been collected during a week at different moments of the day obtaining more than 3000 observation sequences. People are tracked on the images and their position is projected to the ground plane using the omography matrix of the camera. After filtering those sequences which where too short (less than 50 observations) or too long (more than 250) a total of 2048 sequences were kept, which are shown in Figure 2. Figure 3 shows the corresponding set of GP that have been learned using the dataset. Section V-B shows how prediction is performed. We made experiments of our algorithm using the perception possibilities and the kynematic model of the wheelchair robot in figure 4(a). The robot is a two wheeled robot that can turn on the spot and that is equipped with encoders on the two wheels and a laser range finder SICK, parallel to the ground, positioned in the front side at an height of $9cm$. Results have been obtained using the cycabtk simulator ([30]). This allows to analyse the performance of Risk-RRT method in many conditions with a chosen number of

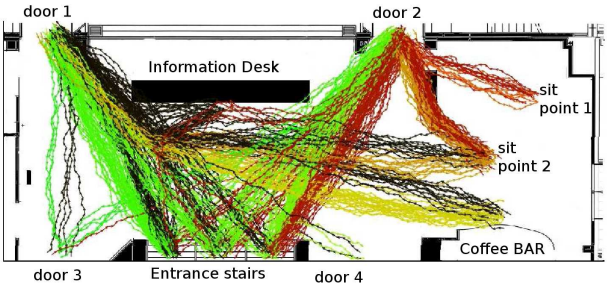


Fig. 2. Trajectory dataset in the hall at INRIA Rhône Alpes

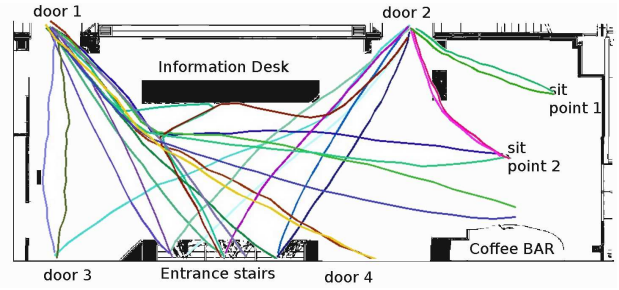


Fig. 3. Learned GP means corresponding to the trajectory dataset

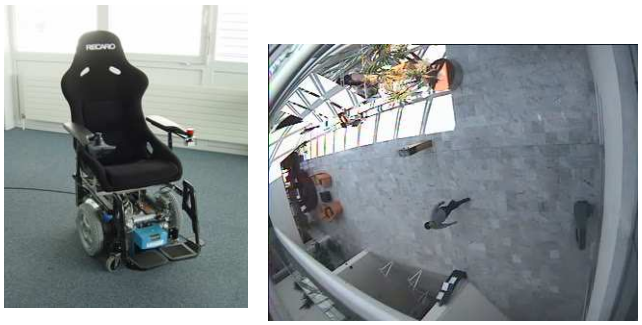


Fig. 4. (a) The wheelchair robot; (b) the hall at INRIA Rhône Alpes: view from fixed camera.

obstacles. The map of the hall has been acquired off-line with the real wheelchair robot and inserted in the cycabtk simulator ([30]). The recorded trajectories have been linearly interpolated to simulate people moving along them. The simulated pedestrians move along trajectories that are randomly selected from the recorded dataset. Section V-C shows simulation results obtained when perception, localization and execution are supposed to be perfect. This allows to alalyse the performance of our method alone. Static obstacles are represented by known polygons, while moving obstacles are represented by circular objects. The current state of the moving obstacles is known, while their future trajectory is predicted on the basis the learned GPs. The robot moves in the environment, looking for a randomly drawn goal.

Section V-C shows the results obtained in the simulator when the mapping, pedestrian detection and tracking, localization algorithm have been implemented. A fast control algorithm based on a non-linear feedback law has also been implemented to efficiently track the planned trajectory. Tests have been running into the Cycabtk simulator and on going work aims at having experiments with the real robot. A map of a priori known "virtual" obstacles have been considered. The virtual obstacles represent forbidden or untraversable areas that are impossible to detect with the sensor equipment (descending stairs, tables, windows and mirrors). The virtual obstacles are summed to the perceived map in the planning phase, while they are ignored by the mapping and localization processes. The goals are designed by the user at anytime. Moving obstacles are simulated pedestrians which are detected and tracked using the laser observations. Examples and results are shown in Section V-D.

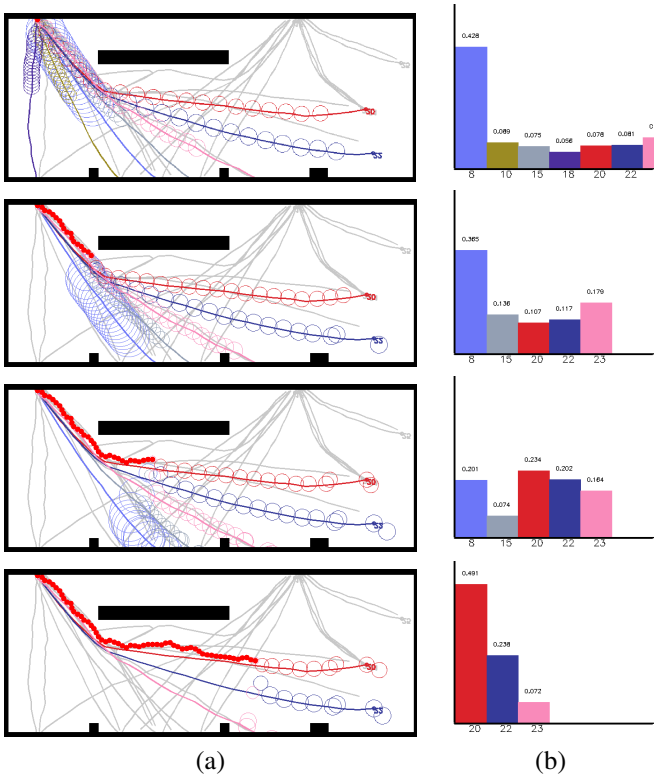


Fig. 5. Trajectory prediction based on Gaussian Processes representation: Fig.s(a) show the considered observation history (red dots) and the prediction obtained for 15 timesteps ahead: circles are centered on the mean of each Gaussian component and have radius equal to 1 time the standard deviation. Fig.s(b) show the likelihood corresponding to each GP in the mixture. Only likelihoods bigger than 0.05 are shown.

B. Prediction of moving obstacles

Figure 5 shows the Gaussian mixture prediction at 4 different timesteps. Column (a) shows the environment, the observation points (red dots) and the prediction obtained. The gray lines show the means of all the typical patterns of the environment; the colored lines are the patterns that are retained for prediction after the gating. The ellipses represent the Gaussian mixture predicted for the next $10s$ with a discretization of $0.5s$. The center of one ellipse is on the mean of the Gaussian component of the prediction and its radius is equivalent to one time its covariance. Column (b) shows the estimated likelihood for the GPs retained for prediction. In the first timestep all the trajectories originating at the door where the pedestrian is observed are likely. The prediction gets more precise as the history of observations gets longer: after some more observations, only the patterns going toward the right are retained and in the following timesteps the red path becomes prominent with respect to the others.

C. Risk-RRT Results on INRIA hall

We simulated the robot navigating among circular obstacles with trajectories that are chosen randomly from the trajectory dataset. The static environment is supposed to be known and the perception of the robot is simulated. The timestep chosen is of $0.5s$. Figure 6 shows some snapshots from the obtained results. The robot is the green rectangle and perceives the circular obstacle (red full point). The goal of the robot is the black cross on the right. Colored lines represent the portion of the GPs means retained for prediction at each time from the nearest point to the last observation to the end. Colored ellipses represent the prediction, as explained in Figure 5. The tree explored by the robot is drawn by the blue lines. Lighter blue means lower likelihood. The red line is the path chosen each time. Fig. 6(a) shows the planning at the first timestep, when no obstacle has been detected yet: the path leads straight toward the goal. In Fig. 6(b) an obstacle has been detected for the first time. Many patterns are possible: the probabilities of collision on the search tree are modified and the most likely path now gets the robot further from the obstacle. In Fig. 6(c) the pedestrian is believed to go toward the bottom part of the environment: the robot plans to drive directly toward the goal. Few timesteps after (Fig. 6(d)) the prediction is modifed: the pedestrian will go toward the right. The robot finds itself on the trajectory of the pedestrian: the trajectory searched are not safe. After some timesteps, a new solution is found: in Fig. 6(e) the robot plan is to get away from the path of the pedestrian before going toward the goal. In Fig. 6(f) the robot approaches the goal from upwards to avoid crossing the trajectory of the obstacle.

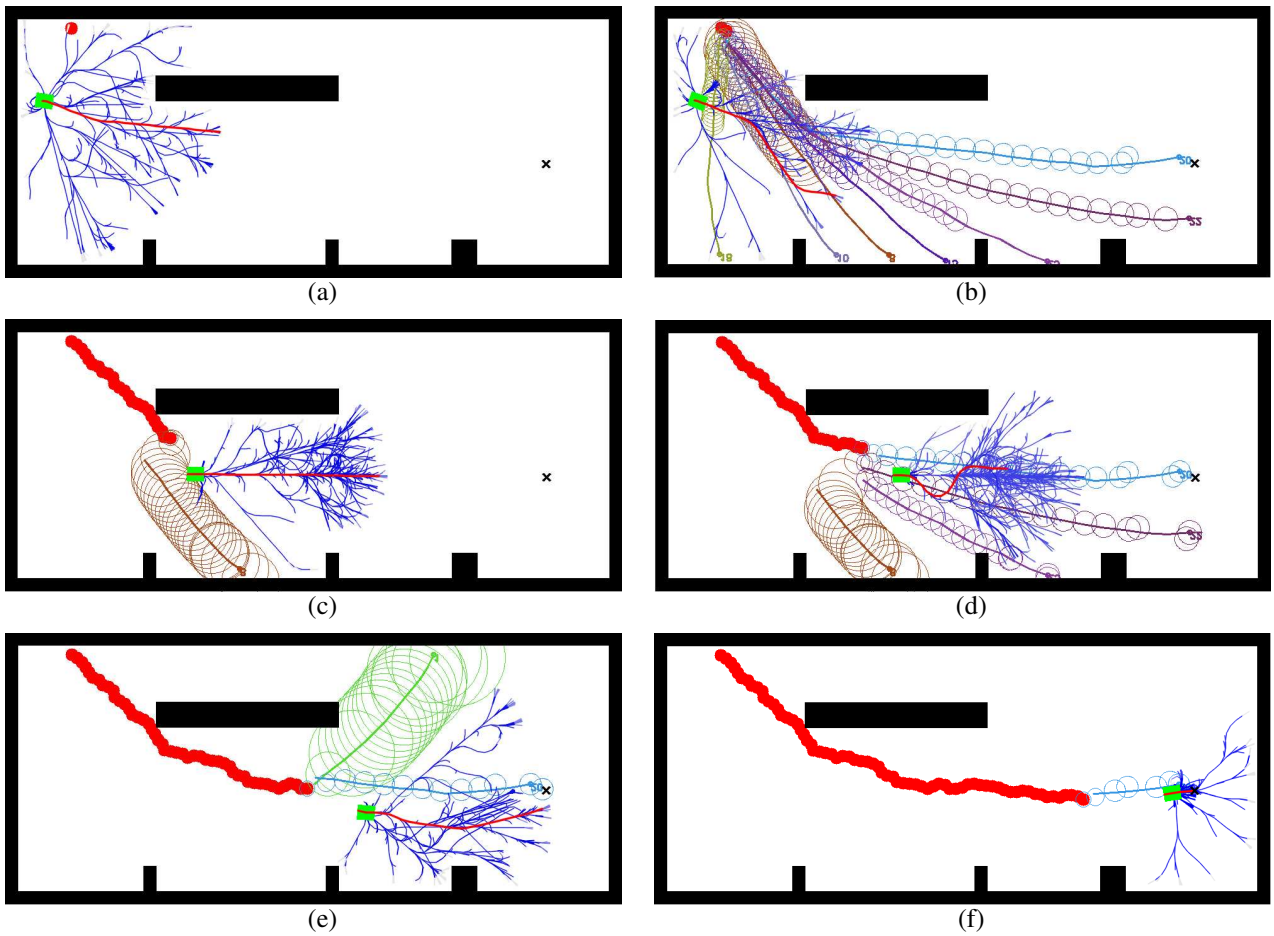


Fig. 6. The robot moves in a simulated environment with a moving obstacle. The prediction of the obstacle is given by a Gaussian mixture based on the pre-learned Gaussian processes (green). The exploration tree maintains an estimation of the likelihood of the path that is adapted to the incoming observation.

Figure 7 shows snapshots of this environment where the robot (green rectangle) aims to reach a goal (black cross) while avoiding pedestrians crossing the hall (colored points). For clarity, the predicted covariance and the search tree are not shown; only the mean of the Gaussian elements with probability bigger than 0.05 are shown: the prediction corresponding to a pedestrian is of its same color; the color intensity is proportional to the weight of the element in the Gaussian mixture. In this examples, there are 6 pedestrians in the environment. In each line an avoidance maneuver is shown. In Fig.7(a) the robot plans a trajectory to go toward the goal and avoid P1. In Fig.7(b) the prediction of P2 and P3 is changed, which oblige the robot to an avoiding maneuver. In Fig.7(c) the robot passes the crossing point safely, after that the obstacles have passed, and can go straight to the goal. Similarly, in Fig.7(d) the robot plan a trajectory which avoids P1; after the change in prediction of P2, the robot changes its plan and lets the obstacle pass (Fig.7(e) and (f)) before directing toward the goal.

Table III shows results obtained making the robot reach 100 goals in the INRIA hall environment and simulating various number of pedestrians. Due to the randomized nature of the planning algorithm, experiments have been repeated several times, and the average results are shown. Pedestrians move independently of the robot and of each other, they follow passively a predefined trajectory. A collision is detected when the distance between the rectangular robot and the position of the pedestrian is less than $30cm$. All collisions were detected when the robot is stopped: the robot does not collide with static obstacles and the collisions with moving obstacles happen when the robot is already stopped (third column). The number of *passive* collisions augments with the number of obstacles in the environment. This is mostly due to the dimensions and the physical capabilities of the robot, which can find itself trapped by multiple moving obstacles or between a moving and a static obstacle with the impossibility to quickly move backward or accelerate or steer as much as necessary to avoid collision. In this case the safest maneuver is to stop as soon as possible. It is possible to notice that the obstacles simply follow their trajectory ignoring the presence of other obstacles and of the robot. In a real environment one could argue that the detected collisions would be avoided by the obstacles and that in general a collision means that the robot "bothered" the obstacle along its preferred trajectory. Also, a collision can be caused by the position of a goal near a door and the sudden appearance of a pedestrian. Tests done giving to the robot littler dimension or augmenting the maximum steering angle or just allowing a bigger ending distance from the goal show

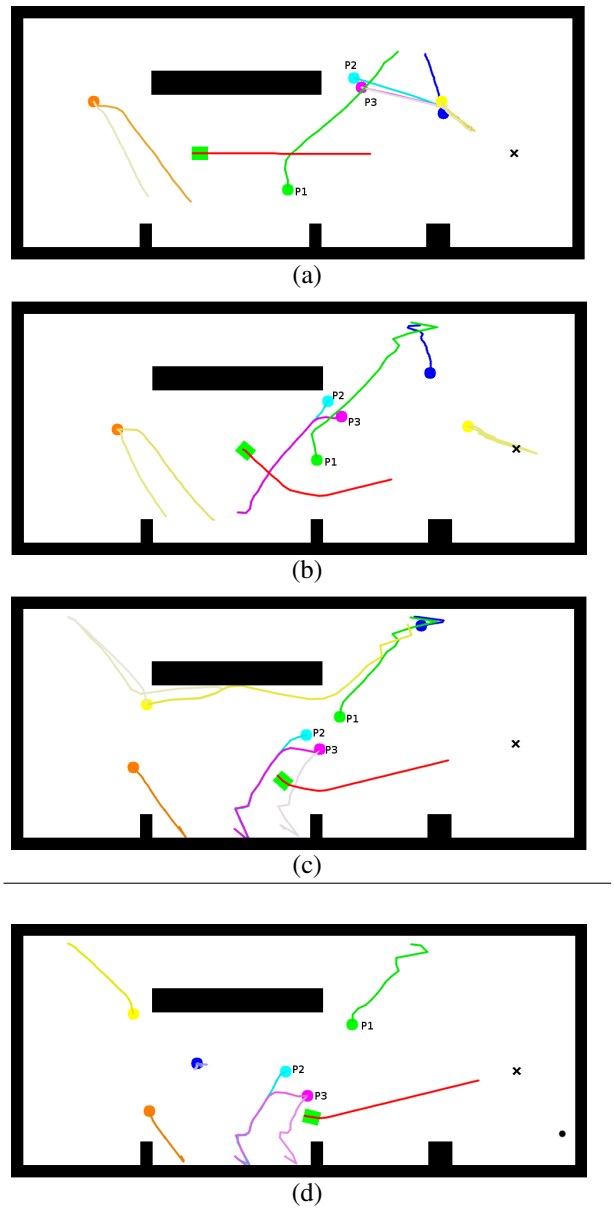


Fig. 7. Partial trajectories avoiding moving obstacles in INRIA Hall.

# pedestrians	# collisions	# collisions with robot vel.$\neq 0$	total time (s)	% time
0	0	0	663.5	1
4	1.5	0	754	1.14
6	2.3	0	951.5	1.43
8	3.8	0	1094.8	1.65
10	6.3	0	1187.6	1.79
12	7.4	0	1240.7	1.87

TABLE III
RESULTS FOR 100 GOALS REACHED AND DIFFERENT NUMBER OF PEDESTRIANS. AVERAGE RESULTS WITH GP PREDICTION, OBTAINED REPEATING THE EXPERIMENT 10 TIMES.

a sensible reduction in the number of collisions. The fourth column of the table indicates the mean time needed for the robot to accomplish the navigation task; last column indicates the percentage of this time relative to the case where there are no moving obstacles in the environment.

D. Navigation Results

To test the navigation strategy, the localization and mapping algorithm, the pedestrian detection and a control law for the wheelchair have been implemented. The laser sensor, the odometry and the execution error are simulated in Cycabtk [30]. The perception process, the planning process and the execution process are developed respectively as independent modules, which run in parallel and at different frequencies. Information is passed from one module to the others using a shared memory system (Hugr, see [30]). The perception module concerns the localization and mapping algorithm and the pedestrian detection and tracking algorithm: the algorithms described and developed by Vu in [28] have been used. The perception module runs at about $10Hz$. The planning takes in input the static map, the tracked obstacles, the estimated position of the robot and the goal defined by the user. The localization information is used to update the search tree and reset it if the robot is far from what expected. After performing prediction on the basis of the known GPs, a trajectory is elaborated in the remaining time and passed in output. We imposed a frequency of $2Hz$ to the planning module. The execution process works at high frequency ($\sim 50Hz$). The localization and the odometry data are used to estimate the current position. The last trajectory planned is interpolated to extract the desired pose of the robot at each instant. The error is used by a non-linear feedback law to choose the best control. Fig. 8 shows a screenshot of the cycabtk simulator (on the left) and the visualization of the map and the planned trajectory: the robot is in green; the gray represents unknown areas, the black and white are respectively free and occupied area. Tracked pedestrians are represented as red circles, along with their velocity vector. Fig. 9 shows screenshot during navigation in a

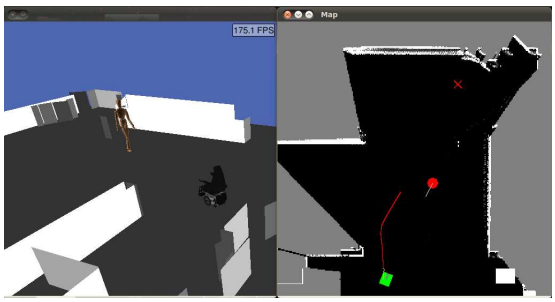
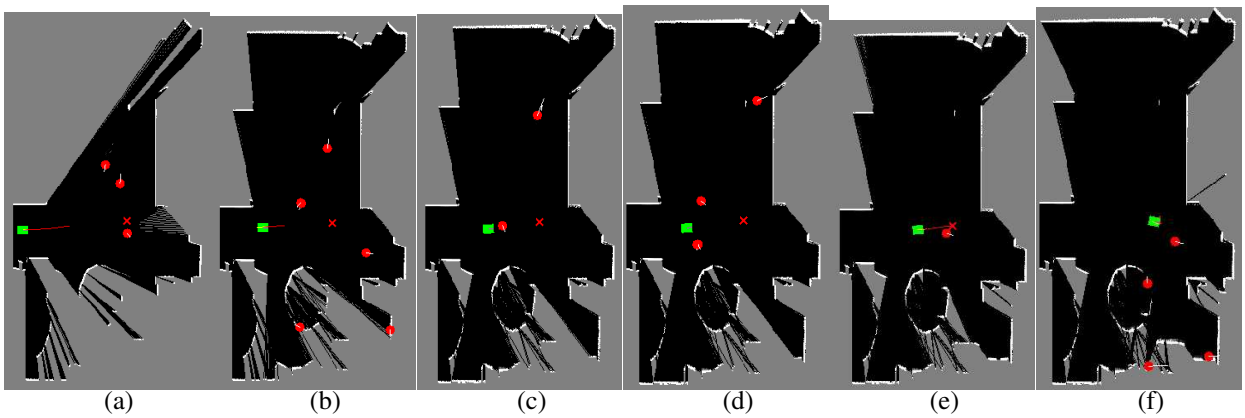


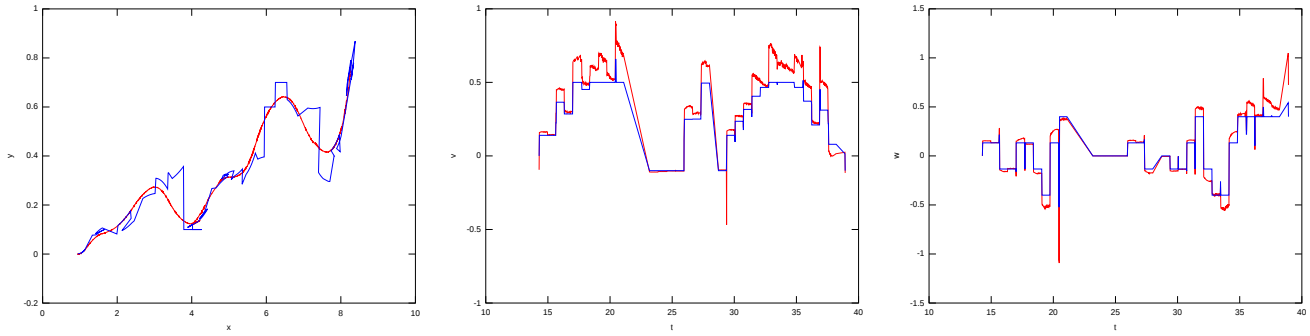
Fig. 8. Screenshot from simulation results: on the left the cycabtk simulator, with the entrance hall environment, the wheelchair and a pedestrian; on the right, the map and planned trajectory.

scenarion with 3 moving obstacles. Fig. 10 shows the reference trajectory and controls versus the obtained trajectory. We can see that the reference trajectory is not continuous: this is due to the reinitialization of the search tree caused by localization and execution error during navigation. In this example, the robot stops to let an obstacle pass (see $t = 25$) before continuing toward the goal. The results obtained are promising: even with localization and execution error, and with lower computational



(a) (b) (c) (d) (e) (f)

Fig. 9. Navigation example: the robot stops to let an obstacle pass (c-d) and avoids a second one (d-e-f).



(a) (b) (c)

Fig. 10. Reference (blue) versus real (red) values: (a) robot trajectory; (b) linear velocity; (c) angular velocity.

power available, the planner gives in output a safe trajectory. With respect to the results presented in the previous section we observed that the robot tends to brake and stop more often. In our opinion this is due to the localization and execution error on one side and on the low computational power available: all the processes (perception,prediction, planning, control, and sensor simulation) run on the same CPU, which shortens the time available for planning: the resulting search tree is often quite poor. Also we observed collisions due to miss detections of the pedestrians approaching from outside the field of view of the laser.

VI. Conclusions

In this article, we proposed a way to integrate motion prediction uncertainty in a partial motion planning. Results highlight the possibility and importance to take into account the knowledge about the behavior of moving obstacles such as pedestrians. Predicting the obstacles position on the basis of the typical patterns enables the robot to rely on a good knowledge of the environment configuration in the near future. The uncertainty of the prediction of the position of the objects has a finite dispersion and the robot has a more precise idea of where new obstacles may come from. From the point of view of navigation, this means that the robot can plan longer paths, the necessity of re-planning is reduced and better global performance are observed (shorter global path, shorter time to reach the goal). However, it is worth being noticed that the algorithm does not depend on the way motion prediction is done and if no typical pattern is observed or known by the robot, a prediction based on Kalman filter with a conservative hypothesis is used. Complete work using motion prediction based on Kalman filters and Hidden Markov Models is described in [31].

VII. Perspectives

Our algorithm has been tested on many scenarios in simulated environments reproducing observed typical behaviors in real environments. The perception used for learning typical patterns was done on an off-board dedicated architecture and the obtained information is strictly correlated with the considered environment. Also, the information gathered discards correlation between the movements of multiple obstacles. On going work deals with real world experiments with the wheelchair. In future work we would consider the case where the behavior prediction depends not only on typical patterns, but also on the configuration of other obstacles and the robot itself. We plan to take into consideration interaction between people and to give the robot this information to perform better prediction and compliant motion.

Acknowledgment

The authors would like to thank the Marie Curie Est-program and the VISITOR project for the scholarship financing Chiara Fulgenzi's PhD and Christopher Tay for his

software and work concerning the Gaussian Processes and typical patterns.

REFERENCES

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

[2] D. A. Vasquez Govea, "Incremental learning for motion prediction of pedestrians and vehicles," Ph.D. dissertation, Institut National Polytechnique de Grenoble, Grenoble (Fr), February 2007.

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

[4] S. LaValle and J. Kuffner, J.J., "Randomized kinodynamic planning," *International Conference on Robotics and Automation*, vol. 1, pp. 473–479 vol.1, 1999.

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

[6] D. Makris and T. Ellis, "Finding paths in video sequences abstract," in *British Machine Vision Conference*, 2001, pp. 263–272.

[7] I. Junejo, O. Javed, and M. Shah, "Multi feature path modeling for video surveillance," *Proceedings of the 17th International Conference on Pattern Recognition.*, vol. 2, pp. 716–719 Vol.2, Aug. 2004.

[8] D. Vasquez and T. Fraichard, "Motion prediction for moving objects: a statistical approach," in *Proceedings of the IEEE International Conference on Robotics and Automation*, New Orleans, LA (US), April 2004, pp. 3931–3936. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2004/VF04

[9] M. Bennewitz, W. Burgard, and S. Thrun, "Learning motion patterns of persons for mobile service robots," *Proceedings of the International Conference on Robotics and Automation*, vol. 4, pp. 3601–3606 vol.4, 2002.

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

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

[12] 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.

[13] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pp. 802–807 vol.2, May 1993.

[14] V. Delsart and T. Fraichard, "Navigating dynamic environments using trajectory deformation," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, Sept. 2008, pp. 226–233.

[15] J. Canny, "Constructing roadmaps of semi-algebraic sets I," *Artificial Intelligence Journal*, vol. 37, pp. 203–222, 1988.

[16] J. T. Schwartz and M. Sharir, "On the Piano Movers' Problem: II. General techniques for computing topological properties of algebraic manifolds," *Advances in Applied Mathematics*, vol. 12, pp. 298–351, 1983.

[17] L. E. Kavraki and J.-c. Latombe, "Probabilistic roadmaps for robot path planning," 1995.

[18] R. A. Brooks and T. Lozano-Pérez, "A subdivision algorithm in configuration space for findpath with rotation," *IEEE Transactions on Systems, Man, & Cybernetics*, vol. SMC-15, no. 2, pp. 224–233, 1985.

[19] J. Barraquand and J.-C. Latombe, "A Monte-Carlo algorithm for path planning with many degrees of freedom," in *Proceedings IEEE International Conference on Robotics & Automation*, 1990, pp. 1712–1717.

[20] P. Bessiere, J.-M. Ahuactzin, E.-G. Talbi, and E. Mazer, "The "ariadne's clew" algorithm: global planning with local methods," *Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on*, vol. 2, pp. 1373–1380 vol.2, Jul 1993.

[21] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics and Automation Magazine, IEEE*, vol. 4, no. 1, pp. 23–33, Mar 1997.

[22] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *International Journal of Robotics Research*, vol. 17, pp. 760–772, 1998.

[23] D. Ferguson and A. T. Stentz, "Anytime rrts," in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '06)*, October 2006, pp. 5369 – 5375.

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

[25] A. Stentz, "The focussed d* algorithm for real-time replanning," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995, pp. 1652–1659.

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

[27] A. Foka and P. E. Trahanias, "Real-time hierarchical pomdps for autonomous robot navigation," *Robotics and Autonomous Systems*, vol. 55, no. 7, pp. 561–571, 2007.

[28] 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.

[29] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Probabilistic motion planning among moving obstacles following typical motion patterns." in *IEEE/RSJ International Conference on Intelligent RObots and Systems*, St. Louis, Missouri États-Unis d'Amérique, 2009. [Online]. Available: http://hal.inria.fr/inria-00398059/en/

[30] Http://cycabtk.gforge.inria.fr/wiki/doku.php?id=start.

[31] C. Fulgenzi, "Autonomous navigation in dynamic uncertain environment using probabilistic models of perception and collision risk prediction." Ph.D. dissertation, Institut National Polytechnique de Grenoble - INPG, 06 2009. [Online]. Available: http://tel.archives-ouvertes.fr/tel-00398055/en/

Chiara Fulgenzi received her master degree in Electronic Engineering at UNIVPM (Università Politecnica delle Marche) in Ancona, in July 2005. From November 2005 she works at INRIA Rhône-Alpes, LIG Laboratory, e-Motion team. In June 2009 she obtained her PhD in applied Mathematics and Informatics working on autonomous navigation in dynamic uncertain environment, She has now a Post-doc position in the same team. Her current research interests mainly lie in Intelligent Vehicles, autonomous navigation and probabilistic reasoning.

Anne Spalanzani received her PhD in Cognitive and Computer Science from the University Joseph Fourier of Grenoble, France, in 1999. She is associate professor in e-motion team of INRIA and the University Pierre Mends France of Grenoble since 2003. She works on perception and autonomous navigation integrating probabilistic prediction of risk of collision and social interactions.

Christian Laugier received the PhD and the ”State Doctor” degrees in Computer Science from Grenoble University (France) in 1976 and 1987 respectively. He is a First class Research Director at INRIA (French National Insitute for Research on Computer Science and Control). He is the Scientific Leader of the e-Motion project-team common to INRIA Rhne-Alpes and to the LIG (Laboratory of Informatics of Grenoble), and he is in charge of the Scientific Relations with Asia & Oceania at the International Affairs Department of INRIA. Since January 2007 he is also Deputy Director of the LIG Laboratory. His current research interests mainly lie in the areas of Motion Autonomy, Intelligent Vehicles, Decisional Architectures and Bayesian Reasoning.