A Navigation Framework for Multiple Mobile Robots and its Application at the Expo.02 Exhibition

Kai O. Arras, Roland Philippsen, Nicola Tomatis, Marc de Battista, Martin Schilt, Roland Siegwart

Autonomous Systems Lab
Swiss Federal Institute of Technology Lausanne (EPFL)
CH–1015 Lausanne, Switzerland, kai-oliver.arras@epfl.ch

Abstract

This paper presents a navigation framework which enables multiple mobile robots to attain individual goals, coordinate their actions and work safely and reliably in a highly dynamic environment. We give an overview of the framework architecture, its layering and the subsystems reactive obstacle avoidance, local path planning, global path planning, multi-robot planning and localization. The latter receives particular attention as the localization problem is a key issue for navigation in unmodified and difficult environments. The framework permits a lightweight implementation on a fully autonomous robot. This is the result of a design effort striving for compact representations and computational efficiency.

The experimental testbed was the "Robotics" pavilion at the Swiss National Exhibition Expo.02 where ten fully autonomous robots were interacting with more than half a million visitors during a five-month period on 3,316 km.

1. Introduction

Navigation responds to three questions: 'where am I?', 'where am I going?' and 'how do I get there?'. A navigation framework has the task to offer the simplest possible interface to these questions, hiding their complexity to the user or the application layer.

At the inside, navigation deals with various constraints on different time scales and levels of abstraction. A common approach to structure the problem is a three-layered architecture which consists in a planning layer, an execution layer and a reactive behavior layer [6, 23] (figure 2).

- The *planning layer* decides how to achieve high-level goals using a model of the environment. Planning takes places under constraints of task-specific cost functions and limited resources.
- The *execution layer* subdivides a plan into executable subplans, activates and deactivates behaviors and supervises their completion.
- The *reactive behavior layer* interfaces the robot's sensors and actuators. In mobile applications, it acts as a position controller under constraints of a dynamic environment, the robot shape, vehicle kinematics and dynamics.

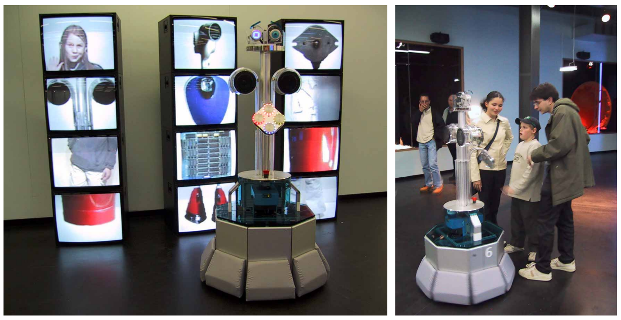

Figure 1. *Robox at Expo.02*

Controls, abstracted sensor data and status information flow vertically between layers. Typically, controls such as plans and subplans are passed to lower levels and information such as status codes and termination flags are passed to higher levels. In case of failures in a layer (e.g. path blocked), requests for revised controls are sent to higher layers (e.g. replanned path). Time scale and abstraction increase from bottom to top, model fidelity and real-time concerns increase from top to bottom.

We adopt this three-layered architecture here as it accommodates deliberative and reactive behaviors, allows for constraint distribution over the layers and embodies an intuitive way of increasing abstraction from bottom to top. It is further suitable for applications with manipulators and multiple robots [6, 23, 24].

The application we envisage are ten fully autonomous mobile robots deployed in a mass exhibition with up to 500 visitors per hour. Their task includes tour giving, en-

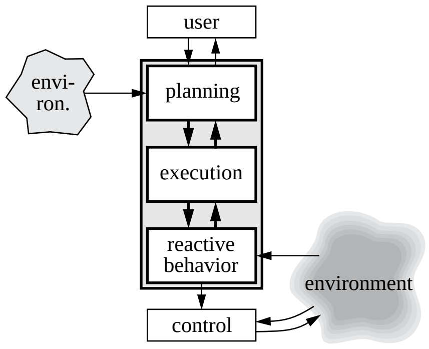

Figure 2. *The layered navigation framework*

tertaining and picture taking of visitors. They share the same space and the same goals for the tours and operate in an unmodified environment. The paper presents how the architecture was adapted to suit our needs and the choices we made for its components. Further, operating experience with this framework from in the “Robotics” pavilion at the Swiss National Exhibition Expo.02 is discussed.

2. The Navigation Framework

Today we (still) face limited computational resources in embedded systems for real-time. While radio-linked off-board hardware might be an alternative for a single robot, it would result in prohibitive bandwidth costs for multiple robots, if, for instance, raw sensor data for localization were transmitted. As we want the robots to be truly autonomous, we look out for compact representations and computational efficiency.

2.1 Environment Model

Our approach to environment modeling is feature-based using geometric primitives such as lines, segments and points (sometimes called landmarks). The environment topology is encoded in a weighted directed graph with nodes and edges between the nodes. Neither for global path planning nor for localization we use a free space model like occupancy grids. The advantage of this choice is compactness: in indoor environments, a map of this type (features plus graph) requires typically around 30 bytes per m^2. Further, scaling to 3d is polynomial, whereas grid maps scale exponentially.

The graph has two types of nodes: *station nodes* and *via nodes*. Station nodes correspond to application-specific (x, y, θ)-locations in space with a meaning. Examples from Expo.02 include: showcase with industrial robot, tour welcome point or location to trigger picture caption. Via nodes have two tasks. First, they correspond to topology-relevant locations like doors or corridor-crossings. Thereby the graph models the environment topology. Second, in environments with large open spaces, they further provide topological redundancy by locations with favorable traversability. Favorable, for instance, with respect to visitor flow criteria or other specific requirements from the application.

The map further contains so called *ghost points*. Ghost points are (x, y)-positions in the world reference frame which act as invisible barriers. If the environment contains forbidden areas undetectable for the robot’s sensors (e.g. staircases, glass doors, exits, etc.) ghost points are used to prevent the robot to go there by injecting them into the sensor data as virtual readings (see also section 2.4 and [12]).

The Expo.02 environment covers a surface of 315 m^2 and has 15 places of interest for the robots. The map contains 44 segments on 44 lines, 15 station nodes, 37 via

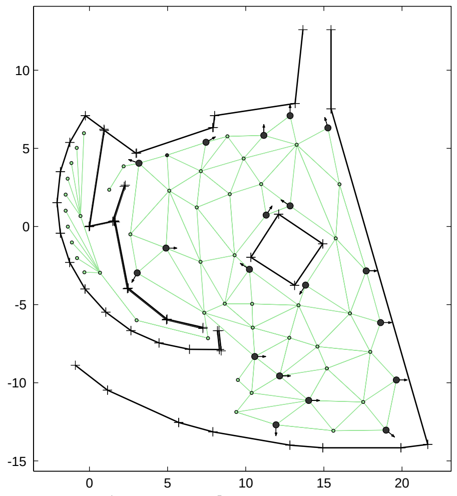

Figure 3. *The Expo.02 map*

nodes and 20 ghost points (figure 3). Its exact memory requirement is 8 kbytes or 26 bytes per m^2. The ghost points (not shown in fig. 3) are at the entrance (bottom of fig. 3) and the exit (top) of the circulation area.

2.2 Global Path Planning

With a topological graph, global path planning becomes a graph search problem for which a multitude of algorithms exist. From simple depth-first search with fixed costs to dynamic programming techniques and probabilistically learned cost functions for edge traversability [15]. Global path planning in our case uses a priority-first search [22] and costs assigned to edges and nodes [27]. In a single-robot context costs are fixed, for multi-robot planning the costs of nodes are variable and depend on the distance to other robots. In the multi-robot case, besides paths, goals are shared as well, and must be negotiated among the robots. See section 2.5.

At Expo.02 we give visitors the opportunity to choose their next station of a tour by themselves. This closes the first loop to the environment which is asynchronous and has a cycle time in the order of 0.01 Hz (figure 7). Path planning in the graph of figure 3 is a matter of a few milliseconds in our implementation.

2.3 Command Queue

Paths from the global planner consist in a list of nodes. In the execution layer, a command queue passes the list to the behavior layer in a node-by-node manner. For via nodes it actives a (x, y)-position-only variant of the obstacle avoidance, in case of the last list node, the full pose (x, y, θ) must be met. Near the last node in the list (typically a station node), localization gets deactivated (10 cm in our implementation). This is because localization causes the robot pose estimates to be noisy such that for any position controller the goal cannot be reached. The reached condition for via nodes is also treated on this level. It is satisfied when the robot enters a large disk around the node (2 m in our implementation).

2.4 Local Path Planning and Obstacle Avoidance

For mobile robots, the primary role of the reactive layer is that of a position controller. A number of constraints must be accounted for on this level: vehicle shape, vehicle kinematics, vehicle dynamics and, of course, environment dynamics which in the case of a mass exhibition is extensive. Since purely reactive obstacle avoidance methods usually suffer from local minima problems, we divide the task into a reactive and a path planning sublayer [7, 21, 2].

For the reactive sublayer we rely on the idea of the dynamic window approach (DWA) [11]. The method uses a simple model of the vehicle dynamics (maximal acceleration / deceleration) and can – with the appropriate extension [21, 2] – take into account an arbitrary robot shape. In comparison to the original version of the DWA our approach differs in the following points:

- Working with differential drive robots, the objective function trading off speed, heading and clearance are calculated in the actuator phase space (v_l, v_r) instead of the Cartesian (v, w) -space. This models the acceleration limits of the vehicle physically more properly.
- As in [21] and [2], we account for polygonal robot shapes. The robot shape is not hard-coded in our implementation and can be specified at boot time.
- Instead of using the distance to collision as a clearance measure, we use time to collision. This solves a singularity when turning on the spot (any collision would seem instantaneous because the distance travelled seems zero). It also means the robot will choose more clearance when travelling at higher speeds.
- Ghost points from the global map are taken into account. After a global-to-local transform they are injected as virtual sensor readings.

The dynamic window is part of the time- and safety-critical software of our robot. We therefore install this process as a deadline-driven real time task with a 10 Hz frequency. Special attention was paid to optimize its execution time to be short and predictable. For reasons similar to those mentioned in [21], we use a look-up table for the clearance measure.

The second sublayer is a path planner which operates locally as it relies on sensory data without memory. A modified elastic band [20] is employed which generates smooth trajectories around obstacles (figure 4) and uses an NF1 navigation function [17] for initialization. Although the NF1 always yields topologically correct solutions (within its scope and if a solution exists), it generates unsmooth trajectories with the tendency to graze obstacles.

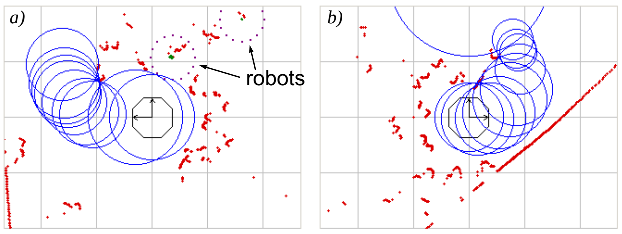

Figure 4. *Two situations from Expo.02 which show how the elastic band finds a smooth path around people. In a) there are two robots virtually blown up with ghost points.*

With the elastic band, The initial NF1 plan continuously evolves towards a smoother curve.

Updates of the band are implemented in a non-time critical thread which runs at several Hz. As soon as the elastic band "snaps", replanning is initiated. At Expo.02, this takes place typically in the order of 0.1 Hz.

For path planning and evolution of the elastic band, the robot is assumed to be circular and heuristics are used to ignore some sensor readings. This results in simplified and speed-up implementations. The simplifications are acceptable because the DWA ensures the dynamic, kinematic, and geometrical constraints.

The modifications of the DWA and the elastic band are described in more detail in [19]. At the lowest level finally, the speed controller, also installed as a real-time task, runs at a 1 kHz frequency (figure 7).

2.5 Multi-Robot Planning

For multi-robot planning we distinguish goal coordination and path coordination.

For the paths, environment dynamics from visitors is important. Visitors who play with a robot easily deviate the vehicle from its path and provoke path collisions where at planning time no collision had occurred. We therefore face the problem of replanning paths for all robots on-the-fly. This has to happen in real-time since we do not want the robots to stop and wait each time we detect a path deviation somewhere.

For this we employ a potential field approach where graph nodes receive costs proportional to the distance to a robot. Each robot assigns weights to those nodes which are in its current plan. The resulting graph superimposes the weights of all robots. Using this graph, the global planner delivers cost-optimal paths that contain nodes which are currently unused by other robots.

Theoretically, there is no guarantee on collision freeness with potential fields. One can construct (pathological) situations where it is cheaper for two robots to use the same node at the same time. More sophisticated methods [17] can provide a guarantee but are computationally unfeasible in our application. The advantage of this technique is its efficiency. It enables multiple robots to adapt their plans quickly and requires a minimum of shared information: a list of elements (i, w_i), with i being the node identifier and w_i its weight – a matter of a few bytes.

Coordination of the goals is necessary since a limited number of shared goal locations is to be allocated to multiple robots. Path coordination cannot avoid that several robots choose simultaneously the same station node since robots with the same goal would, due to the lack of redundancy, insist to go there regardless the costs.

To give visitors the choice of their next tour station the robot makes a proposition which is based on the currently unoccupied stations, the list of stations included in the tour and the stations already visited. The selected station is then reserved for this robot. More details on goal negotiation and implications for visitor flow can be found in [16].

2.5.1 The Robot-Sees-Robot Problem

Multi-robot coordination so far presented, depends on the knowledge of the robot positions. Even with a reliable and accurate localization, this creates critical interdependencies. Robots shall therefore be able to see each other on a raw data level. This, however, is not straight-forward with platforms of the same mechanical design which all measure at the same height since, at this height, the true vehicle size will be underestimated from the sensor readings.

Our approach is to mount two retro-reflecting beacons onto the vertical profiles in the blind zone between the two Sick laser scanners (figure 1). The Sick LMS 200 sensors provide an intensity signal which allows to easily extract the reflector information. We then use ghost points to artificially create a virtual robot contour at the extracted reflector positions (see also figure 4). Thus, obstacle avoidance provides an anytime fall-back solution.

2.6 Localization

Localizing a robot robustly in a mass exhibition environment is certainly a challenge. In former exhibition projects, localization was based on off-board hardware [8, 26] or environment modifications [18]. In our earlier work we employed features and an extended Kalman filter (EKF) [3]. This is also the approach for the three museum robots described in [13]. However, a robot doing (single-hypothesis) pose tracking can loose its track as the inherent data association problem is ignored. With the localization technique introduced in [1], we address the data association problem and extend the conventional EKF localization approach to a global localization technique.

Unlike POMDP or Markov approaches [8, 26] where locations[1] are generated before they get evaluated by the exteroceptive sensors (as a grid or as particles), our approach to localization turns this process around: locations are generated as a direct consequence from sensory information. Features tell us *when* and *where* to place a location hypothesis. This allows to maintain as many hypotheses as necessary and as few as possible.

The technique for hypothesis generation is a constraint-based search in an interpretation tree [14, 10, 9]. This tree is spanned by all possible local-to-global associations, given a local map of observed features $L = \{l_i\}_{i=1}^{p}$ and a global map of model features $G = \{g_j\}_{j=1}^{m}$. Furthermore, besides track formation, in [1] we present an algorithm for track splitting under geometric constraints. It relies on the same idea as hypothesis generation (search in an interpretation tree), thus forming a consistent framework for global EKF localization.

We briefly outline the approach (more details in [9,1]): The search space for hypothesis generation is the space of all possible associations of the observed features l_i and the modeled features g_j. The search space has the structure of a tree with p levels and $m+1$ branches [14]. p is the number of observed features in L, m the number of modeled feature in G. The extra branch (called star branch) allows correct associations in the presence of outlier observations (false positives) and thus accounts for environment dynamics and map errors. During tree traversal, statistically feasible *pairings* $p_{ij} = \{l_i, g_j\}$ are sought given all uncertainties associated to the features. A pairing says that the observed feature l_i and the modeled feature g_j denote the same physical object in the environment (g_j is called an 'interpretation' of l_i). Although the problem is of exponential complexity, geometric constraints reduce enormously the space to be explored. The constraints can be classified into two categories [9], *location independent constraints* (unary and binary) and *location dependent constraint* (rigidity, visibility and extension). The latter category requires the robot location L_h:

Unary constraint. We accept the pairing p_{ij} if l_i and g_j are of the same type, color, size or any other instrinsic property. Examples: l_i and g_j are both features of type (x, y)-point, or the length of the observed segment l_i is smaller or equal than the length of the modeled segment g_j.

Binary constraint. Given a valid pairing p_{ij} we will accept the pairing p_{kl} only if the two local features l_i and l_k are compatible to the two global features g_j and g_l. Examples: l_i and l_k are lines with the angle φ_{ik} between the lines. Then, the pairing p_{kl} is considered compatible if the angle φ_{jl} is the same. With point features, for instance, the distances $l_i - l_k$ and $g_j - g_l$ must correspond.

Visibility constraint. This constraint only applies to model features. It tests whether g_j is visible from the robot position L_h. Example: lines or segments can always be seen only from one side. If the robot is behind a wall, one of the two lines modeling the wall is invisible. With sensor specific parameters, the visibility constraint rejects features

1. We use the terms *location*, *position* and *pose* interchangeably. They denote all the full (x, y, θ) vehicle pose

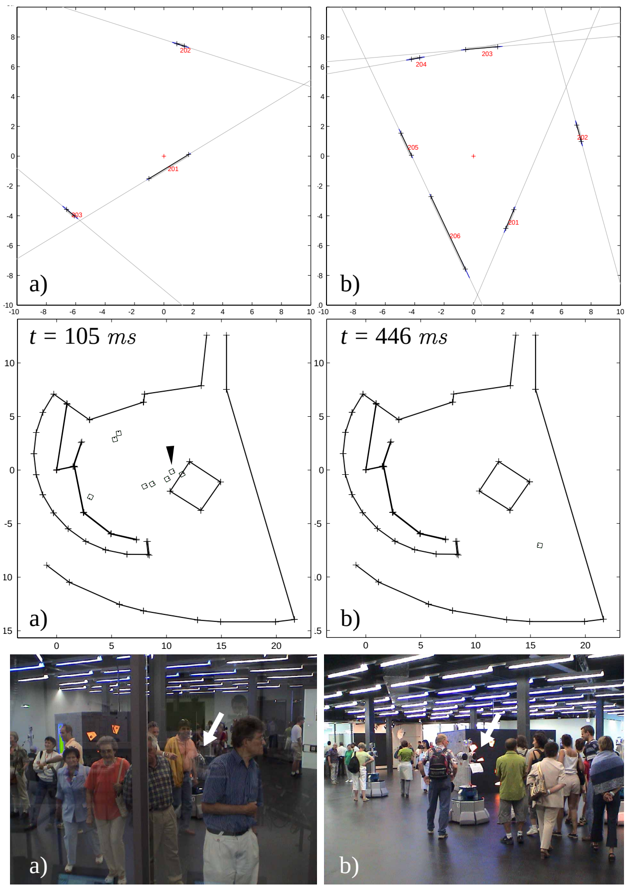

Figure 5. *Global EKF localization. Given the local maps in a) and b), hypotheses are generated at locations where the local map geometrically 'fits' into the global map. In a) there are 8 hypotheses; the location is ambiguous. In b) there is a single hypothesis; the robot is instantaneously localized. 't' denotes the execution time. The experiments were carried out at the positions shown below.*

which are not detectable, for instance, because they are farer away than a maximal perception radius.

Rigidity constraint. A pairing p_{ij} is considered compatible if l_i and g_j, transformed into the same coordinate system given L_h, coincide (are at the same position). This is what happens in the matching step of any EKF localization cycle. Usually, g_j is transformed into the frame of l_i.

Extension constraint. A pairing p_{ij} is considered compatible if l_i and g_j, transformed into the same coordinate system given L_h, fully overlap. Example: an observed segment l_i must be fully contained in the transformed g_j seen from the location L_h.

These constraints allow to discard whole subspaces (subtrees) from the search each time when an incompatible pairing is found at the root of such a subtree. With the uncertainties associated to local and global features, all decisions make use of the Mahalanobis distance on a significance level α.

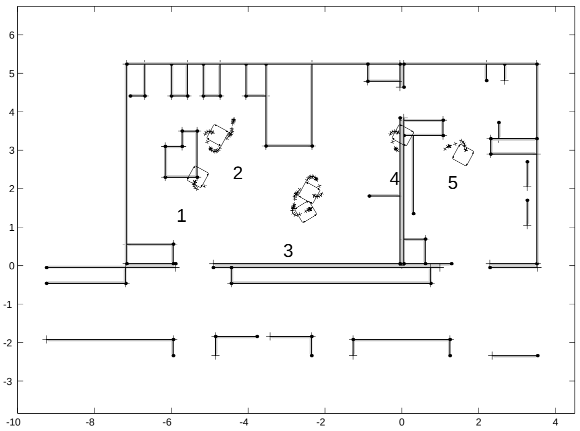

Figure 6. *Multi-hypothesis tracking. Starting with five hypotheses, track #3 turns out to be the true one after the last track (#2) was rejected at a distance of 1.89 meters.*

Tree traversal is implemented as a recursive back-tracking search algorithm [9]. The strategy is to first find a minimal number of valid pairings with location independent constraints such that a location estimate can be determined in order to apply location dependent constraints, too. Each time when the algorithm reaches the bottom of the tree, that is, the end of a branch where all observed features l_i could have been assigned to a model feature g_{j_i} or to the star-branch, we have a valid robot location hypothesis $h = \{S_h, L_h\}$. The pairings which support the hypothesis are put together into the *supporting set* $S_h = \{p_{ij_i}\}_{i=1}^{p}$.

2.6.1 Estimating the Robot Location

With the supporting set, the (x, y, θ)-pose of the robot is not yet known. This is what the extended information filter (EIF) does. Given the set of pairings with all associated uncertainties, it estimates the robot location and its covariance in the least square sense. The difference between the EIF and the EKF is that the former is the batch estimator formulation of the latter (which is recursive). This is needed, because, during hypothesis generation, there is no a priori knowledge on the robot location which formally means that the state prediction covariance, usually called $P(k|k+1)$, is infinite. With the EIF, this can be properly expressed as $P^{-1}(k|k+1) = 0_{3\times 3}$ since covariance matrices are represented in the information matrix form, that is, by their inverse.

Figure 5 shows two examples of hypothesis generation in the Expo.02 environment. With multiple discrete hypotheses, to be localized is simply expressed as having a single hypothesis. For line extraction we use the method described in [4]. Extraction times are around 20 ms. Localization was implemented as a non-RT thread. Its cycle time was slowed down to 2 Hz in favor of other concurrent non-RT processes (e.g. path planning, communication). With a single hypothesis to be tracked, cycle times of 10 Hz and more are achievable on the Robox hardware.

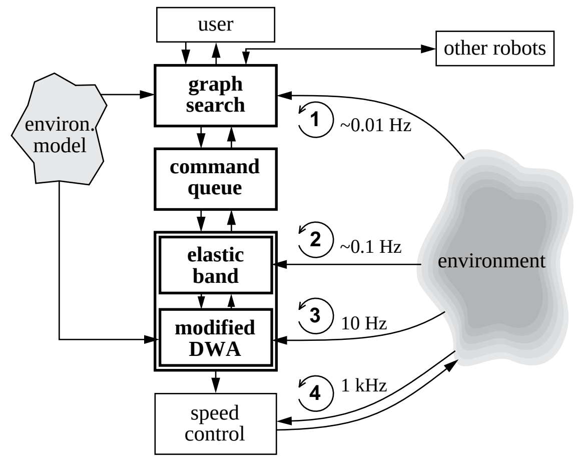

Figure 7. *Layering and control loops with their precise (3, 4) and typical (1, 2) cycle time.*

2.6.2 Multi-Hypothesis Tracking (MHT)

The main reason for lost situations during tracking is incorrect data association. This occurs typically when there is more than one statistically feasible pairing candidate for an observation. Choosing the closest one is the most widely applied strategy called the nearest neighbor standard filter. If it was the wrong one, the KF will become inconsistent and is very likely to diverge. Therefore, besides uncertainties in the *value* of measurements, robust pose tracking must also account for uncertainty in the *origin* of measurements [5].

We look for an algorithm which re-generates hypotheses during tracking as soon as there is no guarantee anymore that the correct association of an observation can be done. In [1] we propose an algorithm for track splitting under geometric constraints which, given a predicted location, a local and a global map, splits up into multiple offspring hypotheses if there is statistical compatibility with several supporting sets. It has the identical structure than the algorithm for hypothesis generation but employs location dependent constraints *only* and does *not* recur with a refined position estimation. In this manner the algorithm finds all supporting sets in the vicinity of the initially predicted location L_h.

After each hypothesis has been tracked, there are three cases: *(i)* hypothesis confirmation, *(ii)* hypothesis rejection and *(iii)* hypothesis split up. Track rejection takes place when the predicted location is not supported anymore by location dependent constraints on the level α. When track splitting occurs, their locations get newly estimated and the best one is taken. "Best" in the sense of most paired observations, and in case of a tie in a goodness-of-fit sense, expressed by the joint Mahalanobis distance. This is a non-Bayesian approach to data association. Fig. 5.a) shows an example: the marked (true) hypothesis is the only one with three paired observations, while the seven other hypotheses have only two pairings and a star-branch match.

Figure 6 shows an experiment how the robot converges towards the true location after a short trajectory. Note that by geometry only (or geometric falsification respectively), false tracks get rejected quickly. No free-space information is needed.

Finally, figure 7 shows the resulting decomposition of the architecture, the four control loops and their cycle times. The DWA sublayer requires the position of the ghost points which explains its connection to the environment model. Localization can be seen as a behavior as it is switched by the execution layer and connected to the robot's sensors. It is however invisible since there is no direct connection to other components or behaviors. It acts in the background continuously correcting the odometry. Thereby, all other framework components can consider odometry as being "perfect". For multi-robot coordination, robots are connected on the level of the planning layer.

3. Implementation

In view of the requirements from the Expo.02 application and former experience in robot design and system integration in our lab, the decision was taken to construct a robot from scratch.

The outcome, *Robox*, is shown in fig. 1 and is described in detail in [25]. For navigation, a PowerPC G3 at 380 MHz serves as main CPU running the deadline-driven real-time operating system XO/2. The robot sports two SICK LMS 200 and is made for fully autonomous operation. It does not rely on off-board resources with one exception at Expo.02: a central server for multi-robot planning. The approach described in section 2.5 does not require this at all but with the implementation using HTTP, opening a connection took more time than data transmission itself. The central unit allowed us to optimize latencies and transmission times (see also [16]).

4. Operation Experience and Discussion

During Expo.02 (May 15th–October 20th, 2002), 686'405 persons visited the "Robotics" exhibition (4317 per day, around 400 per hour). Assuming an average visit duration of 15 minutes (typical for mass exhibitions), around 100 persons share the 315 m^2 circulation area with ten robots. In other words, the robots encounter environment dynamics similar to a railway station at rush-hour, mostly benign but also hostile (people who run into, play with, try to outwit or kick the robots). The overall travel distance was 3,316 km during an overall operation time of 13,313 hours.

Safety. It was never observed that a robot was the cause of a dangerous situation e.g. with small children, elderly or handicapped people (fig. 8). The lack of additional sensors close above floor (IR or ultrasonic) is easily bearable with the combination of tactile plates and soft bumpers.

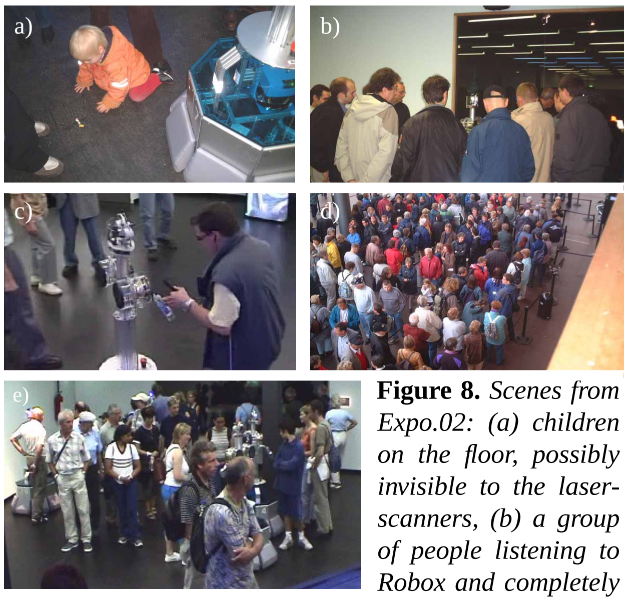

Figure 8. *Scenes from Expo.02: (a) children on the floor, possibly invisible to the laser-scanners, (b) a group of people listening to Robox and completely blocking its path, (c) a blind visitor interacting with Robox, (d) the queue in front of the Robotics pavilion, (e) a compact group of visitors (~30) and robots (four).*

Blocked-situations due to bumper contact were also handled by the interactive part (robot expressing friendly menaces). Further, there was almost no vandalism. Visitors kicked the bumpers regularly and touched the (unprotected) two pan-tilt eyes of the face but never caused a real damage.

Reliability. The division of obstacle avoidance into a purely reactive part with high model fidelity and a planning part is a powerful conjunction. Very often, groups of visitors form a *U*-shaped obstacle or leave only small “holes” for passage. The robots have no difficulty to escape from such situations, and due to the fact that we account for Robox’ true octagonal shape, narrow passages are efficiently used (fig. 4). Further, the elastic band generates smooth and good-looking trajectories. No problems with mutually interfering laser scanners from different robots have been observed.

We were surprised by the reliability of localization in view of the environment dynamics and the fact that we used laser data only. A fall-back solution with lamp features extracted from a camera looking to the ceiling was prepared but had not to be employed. However, lost situations occurred for two main reasons: staff members that push/rotate the robot in order to untangle congestions of robots or visitors and robots (see below), and visitors imitating this behavior. Global localization as illustrated in fig. 5 was then useful since it allowed to instantaneously relocalize the robot within all people, enabling it to resume operation. The geometry of the Expo.02 environment was helpful here since it contained little symmetry.

We believe that the use of geometric features for navigation is a very appropriate choice for localization, particularly for highly dynamic environments. During feature extraction, sensor readings are filtered out that do not satisfy the spatial model assumption from the feature (lines in our case). Thereby, the extraction process says which reading is to be taken for localization and which one is to be ignored. This filter relies typically on sound regression techniques and works independently on whether the robot is localized or not. People standing in front of the robot do not produce evidence for the line extraction since legs are not linear or too short (see also [4]). Spurious segments, for example by line-like objects carried by people, do occur and are treated by the star-branch in the localization algorithm.

The environment contained walls which differ by less than five degrees in angle (bottom wall in fig. 3). Line extraction had to be tuned for a slight tendency to oversegmentation in order to correctly detect the true wall segments. Otherwise the observed line segments would have been too long and the unary constraint testing on the segment length would have prevented their match.

However, path-coordination (section 2.5) was not always available and the robots relied on the fall-back solution of section 2.5.1 during this period. But the concept for the robot-sees-robot problem turned out to be an oversimplification. The employed model (blowing up a virtual contour at the detected reflector position) caused the robots occasionally to block each other. This happened when the reflectors of two nearby robots were occluded (e.g. by visitors) and became suddenly visible (when they moved away) causing the virtual contours to overlap and to be within the hull of the other robot. The motion planner then stopped both vehicles and pavilion staff members had to intervene. By using a switch disconnecting the motors from the amplifiers, the 115 kg robot can be easily pulled away (such an intervention took a few seconds). However, wrong manipulation of the switch made the robot slip and caused unmodeled odometry errors of such an extent that the robot often went lost during such interventions (robot kidnapping). Multi-hypothesis tracking presented in section 2.6.2 aims at that problem. But recovering from robot kidnapping is hard. The main problem is to reliably decide *when* in the set of locally generated hypotheses the robot has to jump to *which* hypothesis. The question persists and motivates future work in this direction.

Precision. Localization accuracy varies and depends on how much persons occlude the sensors and how much segments have been paired. Goal nodes were typically reached with a relative error less than or equal to one centimeter. This was measured by the traces on the floor at goal locations the robots left after several months of operation.

5. Conclusions

On 3,316 km travel distance, the framework and its components as presented in this paper met the requirements of safe and reliable operation in a highly dynamic environment very well. To the knowledge of the authors, this project was the world-wide biggest installation of interactive autonomous robots so far. Being the first one, we believe that it will not remain the last robot exhibition of its kind. We have shown that the deployment of autonomous, freely navigating robots in a mass exhibition is feasible.

We have also demonstrated that the feature-based approach to localization is an adequate choice for highly dynamic environments. This has been showed in former work with two museum robot installations [8, 26] for the Markov localization approach. From its application at Expo.02, we conclude that the feature-based approach allows to be robust and precise, retaining its efficiency also in the "globalized" form of section 2.6.

Finally, a mass exhibition with robots is not the experimental platform as we initially hoped. The pavilion was open between ten and a half and twelve hours per day, nonstop during 159 days. In the evening batteries were empty and had to be charged for the next morning. All experiments had to be done under daytime conditions, that is, within the visitors. The contracts did not allow to reduce or even close the exhibition.

Partly as a consequence, the robots at Expo.02 still required manual intervention. The main reason for this is twofold: the abovementioned problems on the level of the robot collective and the sheer mass of people on a relatively small surface exposing and amplifying the slightest insufficiency in hardware and software.

The problems encountered reveal relevant directions for future research: MHT to address the robot kidnapping problem and refined models of visitors and robots for multi-robot coordination in populated environments.

Further spin-offs of this project were: a great visitor feedback, a strong team effort, close collaboration with exhibition makers, visitor flow experts, architects, scenographers and industrial designers, and an overall unique experience in many technical and non-technical respects.

References

[1] Arras K.O., Castellanos J.A., Siegwart R., "Feature-Based Multi-Hypothesis Localization and Tracking for Mobile Robots Using Geometric Constraints," IEEE Int. Conf. on Robotics and Automation, Washington DC, USA, 2002.

[2] Arras K.O., Persson J., Tomatis N., Siegwart R., "Real-Time Obstacle Avoidance for Polygonal Robots With a Reduced Dynamic Window," IEEE Int. Conf. on Robotics and Automation, Washington DC, USA, 2002.

[3] Arras K.O., Tomatis N., Jensen B., Siegwart R., "Multisensor On-the-Fly Localization: Precision and Reliability for Applications," Robotics and Autonomous Systems, vol. 34, issue 2-3, pp. 131-143, February 2001.

[4] Arras K.O., Siegwart R.Y., "Feature Extraction and Scene Interpretation for Map-Based Navigation and Map Building," Proc. of SPIE, Mobile Robotics XII, Vol. 3210, 1997.

[5] Bar-Shalom Y., Li X.-R., Multitarget-Multisensor Tracking: Principles and Techniques, ISBN 0-9648312-0-1, 1995.

[6] Bonasso R.P., Firby R.J., Gat E., Kortenkamp D., Miller D., Slack M.,"Experiences with an Architecture for Intelligent, Reactive Agents," Journal of Experimental and Theoretical Artificial Intelligence 9(2), 1997.

[7] Brock O., Khatib O., "High-Speed Navigation Using the Global Dynamic Window Approach," IEEE Int. Conf. on Robotics and Automation, Detroit, USA, 1999.

[8] Burgard W., Cremers A.B., Fox D., Hähnel D., Lakemeyer G., Schulz D., Steiner W., Thurn S., "Experiences with a Interactive Museum Tour-Guide Robot," Artificial Intelligence 00(1999): 1-53, 1999.

[9] Castellanos J.A., Tardos J.D., *Mobile Robot Localization and Map Building: A Multisensor Fusion Approach*, Kluwer, 1999

[10] Drumheller M., "Mobile Robot Localization Using Sonar", IEEE Transactions on PAMI, 9(2), p. 325-32, 1987.

[11] Fox D., Burgard W., Thrun S., "The Dynamic Window Approach to Collision Avoidance," IEEE Robotics and Automation Magazine, 4(1), pp. 23-33, March 1997.

[12] Fox D., Burgard W., Thrun S., "A Hybrid Collision Avoidance Method For Mobile Robots," IEEE Int. Conf. on Robotics and Automation, Leuven, Belgium, 1998.

[13] Graf B., Schraft R.D., et al., "A Mobile Robot Platform for Assistance and Entertaiment," International Symposium on Robotics, Montreal, Canada, 2000.

[14] Grimson W.E.L., Lozano-Pérez T., "Localizing Overlapping Parts by Searching the Interpretation Tree", IEEE Transactions on PAMI, 9(4), p. 469-82, 1987.

[15] Hu H., Brady M., "Dynamic Global Planning with Uncertainty for Mobile Robots in Manufacturing," IEEE Transactions on Robotics and Automation, 13(5):760-7, 1997.

[16] Jensen, B., Froidevaux G., Greppin X., Lorotte A., Mayor L., Meisser M., Ramel G., Siegwart R., "The Interactive Autonomous Mobile System RoboX. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Lausanne, 2002.

[17] Latombe J.-C., *Robot motion planning*, Kluwer, 1991.

[18] Nourbakhsh I., Bodenage J., Grange S., Lutz R., Meyer R., Soto A., "An Affective Mobile Robot Educator with a Full-time Job," Artificial Intelligence 114(1-2): 95-124, 1999.

[19] Philippsen R., Siegwart R.,"Smooth and Efficient Obstacle Avoidance for a Tour Guide Robot", IEEE Int. Conf. on Robotics and Automation, Taipei, Taiwan, 2003.

[20] Quinlan S., Khatib O., "Elastic bands: connecting path planning and control," IEEE Int. Conf. on Robotics and Automation, 1993.

[21] Schlegel C., "Fast Local Obstacle Avoidance Under Kinematic and Dynamic Constraints," IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Victoria BC, Canada, 1998.

[22] Sedgewick R., *Algorithms*, 2nd ed., Addison-Wesley, 1988.

[23] Simmons R., Goodwin R., Haigh K., Koenig S., O'Sullivan J., "A Layered Architecture for Office Delivery Robots," First Int. Conf. on Autonomous Agents, 1997.

[24] Simmons R., Smith T., Dias M.B., Goldberg D., Hershberger D., Stentz A., Zlot A., "A Layered Architecture for Coordination of Mobile Robots," in *Multi-Robot Systems: From Swarms to Intelligent Automata*, Kluwer, 2002.

[25] Tomatis N., Terrien G., Piguet R., Burnier D., Bouabdallah S., Arras K.O., Siegwart R., "Designing a Secure and Robust Mobile Interacting Robot for the Long-Term," IEEE Int. Conf. on Robotics and Automation, Taipei, Taiwan, 2003.

[26] Thrun S., et al., "Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva," Int. Journal of Robotics Research 19(11): 972-99, 2000.

[27] Tschichold-Gürman N., Vestli S.J., Schweitzer G., "Operating Experience with the Service Robot MOPS," 3rd European Workshop on Advanced Mobile Robots, Zurich, 1999.