Towards Real-Time Sensor-Based Path Planning in Highly Dynamic Environments

Roland Philippsen[1], Björn Jensen[2], and Roland Siegwart[3]

[1] LAAS-CNRS, Toulouse, France
roland.philippsen@gmx.net
[2] Singleton Technology, Lausanne, Switzerland
bjoern.jensen@singleton-technology.com
[3] Ecole Polytechnique Fédérale de Lausanne, Switzerland
roland.siegwart@epfl.ch

Summary. This paper presents work on sensor-based motion planning in initially unknown dynamic environments. Motion detection and probabilistic motion modeling are combined with a smooth navigation function to perform on-line path planning and replanning in cluttered dynamic environments such as public exhibitions. The SLIP algorithm, an extension of Iterative Closest Point, combines motion detection from a mobile platform with position estimation. This information is then processed using probabilistic motion prediction to yield a co-occurrence risk that unifies dynamic and static elements. The risk is translated into traversal costs for an E^* path planner. It produces smooth paths that trade off collision risk versus detours.

1 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 turn the problem into an equivalent static problem, which can then be solved with an existing static planner. However, this increases the dimensionality of the representation and requires exact motion models for surrounding objects. The dimensionality increase raises the computational effort (time and memory) to produce a plan, and motion modeling raises difficult prediction issues (computationally expensive, hard to foresee the long-term evolution of real-world environments). Motion prediction becomes even more problematic in the presence of humans, and the robot is usually required to react swiftly rather than optimally. In other words: the time required to compute the plan becomes part of the optimality criterion applied to the plan.

Human behavior is unforeseeable in most situations that include human-robot interaction. As service robots or robotic companions are a highly promising application area, we actively research on-line path planning in environments with up to several hundred humans, as shown in figure 1. Relying on

Fig. 1. EXPO.02 was a very crowded environment. The unpredictable behavior of humans and its dependence on the robot's actions make this a challenging real-world setting.

sensor-based motion modeling is crucial for correctly grounding the robot's plan. We do not try to explicitly model object and robot movements with a time-extended representation, as such complete knowledge is usually not available in the targeted applications: if the humans surrounding the robot do not know where they will be going, how can the robot be expected to incorporate such knowledge during path planning?

1.1 Approach and Contribution

Figure 2 illustrates the problem statement we propose to solve with the Probabilistic Navigation Function (PNF), with the following objectives in mind:

- Avoid basing the plan on invalid assumptions. Instead, use closely sensor-based environment and object models. We identify two main reasons why models can be inappropriate: they might not correspond to the observed movements (humans rarely move with constant velocity along straight lines), or they can be hard to ground (determining model parameters by observation). The co-occurrence estimation underlying the PNF is based on few parameters that can be quickly and robustly extracted from sensory data.
- Avoid the large number of dimensions that would be required for full-fledged $\mathcal{ST}$ (state×time space) planning [5,6]. We use worst-case scenarios to keep the dimensionality low. While this implies that no strict avoidance guarantees can be made, we have argued above that the required models would not be available for the target applications.

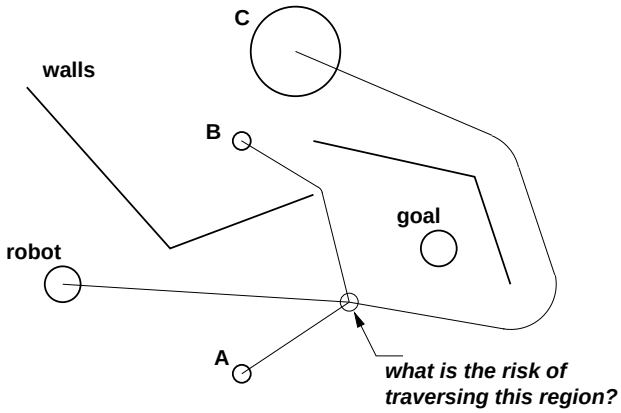


Fig. 2. The Probabilistic Navigation Function PNF was developed to solve the problem sketched in this diagram: *plan a path from robot to goal that strikes a balance between the accumulated risk and the detours necessary to avoid dangerous regions*. The risk is based on co-occurrence probabilities $P_{c,i}$ between the robot and each moving object. $P_{c,i}$ is estimated by taking into account static obstacles, approximated robot and object shapes, as well as maximum speeds. As indicated in this figure, the shortest admissible path from each object to the region of interest is used as worst-case estimate in order to avoid the very large number of dimensions that would be required for a $\mathcal{ST}$ space planning approach.

- Interweave planning and execution. This requires fast (re)planning and a flexible interface to motion control. By performing most computations in the workspace $\mathcal{W}$ or low-dimensional projected configuration spaces $\mathcal{C}$, planning complexity is reduced. The drawback of these speed-ups is the lack of geometrical completeness. Interfacing motion control is done via the steepest negative gradient of the navigation function, which is defined in (almost) all the regions the robot might reach during its movement.

This contribution presents ongoing development of the PNF approach. Prior work has resulted in the two main building blocks of motion detection and smooth navigation functions. Here we concentrate on presenting the link between the two.

To produce a smooth navigation functions (potentials with a globally unique minimum at the goal), we rely on E* [13, 15], a grid-based weighted-region planner, requiring relatively little computational overhead and being capable of dynamic replanning in near real-time[4]. This is an important property for frequently changing environment models as it limits calculations to regions that actually influence the current plan.

On-line motion detection is performed by SLIP [8], 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

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

the presented work is a laser scanner, thus motion detection requires, among other things, a robust scan alignment method. The SLIP approach is based on ICP [3,21] (a comparison of different variants can be found in [17]) 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, and where the Autonomous Systems Lab was present with eleven autonomous Robox tour-guides.

The interface between these two building blocks of motion detection and path planning is developed in the following sections. It is a probabilistic approach to estimating 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.

1.2 Related Work

The PNF is a navigation function, a concept of which the NF1 [11] 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 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 [12, 16], 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, 2, 4–7, 10], 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 state-time representation [5, 6], a velocity space [4, 10], essentially off-line movement simulations [2]), or are limited to constant velocity models [7, 20]. In [1], 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.

2 Algorithm Overview

The components that constitute the PNF approach are shown in figure 3. 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 avoided. Accepting a certain collision risk is useful

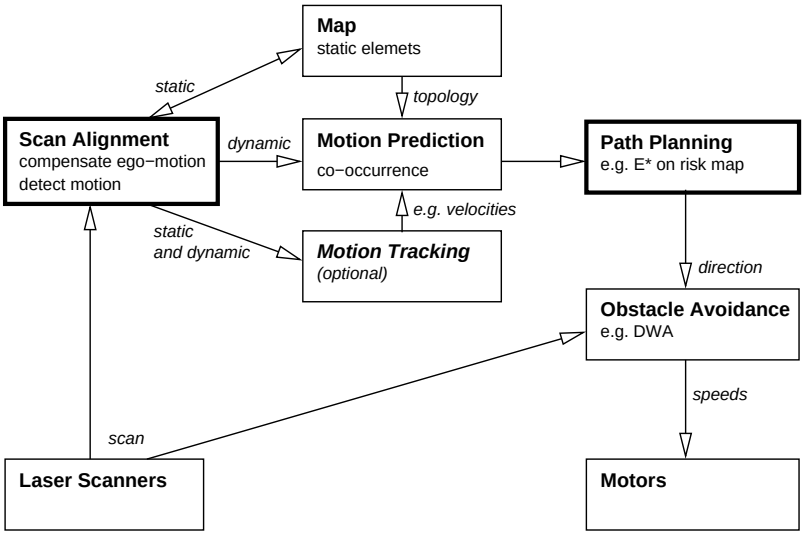
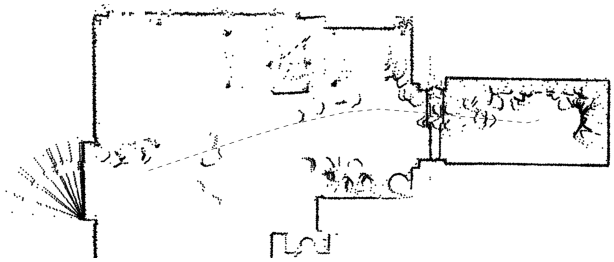


Fig. 3. Overview-diagram of the steps required for calculating the Probabilistic Navigation Function: laser scanner data is aligned with previous scans and separated into static and dynamic objects. Static elements are used to update the map. Dynamic elements can be optionally tracked in order to obtain more information such as velocity vectors. The static and dynamic information is used to calculate co-occurrence probabilities that define a risk map, which is transformed into a navigation function in the path planning step. Reactive obstacle avoidance is then used to execute the plan, which allows to take into account the most recent scan when determining the motor commands.

in the target application[5]. Otherwise, the robot would often not move at all or replan too frequently to be useful e.g. for tour guiding in mass exhibitions. The overall steps of the algorithm are:
Input: Laser scanner data (and odometry / localisation)
Motion detection: SLIP precisely determines the transformation between subsequent scans, then reliably detects motion that takes into account occlusion changes from a moving platform, as summarized in section 3.
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. This is presented in section 4.
Path planning: Compute a smooth navigation function using E*, taking into account a fusion of all co-occurrence risks. This is also presented in section 4.
Output: Direction of travel (steepest gradient) that can be fed into reactive obstacle avoidance. This separation of path planning and execution has proven to work well during Expo.02 [14, 18].

[5] Provided that collision avoidance and other safety features of the robot perform reliably

(a) scan data superimposed using raw odometry



(b) the same scans after SLIP alignment

Fig. 4. 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.

3 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 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.

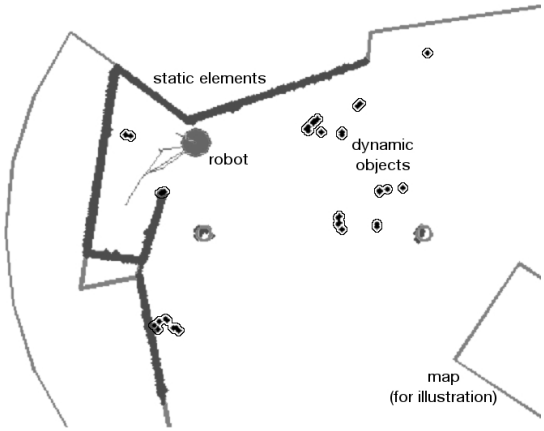
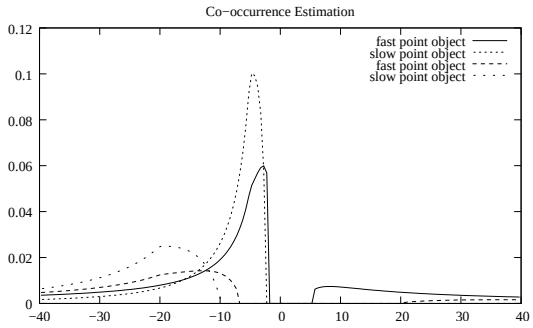


Fig. 5. Example of a motion detection from a mobile robot, using data acquired by the *Photobot* at Expo.02. This was a difficult region for localisation, producing particularly large orientation errors when the robot rotated on the spot. Using SLIP, the scans were aligned very precisely to allow robust 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.

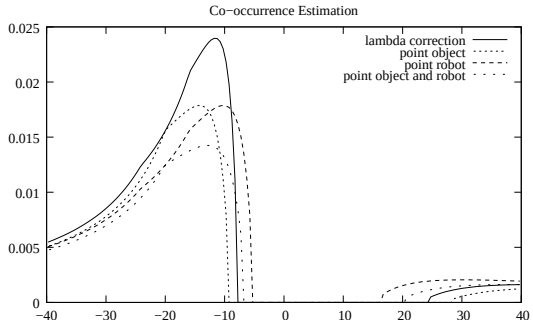
To detect motion on aligned scans, elements without correspondence within a defined distance (derived from the maximal localisation error) are considered outliers. Projective filters are used to distinguish between moving objects and occlusion changes. Non-outliers are used to create a map of the static environment. SLIP then determines which outliers were visible from the previous position. An example alignment result is given in figure 4. 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 5.

4 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 of non-point objects such that it can be fed into the 1D expressions.



(a) effect of relative speed



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

Fig. 6. Co-occurrence probabilities $P_{\mathrm{c},i}$ in the one-dimensional case. The robot is at the origin, collisions may occur on either side of the initial robot position: objects can catch up if they have higher speeds. **Top:** Four cases of dynamic object locations and speeds are shown: objects are located at $x = \{-20, -5\}$ and move with equal or twice the speed of the robot. **Bottom:** The co-occurrence 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. In this example, the object is located at $x = -20$ and moves twice as fast as the robot. A radius of $r_{\mathrm{r}} = r_i = 4$ is used.

4.1 One-Dimensional 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} (as fast as possible) to $R(x)$, and then estimate the probability of the object being there as well. We can easily compute the time it takes the robot to reach any point at distance λ_{r}. By defining a stochastic process for the movement of the object over the dis-

tance λ_i from its current (estimated) position to $R(x)$, we can compute the probability that the object will be there when the robot arrives. We assume the object's velocity to be bounded by v_i, but that the direction of motion can change at each instant. The robot and object are considered to be at the same location if they are within the same interval of size δ, which discretizes space into grid cells.

We start by developing the co-occurrence estimation $P_{\mathrm{c},i}$ in the one-dimensional case, and then we reduce locations in the workspace $\mathcal{W}$ to this 1D formulation. Figure 6 shows the form of probability densities that we calculate. $P_{\mathrm{c},i}$ (1) 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.

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

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:

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

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 $p_{\mathrm{c}} = 0$.

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

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

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

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

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

In general, the robot may travel on arbitrary collision-free 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.

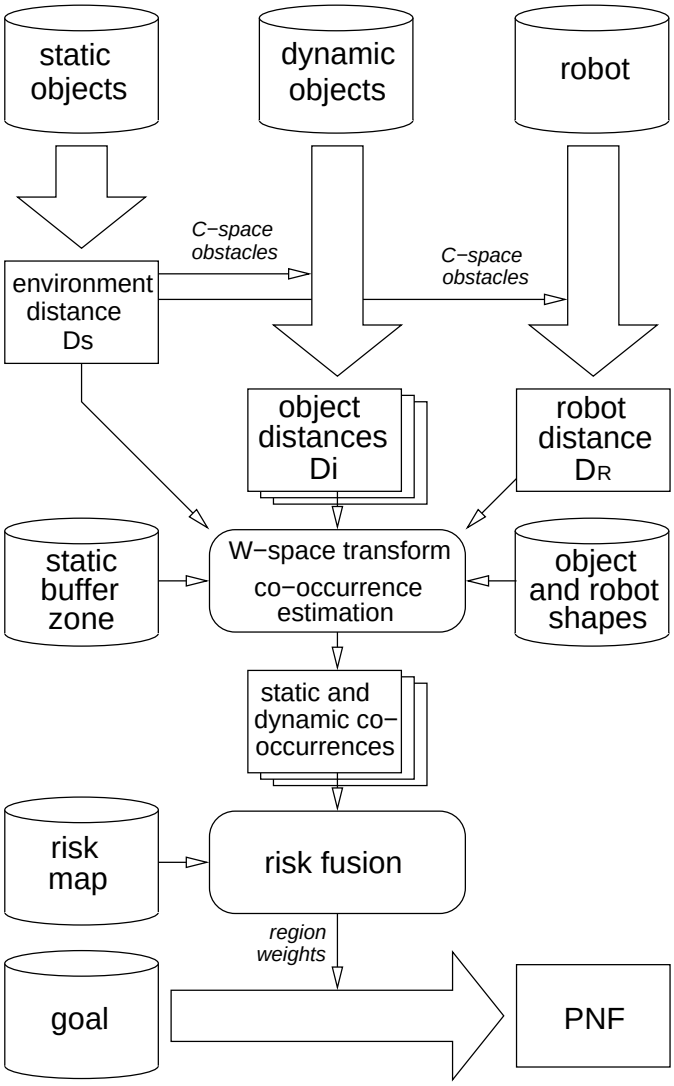


Fig. 7. Processing flow inside the PNF. A-priori information and inputs are denoted with cylinders, rectangles represent the various grid layers, big arrows denote E^* operation, rounded boxes are other types computations. Note that $\mathcal{C}$ obstacles are essentially binary maps of weighted regions that represent each object's and the robot's $\mathcal{C}$-space.

4.2 Risk Fusion and Planning

How can the co-occurrence information be used for path planning? The PNF is based on a combination of co-occurrence probabilities estimated in workspace $\mathcal{W}$, which is then fused and transformed to configuration space $\mathcal{C}$. The main

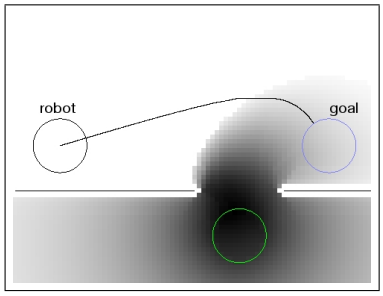
contribution lies in the formulation of multiple layers of co-occurrence probabilities, and how these are fused into a risk map representing all dynamic objects as well as the environment topology. A flow diagram of the PNF algorithm is given in figure 7:

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 $\lambda_i(x) = \min_{y \in A_i(x)} (D_i(y))$, 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_r 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.

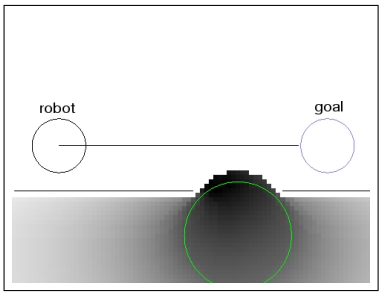
5 Results

Figures 8(a) and 8(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 8(c) and 8(d) show the effect of adding an object that moves with the same speed as the robot, in this case making the path switch topology. Note the zero-risk zone around the robot (compare with figure 6(a)) and the clear line between the robot and the second object in 8(d).

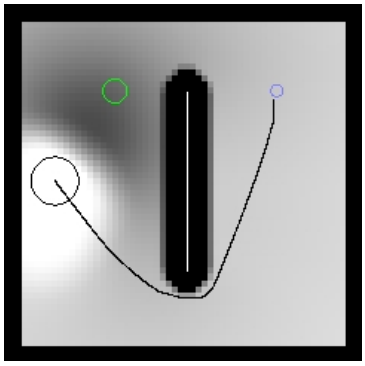
The various mappings in these figures are very smooth, which is a consequence of the interpolation in E*, and could not be achieved with any strictly



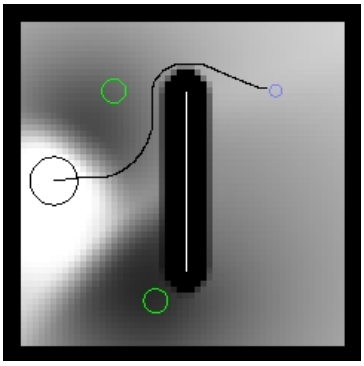
(a) $P_{c,i}$: small object



(b) $P_{c,i}$: big object



(c) P_r: $v_i = 2.5v_r$



(d) P_r: $v_j = v_r$

Fig. 8. As shown in the two examples on the top, PNF takes into account the environment topology individually for each object: When an object is too large to fit through an opening, it cannot interfere with the robot trajectory. The two examples on the bottom illustrate how different object speeds affect the trajectory, and how the addition of a dynamic object can influence the topology of the chosen path.

graph-based method. It has been shown in [13] that in the case of binary obstacle information, E* produces navigation functions very close to the true Euclidean distance (less than 10% error in typical indoor environments), whereas non-interpolating graph planners distort the distances because of the discreteness they impose on the motion choices of the robot. Interpolated dynamic replanning can be done at relatively little extra cost compared to the strictly graph-based D* [19]: the increase per operation is less than 40%, with fewer than 65% increase in the number of operations.

6 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. 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. 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.
2. Enrique J. Bernabeu, Josep Tornero, and Masayoshi 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.
3. P. J. Besl and N. D. Kay. A method for registration of *-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.
4. Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using the relative velocity paradigm. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1993.
5. Th. Fraichard and H. Asama. Inevitable collision states - a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.
6. Thierry Fraichard. Trajectory planning in a dynamic workspace: a 'state-time space' approach. *Advanced Robotics*, 13(1):75–94, 1999.
7. David Hsu, Robert Kindel, Jean-Claude Latombe, and Stephen Rock. Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, 21(3):233–255, 2002.
8. Björn Jensen. *Motion Tracking for Human-Robot Interaction.* PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2004.

9. O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1), 1986.
10. Frederic Large, Sepanta Sekhavat, Zvi Shiller, and Christian 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.
11. J.-C. Latombe. *Robot motion planning*. Kluwer Academic Publishers, Dordrecht, Netherlands, 1991.
12. J. S. B. Mitchell and C. H. Papadimitriou. The weighted region problem: Finding shortest paths through a weighted planar subdivision. *Journal of the ACM*, 38(1):18–73, 1991.
13. Roland Philippsen. *Motion Planning and Obstacle Avoidance for Mobile Robots in Highly Cluttered Dynamic Environments*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2004.
14. Roland Philippsen and Roland Siegwart. Smooth and efficient obstacle avoidance for a tour guide robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.
15. Roland Philippsen and Roland Siegwart. An interpolated dynamic navigation function. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005.
16. N. C. Rowe and R. S. Alexander. Finding optimal-path maps for path planning across weighted regions. *International Journal of Robotics Research*, 19(2):83–95, 2000.
17. 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.
18. Roland Siegwart, Kai-Oliver Arras, Samir Bouabdallah, Daniel Burnier, Gilles Froidevaux, Xavier Greppin, Björn Jensen, Antoine Lorotte, Laetitia Mayor, Mathieu Meisser, Roland Philippsen, Ralph Piguet, Guy Ramel, Gregoire Terrien, and Nicola Tomatis. Robox at Expo.02: A large-scale installation of personal robots. *Robotics and Autonomous Systems*, 42:203–222, 2003.
19. Anthony Stentz. The focussed D^* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.
20. Jur van den Berg, Dave Ferguson, and James Kuffner. Anytime path planning and replanning in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006.
21. Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, 1994.