Toward Online Probabilistic Path Replanning in Dynamic Environments

Toward Online Probabilistic Path Replanning in Dynamic Environments

Roland Philippsen
roland.philippsen@gmx.net
LAAS-CNRS
Toulouse, France

Björn Jensen
bjoern.jensen@singleton-technology.com
Singleton Technology
Lausanne, Switzerland

Roland Siegwart
roland.siegwart@epfl.ch
Ecole Polytechnique Fédérale
de Lausanne, Switzerland

Abstract— **This paper presents work on sensor-based motion planning in initially unknown dynamic environments. Motion detection and modeling are combined with a smooth navigation function to perform on-line path planning in cluttered dynamic environments. The SLIP algorithm, an extension of Iterative Closest Point, combines motion detection from a mobile platform with position estimation. This information is used via probabilistic prediction to estimate a traversal risk function that unifies dynamic and static obstacles. The risk is fed to E* and leads to smooth paths that trade off collision risk versus detours.**

I. Introduction

Path planning in a-priori unknown environments cluttered with dynamic objects is a field of active research. It can be addressed by using explicit time representation to make the problem static and solve it with an existing planner. This increases the dimensionality and requires exact motion models for surrounding objects. The computational effort rises, and motion prediction raises theoretic and practical issues, especially in the presence of humans. As robotic companions are a promising application area, we need on-line path planning in environments with unforeseeable human behavior. It would be inappropriate to add explicit time-representation to the model: if the humans surrounding the robot do not know where they will be going, how can the robot attempt to incorporate such knowledge during planning?

A. Approach and Contribution

Figure 1 illustrates the problem statement of the Probabilistic Navigation Function (PNF). The following objectives led to its ongoing development:

- Avoid invalid assumptions, use sensor-based models. Co-occurrence estimation in PNF uses a few parameters that we quickly and robustly extract from real data.
- Avoid adding dimensions for $\mathcal{ST}$ (state$\times$time space) planning [1] by introducing worst-case scenarios.
- Interweave planning and execution, perform many computations in the workspace $\mathcal{W}$ to keep complexity low.

To produce smooth navigation functions we rely on E* [2], [3]. It is a grid-based weighted-region planner, requires relatively little computational overhead, and is capable of dynamic replanning in near real-time[1].

[1]`http://estar.sourceforge.net/`

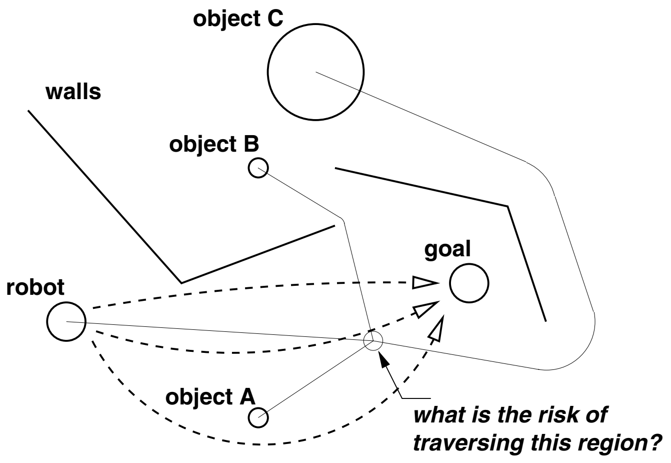


Fig. 1. The Probabilistic Navigation Function plans a path that strikes a balance between the accumulated risk and the detours necessary to avoid dangerous regions. Risk estimation relies on co-occurrence probabilities between the robot and each moving object. The shortest admissible path from each object to the region of interest is used as worst-case estimate in order to avoid the extra dimensions that would be required for $\mathcal{ST}$-space planning.

On-line motion detection is performed by SLIP [4], a scan alignment method. In order to use sensor-based motion modeling, it is of primary importance to compensate for the ego-motion of the robot. The main sensor used in the presented work is a laser scanner, thus motion detection requires robust scan alignment. The SLIP approach is based on ICP [5], [6] (a comparison of different variants can be found in [7]) and has proven its robustness on sensory data taken during Expo.02, a six-month national exhibition which took place in Switzerland during the summer of 2002.

The interface between these two building blocks of motion detection and path planning is developed in the following sections. It is an estimation of co-occurrence probabilities between the robot and surrounding dynamic obstacles (e.g. humans, other robots), given environment constraints and assumptions based on worst-case scenarios.

B. Related Work

The PNF is a navigation function, a concept of which the NF1 [8] is a classical implementation. Navigation functions are like potential fields [9], but are guaranteed to have a unique global minimum. The PNF incorporates a continuous

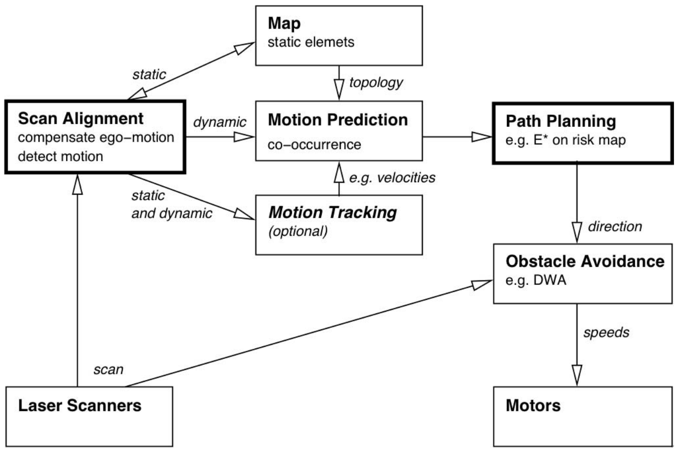


Fig. 2. Steps required for calculating the PNF: aligned laser scanner data is separated into static and dynamic objects. Static elements update the map, while dynamic elements can be optionally tracked (to obtain e.g. velocity vectors). The static and dynamic information is used to calculate co-occurrence probabilities that are fused into a navigation function. Reactive obstacle avoidance is then used to take into account the most recent scans during path execution.

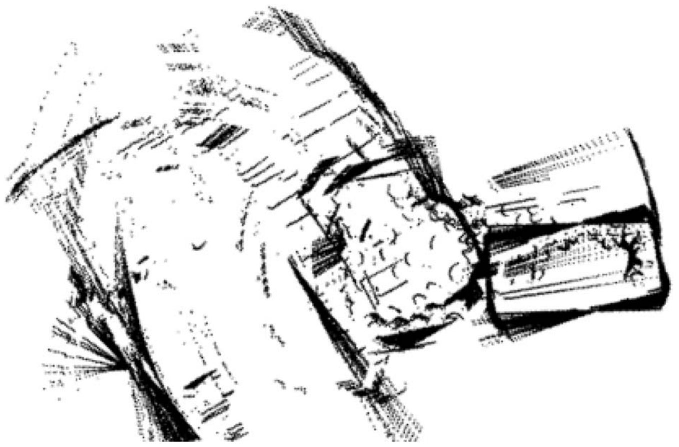
risk measure of traversing regions of $\mathcal{W}$, which is similar to edge costs in graph-based planners, but is conceptually closer to weighted regions.

The weighted region path planning problem is described in [10], [11], but instead of pre-determined regions we use a grid-based risk map. Among the published research that incorporates environment dynamics during path planning [1], [12]–[17], most seem inappropriate for an application in highly cluttered dynamic environments: they either rely on information and computational resources that are not available for such unforeseeable settings (extending $\mathcal{C}$ to a full-fledged $\mathcal{ST}$ representation [1], a velocity space [12], [15], essentially off-line movement simulations [14]), or are limited to constant velocity models [17], [18]. In [16], environment dynamics are treated using worst-case scenarios that take into account the sensor capacities, but it treats all known obstacle information as static during planning.

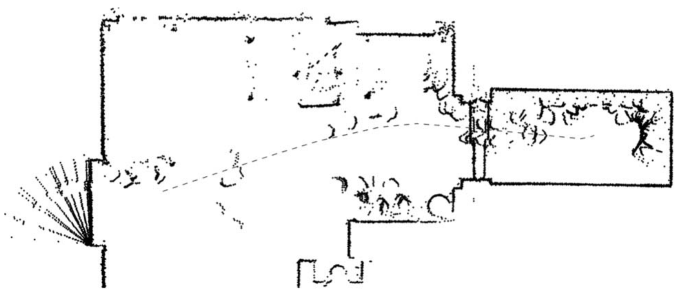
II. Algorithm Overview

The components that constitute the PNF approach are shown in figure 2. It is based on the observation that, as static objects define the environment topology, dynamic objects can be considered traversable if it is reasonable to assume that a given object will not remain at its position once the robot has moved there. The result is an approximate plan that relies on lower level obstacle avoidance to turn it into robot movements. The PNF computes a trade-off between the collision risk of traversing a region and the detour needed if it was to be completely circumnavigated. Accepting a certain risk is necessary for avoiding frequent replanning, which would cripple reactive human-robot interaction, but can be done only if lower-level collision avoidance and other safety features perform reliably. The overall steps of the algorithm are:

Input: Laser scanner data (and odometry / localisation)



(a) scan data superimposed using raw odometry



(b) the same scans after SLIP alignment

Fig. 3. Example of a scan alignment: the information obtained by raw odometry is not suitable for motion detection. After correcting the ego-motion using SLIP, regions of motion are now immediately apparent. The corrected robot path is shown as a dashed line in the lower image.

Motion detection: SLIP precisely determines the transformation between subsequent scans, then reliably detects motion that takes into account occlusion changes from a moving platform.

Determine co-occurrence risk: Translate dynamic objects into a probability of colliding with the given object at a given location, based on assumptions derived from worst-case scenarios.

Path planning: Compute a smooth navigation function using E*, taking into account a fusion of all co-occurrence risks.

Output: Direction of travel (steepest gradient) that can be fed into reactive obstacle avoidance, such as the one in [19].

III. Scan Alignment and Motion Detection

Motion can be detected as differences between successive scans, because moving objects change sensor readings. Additionally, however, differences arise from occlusion changes due to the motion of the robot. Thus, the ego-motion has to be compensated prior to comparing scans. SLIP performs scan matching based on an initial guess from odometry, then iteratively establishes links between points, and transforms

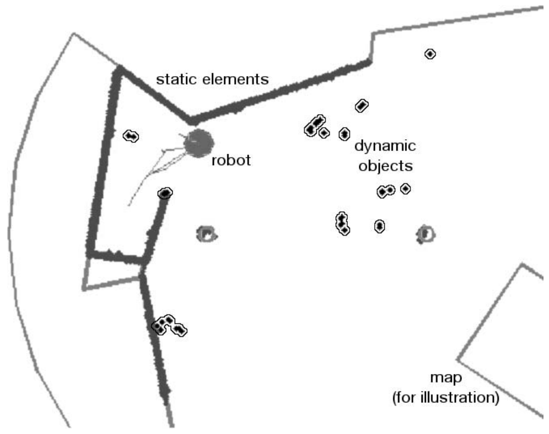


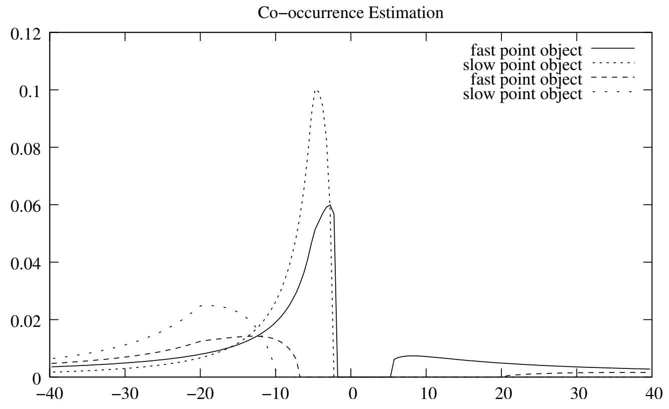
Fig. 4. Example of a motion detection from a mobile robot, using data acquired by the *Photobot* at EXPO.02. This was a difficult region for localisation, producing particularly large orientation errors when the robot rotated on the spot. Using SLIP, the scans were aligned very precisely to allow robust motion detection (dark spots surrounded by thin lines). Note that the map is shown for illustration only, SLIP does not require such a-priori information.

the scans to minimise the remaining distance between the elements. Special care has to be taken to suppress outliers, particularly from moving objects, in order to achieve a high precision. Alignment correction is based on differences between the centers of gravity of the matching point sets in both scans.

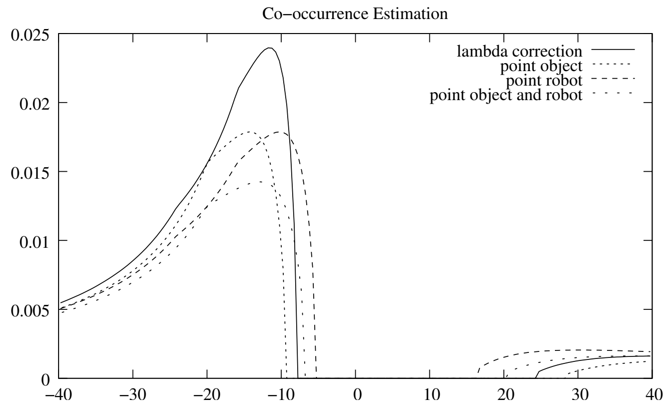
To detect motion on aligned scans, elements without correspondence within a defined distance (derived from the maximal localisation error) are considered outliers. Projective filters are used to distinguish between moving objects and occlusion changes. Non-outliers are used to create a map of the static environment. SLIP then determines which outliers were visible from the previous position. An example alignment result is given in figure 3. Moving elements are clustered by the well known friend-of-friends algorithm to model dynamic objects with associated location (the center of gravity) and size (cluster radius). An example of motion detection from a mobile robot is shown in figure 4.

IV. Planning with Estimated Risk

Conceptually, the co-occurrence models the probability that a given location will be occupied by a static or dynamic object by the time the robot has moved there. In principle, when all future trajectories are known, co-occurrence is a deterministic entity. However, the robot trajectory cannot be known at the planning stage, and the object movements are usually not available under real-world conditions. To cope with this, we first reduce the problem to point objects evolving in one dimension, then apply probabilistic worst-case reasoning to compute a co-occurrence estimate, and finally transform the $\mathcal{W}$-space information such that it represents non-point objects in D_W dimensions. Figure 5 shows the form of probability densities that we calculate.



(a) effect of relative speed



(b) after $\mathcal{W}$-space transform

Fig. 5. $P_{\mathrm{c},i}$ in the 1D case. The robot is at the origin, collisions may occur on either side of its initial position: objects can catch up if they have higher speeds. *(a)* objects are located at $x = \{-20, -5\}$ and move with equal or twice the speed of the robot. *(b)* $P_{\mathrm{c},i}$ is adapted to the robot and object shapes in the $\mathcal{W}$ transform, which computes the distance from a point of interest to the borders of the robot and the dynamic object (at $x = -20$ and moving twice as fast as the robot, object and robot radii $r_{\mathrm{r}} = r_i = 4$).

A. Co-Occurrence Estimation

Instead of attempting to take into account the infinite number of trajectories that could lead to a certain region of interest $R(x)$ at a given time, we consider the case where the robot moves with v_{r} along the shortest *admissible* path to $R(x)$, and then estimate the probability of a dynamic object being there as well. Admissible paths are those along which the robot does not collide with any *static* objects. The length of the shortest such path is denoted $\lambda_{\mathrm{r}}(x)$.

Similarly we define a dynamic objects's admissible paths and corresponding $\lambda_i(x)$, and consider a stochastic process for the movement of the object over λ_i from its current (estimated) position to $R(x)$. We assume that the object's speed and direction of travel (*along* the shortest admissible path) can change several times, but that it is bounded by v_i. This motion model allows to compute the probability that the object will be at $R(x)$ when the robot arrives. We consider them to co-occur if they are then within the same interval of size δ, which discretizes space into grid cells.

We develop the co-occurrence estimation P_{c} in the one-dimensional case. For workspaces with $D_{\mathcal{W}}$ dimensions we compute λ_{r} and λ_i for each region of interest and thus obtain $P_{\mathrm{c}} : \mathbb{R}^{D_{\mathcal{W}}} \to \mathbb{R}$ for each dynamic object i. $P_{\mathrm{c},i}$ has five terms that are switched on or off depending on the robot and object speeds and reflect co-occurrences in different areas, namely far-left, left, center, right and far-right:

$$P_{\mathrm{c},i} = P_{\mathrm{c}}(\lambda_i, \lambda_{\mathrm{r}}, v_i, v_{\mathrm{r}}, \delta) = p_{\mathrm{c,l}} + p_{\mathrm{c,ll}} + p_{\mathrm{c,m}} + p_{\mathrm{c,rr}} + p_{\mathrm{c,r}} \quad (1)$$

where v_{r} is the robot speed, v_i the object speed, λ_i the distance to the object, λ_{r} the distance to the robot, and δ the grid resolution.

Each individual p_{c} represents an estimate for the partial co-occurrence in one of the five different areas mentioned above. Equations (3)-(7) are the expressions for these terms and the conditions under which they are non-zero. The values of $v_{1,2}$, N, and η are defined as:

$$v_{1,2} = \frac{v_{\mathrm{r}}\left(\lambda_i \mp \delta\right)}{2\lambda_{\mathrm{r}}},\ N = \left\lceil \frac{\lambda_{\mathrm{r}}}{\delta} \right\rceil,\ \eta = \frac{N-1}{2Nv_i^2} \quad (2)$$

The bounds $v_{1,2}$ govern the applicability of the individual terms below. For each of the following equations, if the interval condition does not hold, the corresponding $p_{\mathrm{c}} = 0$.

if $(v_1, v_2) \in (-\infty, -v_i) \times (-v_i, 0)$
$$p_{\mathrm{c,l}} = \frac{v_1 + v_i}{v_i} + \eta\left(v_1^2 - 2v_i^2\right) \quad (3)$$

if $(v_1, v_2) \in (-v_i, 0) \times [-v_i, 0)$
$$p_{\mathrm{c,ll}} = \frac{v_2 - v_1}{v_i} + \eta\left(v_2^2 - 2v_1^2\right) \quad (4)$$

if $(v_1, v_2) \in [-v_i, 0) \times [0, v_i)$
$$p_{\mathrm{c,m}} = \frac{v_2 - v_1}{v_i} - \eta\left(v_2^2 + 2v_1^2\right) \quad (5)$$

if $(v_1, v_2) \in [0, v_i) \times [0, v_i)$
$$p_{\mathrm{c,rr}} = \frac{v_2 - v_1}{v_i} - \eta\left(v_2^2 - 2v_1^2\right) \quad (6)$$

if $(v_1, v_2) \in [0, v_i) \times [v_i, \infty)$
$$p_{\mathrm{c,r}} = \frac{v_i - v_2}{v_i} - \eta\left(v_i^2 - 2v_2^2\right) \quad (7)$$

Note that λ denotes distances after the $\mathcal{W}$-transform, such that these equations take into account non-point objects and robots by using the distance to their boundaries, as opposed to their center points. The effect of the $\mathcal{W}$-transform is illustrated in figure 5(b).

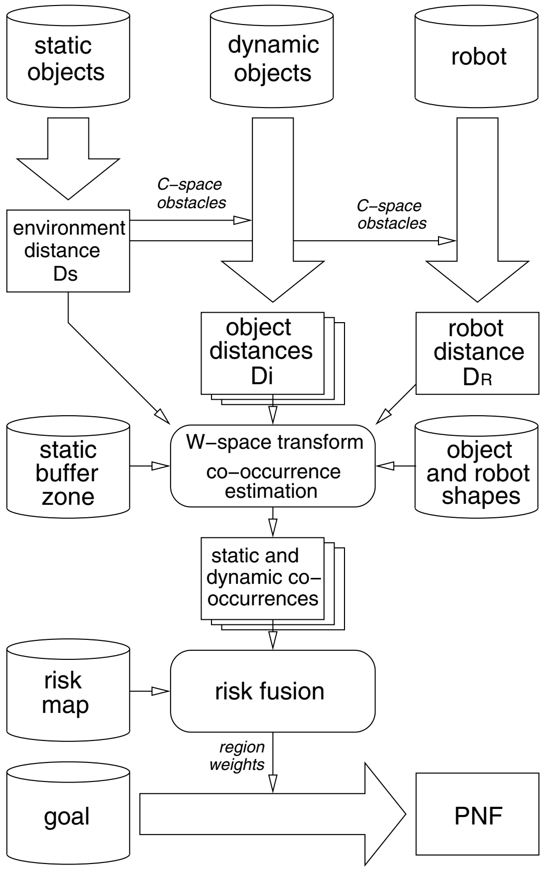


Fig. 6. Processing flow inside the PNF. Cylinders show a-priori information and inputs, rectangles represent grid layers, big arrows denote E* operation, rounded boxes are other computations. $\mathcal{C}$ obstacles are essentially binary maps for each object and the robot.

B. Risk Fusion and Planning

How can the co-occurrence information be used for path planning? The PNF is based on a combination of collision risks estimated in workspace $\mathcal{W}$, which is then fused and transformed to configuration space $\mathcal{C}$. The main contribution lies in the formulation of multiple layers of co-occurrence probabilities, and how these are fused into a risk map representing all dynamic objects as well as the environment topology. A flow diagram of the PNF algorithm is given in figure 6:

1) The distance from each grid cell to the closest static object is computed using E*. The resulting distance map is denoted D_{s}
2) D_{s} is used to compute $\mathcal{C}$-space obstacles for each dynamic object as well as for the robot. Then, similarly to the first step, E* is invoked to calculate the topologically correct distance maps D_i to the current centers of dynamic objects and D_{r} to the center of the robot.
3) The distance maps are transformed to values that can be fed into the one-dimensional co-occurrence equations: the $\mathcal{W}$-space transform computes $\lambda_i(x) = \min_{y \in A_i(x)}\left(D_i(y)\right)$, which represents the distance of a point x to the border of the dynamic object i, where

$A_i(x)$ denotes the object shape placed at location x. λ_r is computed accordingly.

4) λ_i and λ_r are fed into equation (1) to yield dynamic object co-occurrence maps, that is to say $P_{\mathrm{c},i}$ for each object at each grid cell. D_s is similarly transformed into the static co-occurrence map, after optionally applying a buffer zone which can help to smooth the robot behavior close to walls.
5) Risk fusion is performed as $P_\mathrm{r}^{\mathcal{W}} = 1 - \prod(1 - P_\mathrm{c})$ over the dynamic and static co-occurrences in $\mathcal{W}$-space, followed by $P_\mathrm{r}^{\mathcal{C}} = 1 - \prod(1 - P_\mathrm{r}^{\mathcal{W}})$ over the robot shape to expand it to $\mathcal{C}$-space. A tunable risk map (typically of sigmoid form) is used to turn $P_\mathrm{r}^{\mathcal{C}}$ into region weights.
6) Finally, E* is used again, this time to compute the navigation function taking into account the cell weights computed with the help of the risk map. The resulting values are the Probabilistic Navigation Function.

The number of dynamic objects and the size and resolution of the $\mathcal{W}$ and $\mathcal{C}$ representations all influence the computational complexity (see figure 6). Computing D_s, D_i, and D_r requires $O(IW^{D_W})$, where I is the number of objects and W is the number of cells along one axis in a D_W-dimensional $\mathcal{W}$-space. The $\mathcal{W}$-space transform, risk fusion, and final planning step all take place in $\mathcal{C}$, altogether adding $O(IC^{D_C})$, where C is the number of cells along one $\mathcal{C}$-axis in D_C dimensions. Without taking into account SLIP and queue-ordering overhead in E*, the overall computational complexity of the core planning algorithm is thus $O(IW^{D_W}C^{D_C})$.

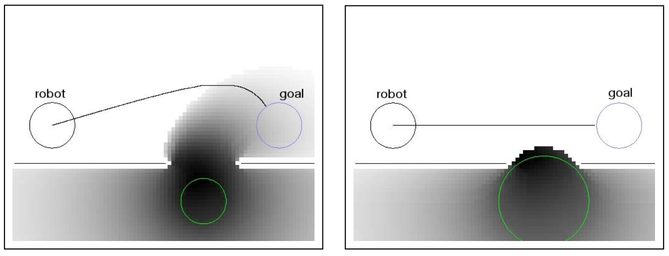
V. Results

Figures 7(a) and 7(b) illustrate how PNF takes into account the environment topology for each object individually, for example in a hallway where objects can loom from rooms. The robot is the circle on the left, with a trace of gradient descent towards the goal on the right. There is a moving object behind the opening. The path is pushed into the free space in order to maximize the distance from the door, but only if the object is actually small enough to fit through. In this example, the robot speed is $v_\mathrm{r} = 0.3$, the object moves with $v_i = 0.2$. Figures 7(c) and 7(d) show the effect of adding an object that moves with the same speed as the robot, in this case making the path switch homotopy class.

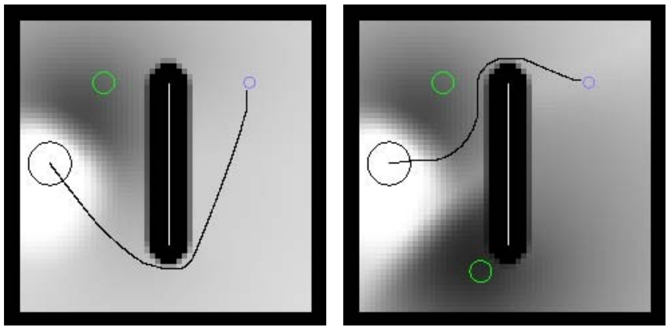
The various mappings in these figures are very smooth, which is a consequence of the interpolation in E*, and could not be achieved with any strictly graph-based method such as D* [20]. However, it has an increased computational cost (up to 40% increase per operation and up to 65% increase in the number of operations).

VI. Discussion

In general, the robot may travel on arbitrary admissible paths from one point to another. Taking this into account would lead to an iterative approach of path planning and estimating the co-occurrence probabilities. But such a method is not only going to suffer from expensive computations, it may also face non-trivial convergence issues. We avoid the need for iteration by assuming that the robot will reach each point as fast as possible, and that the objects follow shortest admissible paths (with varying but bound velocities). These simplifications make it possible to address on-line planning with dynamic objects in real-time.



(a) $P_{\mathrm{c},i}$ of a small object (b) $P_{\mathrm{c},i}$ of a big object



(c) P_r at $v_i = 2.5 v_\mathrm{r}$ (d) P_r at $v_j = v_\mathrm{r}$

Fig. 7. PNF takes into account the environment topology individually for each object. Compare *(a)* with *(b)*: When an object is too large to fit through an opening, it does not interfere with the robot trajectory. *(c)* and *(d)* illustrate how different object speeds affect the trajectory, and how the addition of a dynamic object can influence the topology of the chosen path.

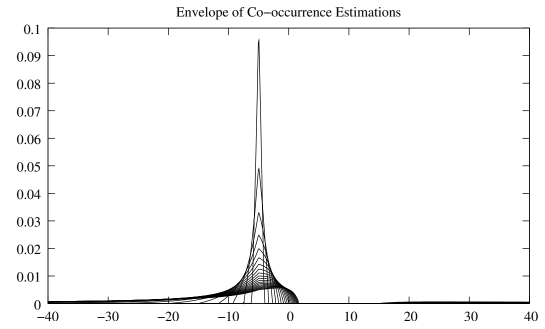
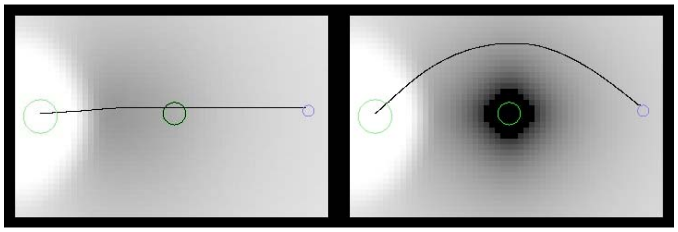


Fig. 8. This ad-hoc alternative motion model simply maximizes the co-occurrence along the v_i axis. We denote with P^* the envelope of P_c curves, and use it to illustrate the local effects of motion models in figure 9.

The motion model underlying the estimation of P_c could be based on different application-dependend assumptions, or it could be extended to take into account more knowledge about dynamic objects. The effects of changing the motion model are visible on the global and local scale:

Globally: Motion models influence the conditions under



(a) P_c (eqs. (3)-(7)) (b) $P^* = \max_{v \in [0, v_i]} P_\mathrm{c}$

Fig. 9. Using P^* (shown in figure 8) as motion model leads to more cautious robot behavior in the presence of very fast objects. In this example, $v_i = 2.5 \times v_r$ and the object between robot and goal hardly influences the trajectory because P_c is smeared, whereas P^* keeps the robot from planning through the object's current position.

which the planned path switches homotopy, as illustrated in figures 7(c) and 7(d).

Locally: During path execution, the direction and steepness of the gradient is determined by the motion model, it thus influences how strongly dynamic objects "repel" the robot.

The local effect can be seen in figure 9, which illustrates an ad-hoc motion model that addresses a potential drawback with very fast objects that are not close to the robot: if $v_i \gg v_\mathrm{r}$ and $\lambda_i > L$ and $\lambda_i > L$, where L is a "near" distance, then P_c is smeared widely across $\mathcal{W}$. The robot thus potentially plans a path through present positions of fast dynamic objects. This behavior could be countered by preserving a high-co-occurrence peak around the current object position, as shown in figure 8.

Note that we use P^* solely for the purpose of this discussion, it is not based on a maximum-likelihood formulation. Given that E* can efficiently update existing plans when new information arrives, and that v_i seldom takes on excessive values in the target applications, the motion model presented in this paper can be considered appropriate.

VII. Conclusion and Outlook

The Probabilistic Navigation Function is an approach to on-line path planning for a-priori unknown dynamic cluttered environments. It incorporates sensor-based motion models into weighted region planning, using a probabilistic risk map based on co-occurrences. Global and local robot behavior can be tuned by changing the motion model. The individual building blocks were designed with on-line constraints in mind: incremental knowledge, frequent changes to environmental information, adapting existing plans, and separating planning from execution.

The finished components of the PNF are scan alignment, probabilistic collision risk estimation, and computation of the navigation function. Verifications have been carried out via simulation. Ongoing work concerns integration and testing on a real robot and implementation of higher-dimensional $\mathcal{C}$-spaces. Motion detection and ego-motion compensation were combined in the SLIP algorithm to segment sensor data into static and dynamic objects. The dynamic information is used to predict future positions, taking into account the available knowledge for each object and the static environment topology. E* is used to plan with co-occurrence information. For execution, we rely on lower level reactive obstacle avoidance guided by gradient descent, an interplay between planning and execution that has proven to perform well.

References

[1] T. Fraichard and H. Asama, "Inevitable collision states - a step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[2] R. Philippsen, "Motion planning and obstacle avoidance for mobile robots in highly cluttered dynamic environments," Ph.D. dissertation, Ecole Polytechnique Fédérale de Lausanne, 2004.

[3] R. Philippsen and R. Siegwart, "An interpolated dynamic navigation function," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[4] B. Jensen, "Motion tracking for human-robot interaction," Ph.D. dissertation, Ecole Polytechnique Fédérale de Lausanne, 2004.

[5] P. J. Besl and N. D. Kay, "A method for registration of *-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[6] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.

[7] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proceedings of the Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, 2001.

[8] J.-C. Latombe, *Robot motion planning*. Dordrecht, Netherlands: Kluwer Academic Publishers, 1991.

[9] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 1, 1986.

[10] J. S. B. Mitchell and C. H. Papadimitriou, "The weighted region problem: Finding shortest paths through a weighted planar subdivision," *Journal of the ACM*, vol. 38, no. 1, pp. 18–73, 1991.

[11] N. C. Rowe and R. S. Alexander, "Finding optimal-path maps for path planning across weighted regions," *International Journal of Robotics Research*, vol. 19, no. 2, pp. 83–95, 2000.

[12] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using the relative velocity paradigm," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1993.

[13] T. Fraichard, "Trajectory planning in a dynamic workspace: a 'state-time space' approach," *Advanced Robotics*, vol. 13, no. 1, pp. 75–94, 1999.

[14] E. J. Bernabeu, J. Tornero, and M. Tomizuka, "Collision prediction and avoidance amidst moving objects for trajectory planning applications," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2001.

[15] F. Large, S. Sekhavat, Z. Shiller, and C. Laugier, "Towards real-time global motion planning in a dynamic environment using the NLVO concept," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.

[16] R. Alami, T. Siméon, and K. Madhava Krishna, "On the influence of sensor capacities and environment dynamics onto collision-free motion plans," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.

[17] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[18] J. van den Berg, D. Ferguson, and J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[19] R. Philippsen and R. Siegwart, "Smooth and efficient obstacle avoidance for a tour guide robot," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[20] A. Stentz, "The focussed D^* algorithm for real-time replanning," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.