Lankenau, A., Röfer, T., Krieg-Brückner, B. (2003). Self-Localization in Large-Scale Environments for the Bremen Autonomous Wheelchair. In: Freksa, C., Brauer, W., Habel, C., Wender K.F. (Hrsg.): Spatial Cognition III. Lecture Notes in Artificial Intelligence 2685. Berlin, Heidelberg: Springer, 34-61.

Self-Localization in Large-Scale Environments for the Bremen Autonomous Wheelchair

Axel Lankenau, Thomas Röfer, Bernd Krieg-Brückner

Bremer Institut für Sichere Systeme, TZI, FB3, Universität Bremen,
Postfach 330440, 28334 Bremen, Germany.
alone@tzi.de, roefer@tzi.de, bkb@tzi.de

Abstract. This paper presents RouteLoc, a new approach for the absolute self-localization of mobile robots in structured large-scale environments. As experimental platform, the Bremen Autonmous Wheelchair "Rolland" is used on a 2,176m long journey across the campus of the Universität Bremen. RouteLoc poses only very low requirements with regard to sensor input, resources (memory, computing time), and a-priori knowledge. The approach is based on a hybrid topological-metric representation of the environment. It scales up very well, and is thus suitable for self-localization of service robots in large-scale environments. The evaluation of RouteLoc is done with a pure metric approach as reference method. It compares scan-matching results of laser range finder data with the position estimates of RouteLoc on a metric basis.

1 Introduction

1.1 Motivation

Future generations of service robots are going to be *mobile* in the first place. Both, in classical application areas such as the cleaning of large buildings or property surveillance, but especially in the context of rehabilitation robots, such as intelligent wheelchairs, mobility will be a major characteristic of these devices. After having shown that it is technically feasible to build these robots, additional requirements will become more and more important. Examples of such demands are the operability in common and unchanged environments, adaptability to user needs, and low material costs. To satisfy these requirements, methods have to be developed that solve the fundamental problems of service robot navigation accordingly. Apart from planning, the primary component for successful navigation is self-localization: a robot has to know where it is before it can plan a path to its goal.

Pursuing these considerations, a new self-localization approach was developed for the rehabilitation robot "Rolland" (see Fig. 1a and [12, 20]) within the framework of the project *Bremen Autonomous Wheelchair*. The algorithm is called RouteLoc and requires only minimal sensor equipment (odometry and two sonar sensors), works in unchanged environments and provides a sufficient precision for a robust navigation in large building complexes and outdoor scenarios.

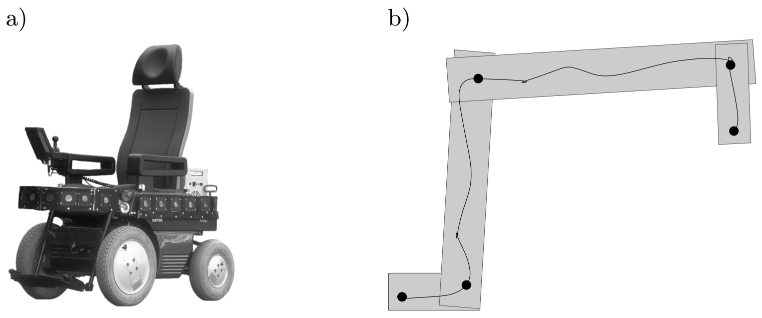

Fig. 1. a) Bremen Autonomous Wheelchair "Rolland". b) Route generalization [17].

1.2 The Bremen Autonomous Wheelchair

The Bremen Autonomous Wheelchair "Rolland" (cf. Fig. 1a) is based on the commercial power wheelchair Genius 1.522 manufactured by the German company Meyra. The wheelchair is a non-holonomic vehicle that is driven by its front axle and steered by its rear axle. The human operator controls the system with a joystick. The wheelchair is equipped with a standard PC (Pentium III 600MHz, 128 MB RAM) for control and user-wheelchair interaction tasks, 27 sonar sensors, and a laser range sensor behind the seat. The SICK laser range finder has an opening angle of 180° toward the backside of the wheelchair and is able to deliver 361 distance measurements every 30 ms. The original Meyra wheelchair already provides two serial ports that allow to set target values for the speed and the steering angle as well as determining their actual values. Data acquired via this interface is used for dead reckoning. The odometry system based on these measurements is not very precise, i.e. it performs well in reckoning distances but it is weak in tracking angular changes. A modular hardware and software architecture based on the real-time operating system QNX allows for the adaptation to an individual user [21]. At the moment, the two main applications already implemented are the *Driving Assistant* and the *Route Assistant* [12].

2 Modeling Locomotion and Environment

Self-localization of robots is usually done by matching the robot's *situation*, i. e. the current (and maybe also the past) sensor impressions and its locomotion, with a representation of its environment, e. g. a map. For a successful matching, it is indispensable that the models for both, the robot's situation and the environment, are comparable. The following two sections present the situation model and the environment model chosen for RouteLoc.

2.1 Situation Model

Röfer [17] introduces an incremental generalization of traveled tracks. The idea is to generalize the locomotion of the traveling robot during runtime to an abstract route description. Such a description represents the route as a sequence of straight segments that intersect under certain angles. Since natural minor deviations occurring while traveling are abstracted away this way, the generalized description of the robot's route from its starting point to its current location is an adequate situation model.

Specifiying abstract route descriptions. Fig. 1b shows the locomotion of the robot as recorded by its odometry system as a solid curved line. The corners recognized by the generalization algorithm are depicted as circles. The rectangular boxes represent the so-called acceptance areas: As long as the robot remains within such a region, it is assumed that the robot is still located in the same corridor. The width of the rectangular boxes is determined with the help of a histogram-based approach from the measurements of two sonar sensors mounted on the wheelchair's left- and right-hand side chassis [17]. Note there may be other generalization algorithms that do not rely on external sensor input.

As a result, the generalization R of the route traveled so far is defined as a sequence of *corners* as follows:

$$R = \langle c_i \rangle, \text{ where } c_i = (\rho_i, l_i),\ i \in \{0, \ldots, n\} \tag{1}$$

In contrast to the concept "corner$_c$" proposed by Eschenbach *et al.* [6], the length of the incoming segment of a corner is not considered here. In (1), ρ_i is the rotation angle between the incoming and the outgoing segment of a corner in a "local frame of reference", i. e., ρ_i describes the relative change in orientation when passing corner c_i. As an example, consider the almost rectangular corner c_1 in the lower left part of Fig. 1b (c_0 is the "virtual" starting corner): ρ_1 is about 86°, because the robot has to turn about 86° to the left when changing corridors at corner c_1. Note that ρ_0 is a "don't care" value, i. e. only the outgoing segment of the first corner is considered, whereas the angle is ignored. The second parameter of a corner as specified in (1) is the length l_i of the outgoing segment.

Incremental generalization of route descriptions. Since the situation of the robot has to be known while it travels, the route generalization must be carried out incrementally and in real-time. Röfer's approach satisfies both requirements. Nevertheless, the incremental generalization has the drawback that it has to partially rely on uncertain knowledge: the distance l_n already traveled in the so far final segment as well as the angle ρ_n to the previous segment may change during runtime depending on the locomotion of the robot. The information about c_n is volatile and not fixed before a new final corner c_{n+1} is detected. This is illustrated in Fig. 2: The upper row of the figure shows three different snapshots of a single trajectory driven by the robot. The respective current location of the robot is indicated by the arrow. Even though this is only a sketch, it is reasonable to expect a similar odometry recording when the robot travels in a straight corridor, turns right after some time and turns left some time later. In the lower row, the corresponding generalizations are shown: In Fig. 2a, no corner

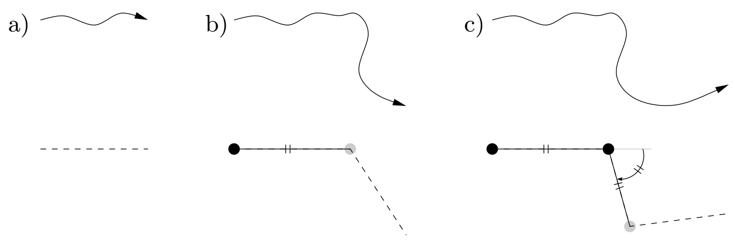

Fig. 2. Fixing of the penultimate corner during the incremental generalization.

has been detected so far, the traveled path completely fits into the imaginary corridor defined by the acceptance area of the segment depicted as a dashed line. In Fig. 2b, the robot has conducted a right turn and seems already to perform a new turn to the left. Nevertheless, it is only then that the robot leaves the acceptance area of the first segment. As a result, the generalization algorithm sets up a new—so far final—corner (indicated by the grey circle) and a new—also so far final—segment (indicated by the dashed line). Simultaneously, the parameters of the first corner c_0 (marked by the black circle) are fixed. Since it is the first corner, the angle is irrelevant; but the length of the outgoing segment is known now. In Fig. 2c, the robot has moved further and has left the acceptance area of the second route segment, resulting in the generation of another new segment. The generalization algorithm positions the third corner and fixes the parameters of c_1: The rotation angle from the first to the second segment and the distance between c_1 and c_2.

The abstraction resulting from this generalization method turns out to be very robust with regard to temporary obstacles and minor changes in the environment. Nevertheless, it is only helpful, if the routes are driven in a network of corridors or the like. Fortunately, almost all larger buildings such as hospitals, administration or office buildings consist of a network of hallways. In such environments, the presented algorithm works robustly.

2.2 Environment Model

In order to localize a robot within a representation of the environment such as a map, the model used for describing the current situation of the robot must be compatible to the model used for the description of the robot's environment, and it should be appropriate for the intended application scenario of the robot. Developing service robots, especially rehabilitation robots, usually means to develop low cost devices. Therefore, the equipment used for self-localization should be as sparse as possible. Nevertheless, mobile service robots such as cleaning robots, surveillance robots, and smart wheelchairs often have to cover a large operation space. That means that the self-localization approach must be able to work in large-scale environments such as complex buildings, university campuses or hospital areas. Especially in the context of rehabilitation robots the environment cannot easily be changed, e. g., by mounting artificial landmarks or beacons at

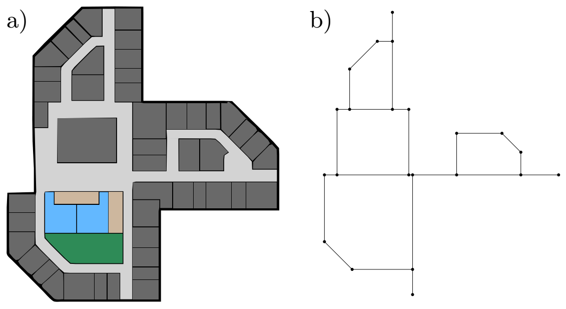

Fig. 3. a) Sketch of a floor. b) Corresponding route graph.

decision points, because they are often part of public buildings. Furthermore, environment changes are very expensive. As a consequence, an approach is needed that requires only minimal sensor equipment, works in unchanged environments, that is able to operate reliably in large-scale scenarios.

Taking into account these aspects, a topological map that is enhanced with certain metric information appears to be an adequate representation of the environment in this context. Adapted from [29], such an environment model will be referred to as *route graph*. In the following, the nodes of a route graph correspond to decision points in the real world (or *places* as called by [29]): hallway corners, junctions or crossings. The edges of a route graph represent straight corridors that connect the decision points. In addition to the topological information, the route graph contains (geo-)metric data about the length of the corridors as well as about the rotation angles between the corridors. For example, Fig. 3a shows a sketch of the second floor of the MZH building of the Universität Bremen. The corresponding route graph is depicted in Fig. 3b. It consists of 22 nodes (decision points) and 25 edges (corridors) connecting them.

Since the route graph (environment model) has to be matched with route generalizations (situation model), it is advantageous *not* to implement the graph as a set of nodes that are connected by the edges, but as a set of so-called *junctions*:

Definition 1 (Junction). *A junction j is a 5-tuple*

$$j := (H, T, \gamma, o, I)$$

where H ("home" of the junction j) and T ("target" of j) are graph nodes that are connected by a straight corridor of length o. The set I consists of all incoming junctions j'_i that lead to j, i. e., $I = \{(H', H, \gamma', o', I')\}$. The function incomings(j) selects the incoming junctions of j, i. e., $incomings(j) = I$. The signed angle γ is the rotation angle between the prolongation of an outgoing segment of some j'_i to the outgoing segment of junction j, i. e. it denotes by how many degrees it has to be turned to travel through j. For left turns, γ is positive; for right turns, γ is negative; $\gamma = 0$ means that j is a so-called "straight junction", e. g. the T-bar of a T-junction (cf. Fig. 4).

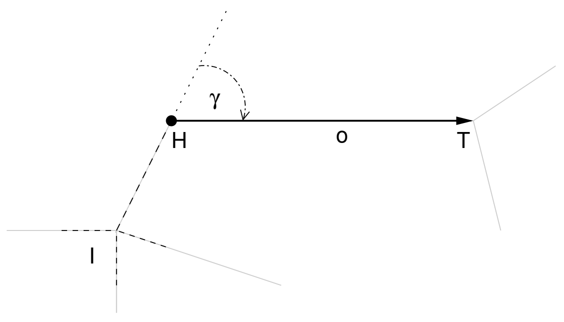

Fig. 4. Junction in a part of the route graph.

Note that outgoing segments of junctions are directed, i. e. junctions are one-way connections between route graph nodes. As shown in Sect. 3.1, the corners of a route generalization are compatible with the junctions of the route graph in that they can be matched and assigned with a real number representing a similarity measure.

Based on definition 1, a route graph G is the set of all junctions:

Definition 2 (Route Graph). *A route graph G is a set of all junctions that are connected:*

$$G = \{j = (H, T, \gamma, o, I) | \exists j' \in G : j' \neq j \wedge j' \in \mathit{incomings}(j)\}$$

While the representation of the environment as a route graph is formally similar to Voronoï diagrams as recently used, e.g., by Thrun [24], Zwynsvoorde *et al.* [30, 31], and Choset [3], the localization approach presented here is not only applicable in sensory-rich (indoor) environments but also in pure outdoor or hybrid scenarios such as the campus example presented below. This is because the generalization of the robot's locomotion is used as reference information for the localization. Thus, RouteLoc does not have to rely on input from proximity sensors as it is necessary for the Voronoï diagram based approaches (a Voronoï diagram is defined on the basis of the sensor-perceived distance of the robot to objects in its environment).

In contrast to metric (grid-based) representations, the route graph is much easier to handle with respect to the required amount of computing time and memory. For example, the campus environment used for experiments in the results section (see Sect. 6 and Fig. 12a) is coded as a list of only 144 junctions (see Fig. 13b). The complexity of RouteLoc is linear in the number of junctions in the route graph. Therefore, it is important to note that covering a larger area with the route graph does not necessarily mean an increase in junctions. Instead, the critical question is, how many decision points are there in the environment.

3 RouteLoc: an Overview

This section is meant to explain how generalized route descriptions as situation model and a route graph as environment model are used for the absolute self-localization of a mobile robot in large-scale environments. First, a sketch of RouteLoc is presented to explain the basics of the algorithm. The simplifying assumptions made here for clarity purposes are then dropped in the detailed description of the algorithm in Sect. 4.

The basic idea of the self-localization approach is to match the incremental generalization of the currently traveled route with the route graph. This matching process provides a hypothesis about the robot's current position in its environment: RouteLoc ongoingly determines the hallway (represented by an edge in the route graph), in which the robot is most likely located at that very moment in time. Since the distance already traveled in the hallway is also known, an additional offset can be derived. As a result, the position of the robot within the hallway is found precise enough for most global navigation tasks. The precision is limited by about half of the width of the corridor the robot is located in, as is shown in Sect. 3.3.

3.1 Matching Route and Route Graph

Due to the dualism between a junction in the route graph and a corner in the generalized route, the chosen situation model and the environment model are compatible. Thus, self-localizing a robot by matching a generalized route with a route graph should in principle be straightforward. Nevertheless, there are some pitfalls that have to be paid attention for.

Since the algorithm has to deal with real data, there are almost no "perfect matches". That means, that even if the robot turned by exactly 90° at a crossing, the angle of this corner as calculated by the route generalization will almost certainly differ from 90°. This is mainly due to odometry errors. On the other hand, two corridors that meet in a perfect right angle in the route graph may well include an angle of only 89.75° in reality. These uncertainties have to be coped with adequately.

A second topic worth considering is the complexity of the matching process: At least in theory, a route can consist of arbitrarily many corners. Therefore, matching the whole generalized route with the route graph in each computation step is not feasible, because—at least in theory—this would require an arbitrarily long period of computing time. A solution to this problem is presented in the following subsections.

Within this section, it is assumed that every corner existing in reality is detected by the generalization algorithm *and* that every corner detected by the generalization algorithm is existing in reality. As mentioned earlier, this assumption is simplistic and unrealistic. Nevertheless, it is reasonable here in order to simplify the explanation of the basic structure of RouteLoc. The details of the algorithm are thoroughly discussed in Sect. 4.1.

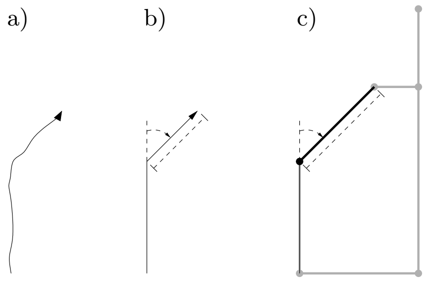

Fig. 5. Direct match of route corner and route graph junction. a) Odometry recorded. b) Corresponding route generalization. c) Matching junction in the route graph.

Direct Match of Route Corner and Graph Junction. If there are only two corners in the route, i. e. $R = \langle c_0, c_1 \rangle$ (the "don't care" corner c_0 and the first "real" corner c_1), a *direct match* of c_1 and some junction j in the route graph is possible (cf. Fig. 5). As mentioned above, a binary decision of whether or not c_1 and j match is not adequate in this situation. Thus, a probabilistic similarity measure is introduced that describes the degree of similarity between the route corner and the junction as a real number between 0 and 1. For the route $R = \langle c_0, c_1 \rangle$ this value represents the probability that the robot is located in j.

The similarity measure m_d for the direct match of a route corner c with a route graph junction j is defined as

$$m_d(c, j) = s_l(c, j) \cdot s_\alpha(c, j) \tag{2}$$

In (2), the similarity s_l of the lengths of the outgoing segment of j and of the route segment of c, is defined as $s_l(c, j)$ with

$$s_l(c, j) = sig\left(1 - \frac{|l_c - d_j|}{d_j}\right) \tag{3}$$

In (3), l_c is the length of the outgoing route segment of c; d_j is the length of the outgoing corridor of junction j. The longer the corridor, the larger the deviation may be for a constant similarity measure.

The similarity of the corresponding rotation angles, $s_\alpha(c, j)$, is defined as

$$s_\alpha(c, j) = sig\left(1 - \frac{||\gamma_j - \rho_c||}{\pi}\right) \tag{4}$$

In (4), γ_j is the rotation angle between the two segments of junction j, and ρ_c is the rotation angle of the final route corner c. Note that the result of this subtraction is always shifted into the interval $[0, \ldots, \pi]$, as indicated by the $|| \ldots ||$ notation. Please also note that these equations will be refined in the following in order to cover some special cases that will be introduced below.

In (3) and (4), the sigmoid function *sig* is used to map the deviations in length and in rotation angle into the intended range. The idea is to tolerate small deviations with respect to the corridors' length or the angles, respectively, whereas large deviations lead to only small similarity values.

If the route R only comprises one corner (the "don't care corner"), i. e., $R = \langle c_0 \rangle$, the angle is ignored, because it is the initial rotation angle that has no meaning (cf. Sect. 2.1), thus $s_\alpha(c_0, j) = 1$. Therefore, the only remaining criterion for a direct match are the segments' lengths, thus $m_d(c, j) = s_l(c, j)$ in this case.

Induction Step. After having defined the direct matching for single-corner routes, the similarity measure has to be extended to longer routes. When a route $R = \langle c_0, \ldots, c_n \rangle$ with $n > 1$ is to be matched with a junction j, it has to be found out, whether there is a direct match between corner c_n and junction j, *and* whether there is one between c_{n-1} and some j' with $j' \in \text{incomings}(j)$, *and* whether there is one between c_{n-2} and some j'' with $j'' \in \text{incomings}(j')$, *and* so on. If so, a sequence of junctions of the route graph is found the whole route R can be matched with.

Thus, the *matching quality* of a complete route R with respect to a specific route graph junction j is defined as follows:

Definition 3 (Matching Quality). *Given a route $R = \langle c_0, \ldots, c_n \rangle$ with $n \geq 0$ and a junction j of the route graph G ($j \in G$), the matching quality $m(R, j)$ of R with respect to j is defined as*

$$m(R, j) = \max \left\{ \prod_{i=0}^{n} m_d(c_i, j_i) \;\middle|\; \exists \langle j_0, \ldots, j_n \rangle : j = j_n \wedge j_{k-1} \in \text{incomings}(j_k) \right\} \tag{5}$$

The definition states that every possible sequence of length $n + 1$ of route graph junctions is considered that fulfills two requirements: the final junction of the sequence must be j and the sequence must be "traversable", i. e., the k-th junction in the sequence must be an incoming junction of the $(k + 1)$-st junction of the sequence. As such a sequence consists of as many junctions as there are route corners, the matching quality can be determined by calculating the product of the direct matching qualities of the sequence junctions and the corresponding route corners. The overall matching quality of the route R and junction j is the maximum of all these products.

The number of such sequences grows exponentially with the length of the route and with the number of junctions in the route graph. Therefore, defining equation (5) is inadequate for a real-time capable localization approach. Fortunately, there is a workaround that dramatically reduces the complexity of calculating the matching quality: following the idea of the incremental route generalization, the matching quality can be defined inductively. In order to determine $m(\langle c_0, \ldots, c_n \rangle, j)$, it is sufficient to know $m_d(c_n, j)$ and $m(\langle c_0, \ldots, c_{n-1} \rangle, j')$

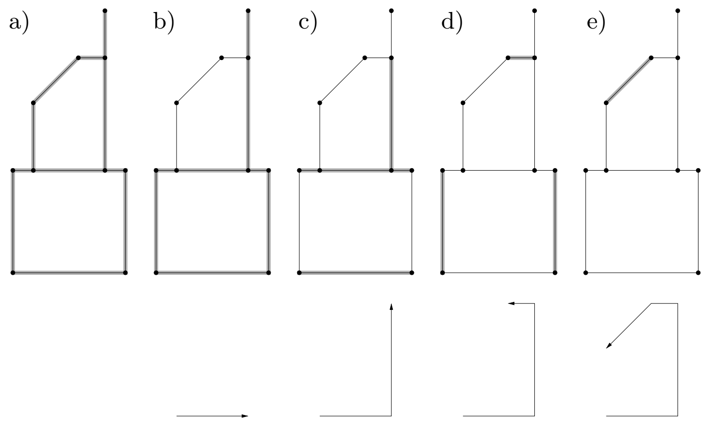

Fig. 6. Matching of route generalization and route graph.

with $j' \in \text{incomings}(j)$. As a result, the defining equation (5) can be refined to

$$\begin{aligned} m(\langle c_0\rangle, j) &= m_d(c_0, j) & , n = 0 \\ m(\langle c_0, \ldots, c_n\rangle, j) &= m_d(c_n, j) \cdot \max_{j' \in \text{incomings}(j)} m(\langle c_0, \ldots, c_{n-1}\rangle, j') & , n > 0 \end{aligned} \tag{6}$$

Calculating this recursion in every step is still impractical because it depends on the length of the route. Fortunately, the recursive function call can be avoided, if each junction is assigned with the probability value for having been in one of its incoming junctions before.

By applying definition 3 to the current route and every route graph junction, the junctions are assigned with a matching quality. The maximum of all the matching qualities provides a hypothesis which junction most likely hosts the robot. This junction is called *candidate junction* j_c for a route R.

$$j_c(R) = \text{argmax}_{j \in G}\{m(R, j)\} \tag{7}$$

Figure 6 presents a step-by-step visualization of the localization process: In the initial situation, no information about the robot's potential location is available. Therefore, every junction in the graph can host the robot with the same likelihood. This is indicated by the edges underlined in grey in the route graph that is shown in the upper row of the figure. After the robot traveled some distance in a corridor (cf. Fig. 6b), three edges in the graph are identified in which the robot cannot be located. The route segment just traveled is longer than the corridors represented by these edges. After completing the first turn (90° to the left, see Fig. 6c), basically only three possibilities remain: Either the robot started in a corridor that is represented by one of the two facing edges depicted vertically in the lower part of the route graph or it started horizontally and its location is in the upper part of the graph afterwards. In Fig. 6d, another left turn yields no new information and thus no reduction of the possible robot locations. As

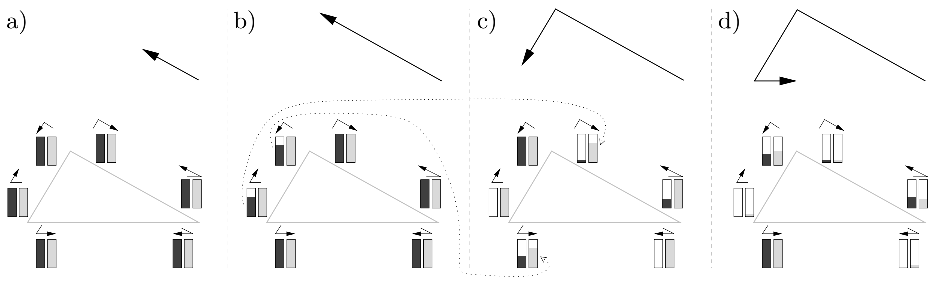

Fig. 7. Propagation of probabilities.

shown in Fig. 6e, the situation clarifies after the following turn: the location of the robot is determined by figuring out a unique candidate junction.

3.2 Propagation

In the previous subsection, it has been motivated that it is necessary to store the probability that the robot was in the incoming segment of a junction before detecting the last corner. But this information has to be transferred to other junctions when a corner in the route is detected by the generalization algorithm. After each corner detection, each junction is assigned with the maximum of the matching qualities of its incoming junctions, as discussed in Sect. 3.1:

$$m_{in}(j) = \max_{j' \in incomings(j)} m(\langle c_0, \ldots, c_{n-1} \rangle, j') \tag{8}$$

Figure 7 shows four snapshots of a route traveled in a triangular environment. The upper part of each column shows the generalized trajectory as recorded by the robot. The arrow indicates the current position. The lower part of each snapshot depicts a route graph that consists of six junctions. Each junction is assigned with two probability values, depicted as partly filled columns. The left column (dark grey filled) indicates the direct matching quality of the final route corner with this junction. The right column (light grey filled) describes the probability of having been in the incoming segment of this junction before. A completely filled column stands for a 100% match, completely empty means something below 10% (but more than 0%). The arrows above the probability columns indicate the junction, e. g. the columns in the lower left corner of the route graph belong to the junction that leads from the left corridor to the lower corridor with a rotation angle of about 120°. From figure 7b to 7c, dotted arrows indicate the propagation.

3.3 Estimating the Robot's Position

Knowing the candidate junction and the offset already traveled in its outgoing segment enables RouteLoc to estimate a metric position of the form "The position

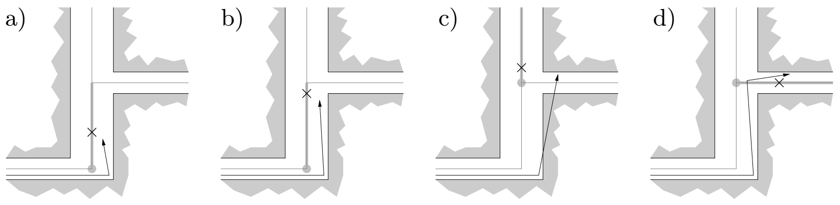

Fig. 8. "Generalization delay" when turning from one corridor to another.

is x cm in the corridor that leads from decision point A to decision point B." One could argue that this metric information is superfluous for the user or for higher level navigation modules, because the corridors *between* the decision points are by nature free from decisions such as turning to a neighboring corridor. Thus, no detailed information about the robot's location between the decision points should be required. Nevertheless, the metric information is indispensable for two reasons: First, not every location that is important for the robot's task can be modeled as a decision point. Consider, e. g., some cupboard a wheelchair driver has to visit in a corridor. Second, when traveling autonomously, the robot often has to start actions or local maneuvers in time, i. e. they have to be initiated at a certain place in the corridor, maybe well before the relevant decision point can be perceived by the robot. This would be impossible without the metric information.

The rest of this section discusses some aspects that are relevant for a successful position estimate.

Ambiguous environment. In some situations, the structure of the environment could turn out to be inadequate to this route-localization approach in its current version. In a square environment, for instance, the algorithm will fail, because every junction remains equally likely to host the robot even if the robot moves through the corridors. This problem of perceiving different places as if they were the same is commonly referred to as *perceptual aliasing*. When traveling in the square environment, four position estimates that are equally likely would be favored, no decision for a specific corridor would be possible.

Similarly, in a straight corridor, the algorithm is almost lost, because it has no means to infer where the robot started. Nevertheless, the longer the robot moves along the corridor, the less estimates are valid, simply due to the length of the trajectory already traveled. But even, if the robot traveled a straight route segment of about the corridor's length, the algorithm still would generate two hypotheses about the robot's position, one at each end of the corridor.

"Generalization delay" when changing corridors. Due to the nature of the generalization algorithm, there exists a certain delay before the change of corridors can be detected. For example, in Fig. 8a, the generalization (depicted as a thin black line; the arrow indicates the current position) of the traveled route is correctly matched with the route graph. The highlighted junction is the candidate junction, resulting in a position estimate which is indicated by the cross. The estimated position differs only slightly from the real position (cf. the paragraph on precision below). In Fig. 8b, the robot almost reached the T-junction. The localization is still correct. In Fig. 8c, the robot already changed corridors by taking the junction to the right. But the generalization algorithm has not yet been able to detect this, because it still can construct an "acceptance area" for the current robot position within the same corridor as before. Therefore, it assumes that the robot passed the T-junction and estimates the robot's position to be in the junction that forms a straight prolongation to the former one. It is not until the robot has traveled some more distance before the generalization algorithm detects the corner (see Fig. 8d). Then, the position estimate is immediately corrected and a precise hypothesis is set up.

Precision of the position estimate. Because of the modeling of the environment and the robot's locomotion, the algorithm is rather insensitive to odometry errors (see Fig. 12b). The offsets normally represent only short distances that result from accumulating straight movements, and almost no rotational motion which often causes dead reckoning errors. Nevertheless, the precision of the algorithm is limited to half the width of the current corridor at right angles to the robot's driving direction and half the width of the previous corridor in the robot's driving direction (see Fig. 9). The error could be even bigger, if the route graph is not correctly embedded in the center of the corridors, as it should be. Note that errors do not accumulate across junctions, but within longer junctions odometry errors may become significant.

The precision does explicitly *not* depend on the length of the traveled route, as every matching of a route corner to a graph junction once again limits the error. Nevertheless, the quality of the position estimate depends on the "quality" of the environment. The results of the experiments presented in Sect. 6 confirm this point of view.

4 Inside RouteLoc: a Deeper Insight

Section 3 uses the unrealistic assumption that the route generalization algorithm creates a new corner for every decision point (junction) the robot passes, and—vice versa—that every generated corner has its counterpart in the route graph and in the real world. This is too optimistic, as is shown below. Section 4.1 copes with this problem and presents a general solution that requires no restrictive assumptions.

Right at the beginning of a robot journey, a few special cases have to be paid attention to: If the robot did not start its travel at a decision point but within a corridor, the standard matching process as described above does not work as fast as it could. Furthermore, a route with no real corner detected so far requires some special attention during the matching process. This is discussed in Sect. 4.2.

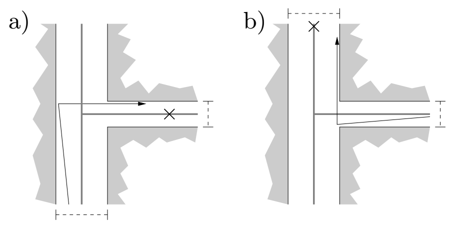

Fig. 9. Precision of the position estimate. a) Entering a narrow corridor from a wide one. b) Vice versa.

Another assumption made in Sect. 3 is that the robot can only change its general driving direction at decision points. This is a straightforward inference from the definition of decision points (junctions) and corridors connecting these decision points. But, there is a decision the robot can make anywhere, not only at decision points: turning around. Since the route graph junctions are directed, such a turning maneuver implies that the robot leaves the current junction. But unfortunately, it does not end in another junction represented in the route graph, because such turning junctions are not available in the route graph. Section 4.3 describes the handling of turning around within corridors.

4.1 On Phantom Corners and Missed Junctions

While the robot travels, the self-localization algorithm is expected to ongoingly present a hypothesis about the robot's current position. This hypothesis is to be updated in regular intervals. In the experiments presented in the results section 6, an update interval of 20cm travel distance has been used. In every update step, the route generalization algorithm checks whether a new corridor has been entered and updates the route description accordingly. Afterwards, the matching process is carried out which leads to a position estimate, as discussed in Sect. 3.3.

In every update step, four different situations can occur with respect to detected or undetected route corners, and to existing or not existing junctions in the route graph:

1. There is no junction in reality and the generalization algorithm correctly detects no route corner (see Fig. 10a). This is the normal case because most of the time the robot moves through corridors.
2. There is a junction in reality and the generalization algorithm correctly detects a corresponding route corner (see Fig. 10b). This was the assumption in the previous section.
3. There is no junction in reality even though the generalization algorithm detects a route corner, a so-called *phantom corner* (see Fig. 10c). Unfortunately, this case is not that rare, because due to odometry drift, long corridors are often generalized to more than one segment.
4. There is a junction in reality but the route generalization algorithm does not detect a corresponding route corner (see Fig. 10d). This is the problem

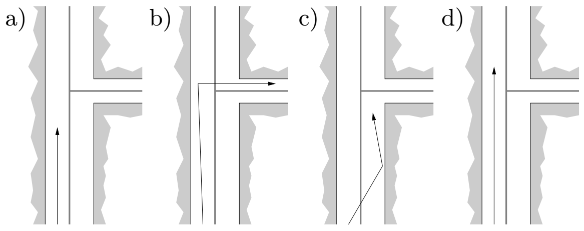

Fig. 10. Special cases handled by RouteLoc.

of *missed junctions* which is not a flaw of the route generalization algorithm but a result of the spartan sensor use of the approach. Nevertheless, the self-localization algorithm is able to handle it.

The correct handling of these four situations is fundamental for the algorithm. They are discussed in the following sections.

There is *no* Junction and *no* Corner is Detected. In Fig. 10a, the standard situation is illustrated: the robot moves within a corridor, no junction in its surroundings, and the route generalization algorithm correctly infers that the robot did not change corridors, but still travels in the same corridor as one step before. In this case, the matching process can be carried out as described in Sect. 3. There is only one restriction: the definition of the similarity measure in (3) assumes that the final length of the route segment to be matched with the junction's outgoing segment is already known. As mentioned above, this is not the case for the currently final segment of the route traveled so far. Therefore, the calculation of the similarity measure $s_l(c, j)$ for the lengths of the final route corner c and a junction j has to be changed in this case to

$$s_l(c,j) = \begin{cases} 1\,, & l_c \leq d_j \wedge c = c_n \\ sig\left(1 - \frac{l_c - d_j}{d_j}\right), & \text{otherwise} \end{cases} \tag{9}$$

In (9), l_c is the length of the route segment of corner c; d_j is the length of the outgoing corridor of junction j. In contrast to the original definition in (3), the similarity is set to 100% not only if the lengths are equal, but also if the final route segment is shorter than the junction segment. This is no surprise, as it is a preliminary match and the currently available information about the final route segment indicates that it matches the route graph junction. Only if l_c happens to be larger than d_j, the similarity measure drops below 100%. Note that (9) replaces (3) as definition of the similarity measure with respect to the segments' lengths.

As long as no corner is detected, there is no need for propagating the probabilities to adjacent junctions. Thus, the similarity values for each junction are only adapted to the current route generalization. Nevertheless, the case of missed junctions has to be kept in mind (see below).

There is a Junction and a Corner is Detected. In some situations, the route generalization algorithm detects corners in the route, as shown in Fig. 10b. If there exists a corresponding junction in the route graph, the matching as described in Sect. 3 will be successful. Note that detecting a new corner in the route fixes the then penultimate corner in its angle and length components. Therefore, the matching is a three-step process in this case: first, the new penultimate corner is matched according to the rules described in Sect. 3.1 and the similarity measure just defined in (9). Second, the probabilities are propagated to the adjacent junctions as discussed in Sect. 3.2. And third, the new final corner is matched as a non-fixed corner according to (9).

There is *no* Junction, but a Corner is Detected. Unfortunately, this case is not as rare as one could expect. As depicted in Fig. 10c, the motion track as recorded by the robot's odometry can significantly deviate from a straight line even if the robot drives in a straight corridor. Especially in very long corridors, the odometry tends to be inaccurate. As an example, consider Fig. 12b that depicts the generalized motion track that has been recorded during experiments on the campus of the Universität Bremen. In the upper left part of the figure, the main boulevard of the campus, which is straight and about 300m long, is partitioned into several segments. This is because the odometry recorded the straight boulevard as a crescent-shaped curve. The erroneously detected "phantom corners" between the segments are a problem for the self-localization algorithm because the probability values have to be propagated through the graph after every route corner detection (see the section on propagation 3.2). If, however, such a detected route corner is a phantom corner, the propagation will be an error.

Therefore, when detecting a corner, the self-localization algorithm has to decide whether it is a corner with a corresponding junction in the route graph, or whether it is a "phantom corner" that results from bad odometry data. As if this were not enough, this decision cannot be made until the information about the route corner is fix. That means, the decision of whether or not a corner is believed to be either real or phantom can only be made with respect to the penultimate, already fixed, corner in the generalized route.

These considerations suggest to pursue two instead of one hypotheses for each junction (see Fig. 11): The first describes how probable it is that the robot is in the outgoing segment of the junction and has been in the incoming segment before the final corner has been detected (i. e., the final route corner is *real*; see Fig. 11c). The second hypothesis describes the probability that the robot is in the outgoing segment of the junction and has already been there before the final corner has been detected (i. e., the final corner is *phantom*; see Fig. 11d).

As a result, two similarity measures for the two hypotheses have to be defined: The similarity measure that assumes the final route corner to be a *real* corner is identical to m_d as defined in (2). It is renamed to m_d^r here. The similarity measure that assumes the final route corner to be a *phantom* corner is called

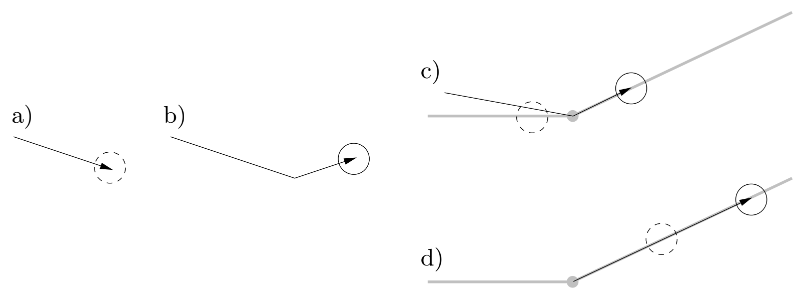

Fig. 11. *Real* and *Phantom* route corners. a) Generalized route before detection of the corner. b) After detection. c) *Real* corner. d) *Phantom* corner.

m_d^p. It uses (9) as measure for the similarity of the segments' lengths, but a different definition s_α^p of the rotation angle similarity:

$$s_\alpha^p(c,j) = sig\left(1 - \frac{\rho_c}{\pi}\right) \tag{10}$$

In (10), the rotation angle ρ_c of the route corner is compared to 0°, instead of to the junction angle as in (4). As a result, the matching probability is close to 100% for very small angles (i. e., detected route corners with a small angle are likely to be phantom corners) and low for significant angles (i. e., detected route corners with an angle of, say, 90° are expected to be real corners with high probability).

The two hypotheses are always considered in parallel, i. e., there are two probabilities for a junction to host the robot: One of them assumes that the final route corner is a real corner, which means that the robot has been in the incoming segment of the junction before the corner has been detected. The other one assumes that the final corner is a phantom corner, which means that the robot has already been in the outgoing segment of the junction before the corner has been detected. As a result, there also exist two matching qualities $m^r(R,j)$ (assuming final corner of R is real) and $m^p(R,j)$ (assuming the final corner of R to be phantom).

When a new final corner is detected in the route, the propagation process copies the superior hypothesis to the adjacent junction. At that time, a decision can be made about whether the real or the phantom probability is the "correct" one, because the corner is fixed in length and rotation angle.

The overall probability of the junction (i. e. the matching quality) is then calculated as the maximum of both hypotheses:

$$m(R,j) = \max\{m^r(R,j), m^p(R,j)\} \tag{11}$$

There is a Junction, but *no* Corner is Detected. It is possible that a corner existing in reality has been passed and has not (yet) been detected by the generalization algorithm. As a consequence, the resulting change of corridors is not recognized (*missed junction*). Usually, this cannot be blamed on the generalization but on the fact that—based only on the locomotion data—one cannot distinguish traveling in a straight corridor with no junctions or crossings from traveling in a straight corridor passing several T-junctions. Therefore, the self-localization algorithm has to solve this problem. In every step, it is checked, whether the outgoing segment of the final route corner c_n is longer than the outgoing segment of the currently considered route graph junction j. If so, it is likely that this route segment is an overlap from a previous junction that leads to j. Note that not only straight predecessors of j (i. e. those that form a $0°$ angle with j) have to be considered here. Every incoming segment of j could have "hosted" the initial part of the route segment of corner c_n. Especially in long corridors with lots of crossings, it often happens that these overlaps stretch over more than one junction.

Due to these considerations, it is always calculated how far the final route segment extends into the outgoing segment of the currently considered junction. This may significantly differ from the length of the final route segment. That is why it is a simplification to use the length l_c of the route segment in (3). Instead, in all equations for the similarity measure ((3), (9)), the distance l_c^+ already traveled in the segment has to be used instead of the length of the so far final route segment l_c (cf. Sect. 4.4).

4.2 Initial Phase Specialities

After solving the phantom corner and missed junction problems in Sect. 4.1, there are two special cases with respect to the early phases of a robot journey that are to be covered by the algorithm, but have not been addressed yet:

- Matching a route $R = \langle c_o \rangle$ that comprises only the initial corner with the route graph
- Starting the robot's journey not at a decision point but somewhere in the middle of a corridor.

These two topics are discussed in the following two paragraphs.

Before the first corner was detected. As discussed in Sect. 2.1, the rotation angle of the initial route corner c_0 is special in that it is a "don't care" value. Even stronger, it may never be used during the matching process, because it has no meaning: it describes the rotation angle between the first route segment and an imaginary but not existing "zeroth" route segment. Therefore, the matching process has to be carried out slightly different as long as no real route corner has been detected. The implementation of this requirement is straightforwardly achieved by a further extension to the similarity measure calculation previously shown in (3) and refined in (9). The equation that includes the "before the first corner" case looks as follows for the assumption that c_n is a real corner:

$$s_\alpha^r(c,j) = \begin{cases} 1\,, & c = c_0 \\ sig\left(1 - \frac{||\gamma_j - \rho_c||}{\pi}\right), & \text{otherwise} \end{cases} \tag{12}$$

and for the assumption that c_n is phantom:

$$s_\alpha^p(c,j) = \begin{cases} 1\,, & c = c_0 \\ sig\left(1 - \frac{\rho_c}{\pi}\right), & \text{otherwise} \end{cases} \tag{13}$$

where c_0 is the initial corner of the route.

Starting in the middle of a corridor. The basic idea of the whole approach is that detected route corners can be identified with certain junctions in the route graph. Then, the similarity measures deliver an adequate means to decide about the matching quality. However, at the very beginning of a robot journey, a situation may occur, where the robot does not start at a place in the real world that is represented by a route graph node. Instead, the starting position could be located somewhere in a corridor in the middle between two decision points. If the robot reached the first adjacent junction, detected a corner, and matched the route with the graph, the length of the driven segment would be significantly too short in comparison with the junction's outgoing segment (because the robot started in the middle). Nevertheless, the route segment perfectly fits into the route graph. Thus, for the first route segment, it must be allowed that it is shorter than the junction's outgoing segment without loss of matching quality. Once again, the equations for the similarity measures are refined to:

$$s_l^r(c,j) = \begin{cases} 1\,, & l_c^+ \leq d_j \wedge c \in \{c_0, c_n\} \\ sig\left(1 - \frac{l_c^+ - d_j}{d_j}\right), & \text{otherwise} \end{cases} \tag{14}$$

$$s_l^p(c,j) = \begin{cases} 1\,, & l_c^+ \leq d_j \wedge c \in \{c_1, c_n\} \\ sig\left(1 - \frac{l_c^+ - d_j}{d_j}\right), & \text{otherwise} \end{cases} \tag{15}$$

4.3 Turning Around Within a Corridor

Nonholonomic vehicles such as the Bremen Autonomous Wheelchair "Rolland" are not able to move in arbitrary directions but they are restricted to bias bearings such as forwards and backwards instead. As a consequence, nonholonomic robots are not able to turn on the spot without shunting. Especially for the wheelchair, there are some corridors that are too narrow to turn at all. Therefore, it is fundamental to know the orientation of the wheelchair within a corridor. This is solved by modeling the corridors as one-way junctions, where the orientation is inherently known (see Sect. 2.2 on route graphs). If the robot turns around in a corridor, it leaves its current junction. But—by definition—leaving a junction means to enter another junction. Unfortunately, there are no junctions in the route graph that connect the two directions of a corridor.

An additional problem is that a turning maneuver can be carried out at any position within the hallway. In contrast to that, leaving the corridor is only possible at junctions.

To overcome these problems in order the able to handle turns, the set of junctions that initially form the route graph G is extended by so-called *"turn-junctions"* at program start as shown:

$$G' = G \cup \left\{ \left(H, T, \pi, |\overline{HT}|, I\right) \mid H, T \in G, I \subseteq G, \forall i \in I : i = (T, H, \pi, |\overline{TH}|, I') \right\} \tag{16}$$

In (16), for each junction j_i in the initial route graph G, all turn-junctions T that can be generated for j_i are added to G. As an example, consider the route graph depicted in Fig. 13b that is used for the experiments presented in Sect. 6. The 144 junctions of this route graph require an additional set of 102 turn-junctions. The upper bound of the number of required turn-junctions for a route graph with n "real" junctions is $2n$. In typical environments, however, it often happens that two or more junctions share one turn-junction, e. g. junctions *cdh* and *kdh* in Fig. 13b both need the turn-junction *dhd*. The incoming and the outgoing segment of these turn-junctions represent the same hallway (forwards and backwards direction) and have a rotation angle of 180°. After having generated the turn-junctions at program start, they are dealt with as if they were "normal" junctions in the sequel. The only exception is that the deviation of the length is ignored when calculating the matching quality of a generalized route corner with such a turn-junction (undershooting is granted for turn-junctions).

4.4 Similarity Measures (final revision)

This section recapitulates the defining equations for the similarity measures including all special cases:

$$m_d^r(c, j) = s_l^r(c, j) \cdot s_\alpha^r(c, j) \tag{17}$$

$$m_d^p(c, j) = s_l^p(c, j) \cdot s_\alpha^p(c, j) \tag{18}$$

$$s_l^r(c, j) = \begin{cases} 1, & l_c^+ \leq d_j \wedge (c \in \{c_0, c_n\} \vee \text{isTurn}(j)) \\ sig\left(1 - \frac{l_c^+ - d_j}{d_j}\right), & \text{otherwise} \end{cases} \tag{19}$$

$$s_l^p(c, j) = \begin{cases} 1, & l_c^+ \leq d_j \wedge (c \in \{c_1, c_n\} \vee \text{isTurn}(j)) \\ sig\left(1 - \frac{l_c^+ - d_j}{d_j}\right), & \text{otherwise} \end{cases} \tag{20}$$

$$s_\alpha^r(c, j) = \begin{cases} 1, & c = c_0 \\ sig\left(1 - \frac{||\gamma_j - \rho_c||}{\pi}\right), & \text{otherwise} \end{cases} \tag{21}$$

$$s_\alpha^p(c, j) = \begin{cases} 1, & c = c_0 \\ sig\left(1 - \frac{\rho_c}{\pi}\right), & \text{otherwise} \end{cases} \tag{22}$$

5 Related Work

The following subsection gives a brief overview about mobile robot self-localization. In Sect. 5.2, RouteLoc is compared to prominent approaches and set in relation to Markov localization methods.

5.1 Self-Localization Techniques

There are two basic principles for the self-localization of mobile robots [1]: *Relative* approaches need to know at least roughly where the robot started and are subsequently able to track its locomotion. At any point in time, they know the relative movement of the robot with respect to its initial position, and can calculate the robot's current position in the environment. It has to be ensured that the localization does not lose track, because there is no way to recover from a failure for these approaches. Modern relative self-localization methods make often use of laser range finders. They determine the robot's locomotion by matching consecutive laser-scans and deriving their mutual shift. Gutmann and Nebel [8, 9] use direct correlations in their *LineMatch* algorithm, Mojaev and Zell [14] employ a grid map as "short term memory", and Röfer [18] accumulates histograms as basic data structure for the correlation process.

On the other hand, *absolute* self-localization approaches are able to find the robot in a given map without having any a-priori knowledge about its initial position. Even more difficult, they solve the "kidnapped robot problem" [5], where—during runtime—the robot is deported to a different place without being notified. From there, it has to (re-)localize itself. That means, the robot has to deliberately "unlearn" acquired knowledge.

The absolute approaches are more powerful than the relative ones and superior in terms of fault tolerance and robustness. They try to match the current situation of the robot—defined by its locomotion and the sensor impressions—with a given representation of the environment, e. g. a metric map. As this problem is intractable in general, probabilistic approaches have been proposed as a heuristics. The idea is to pose a hypothesis about the current position of the robot in a model of the world from which its location in the real world can be inferred. A distribution function that assigns a certain probability to every possible position of the robot, is adapted stepwise. The adaptation depends on the performed locomotion and the sensor impressions. Due to the lack of a closed expression for the distribution function, it has to be approximated. One appropriate model is provided by grid-based Markov-localization approaches that have been examined for some time: they either use sonar sensors [4] or laser range finders [2] to create a probability grid. As a result, a hypothesis about the current position of the robot can be inferred from that grid. Recently, so-called Monte-Carlo-localization approaches have become very popular. They use particle filters to approximate the distribution function [7, 25]. As a consequence, the complexity of the localization task is significantly reduced. Nevertheless, it is not yet known how well these approaches scale up to larger environments.

Apart from these purely metric representations of the environment, Kuipers *et al.* propose the integration of metric and topological concepts with their "spatial semantic hierarchy" [11]. The idea is pursued by Simmons and Koenig [23] and Nourbakhsh *et al.* [15] by augmenting topological maps with metric information. The resulting self-localization methods also work probabilistically on the basis of the odometry and a local model of the environment perceived with the sensors. A very recent approach by Tomatis *et al.* combines map-building and self-localization [27]. They employ a 360° laser range finder and extract features such as corners and openings which are used to navigate in a global topological map. In addition, the laser-scans are searched for line structures (walls, cupboards, etc.) which build the basic data structure for several local metric maps (one for each node of the topological map).

5.2 Comparison between RouteLoc and prominent approaches

A number of prominent self-localization algorithms use the Markov localization approach, some of them with toplogical representations of the environment [23, 15, 27], others with metric maps [2, 7, 25]. In the robotics community, it is referred to as "Markov localization" if the algorithm somehow exploits the so-called *Markov assumption* [22]. It states that the outcome of a state transition may only depend on the current state and the chosen action. The outcome does explicitly not depend on previous states or actions.

RouteLoc is no pure Markov localization: while the matching and propagation process as presented in Sect. 3 satisfies the Markov assumption, the necessary handling of the missed junctions and phantom corners violates it. Apart from the "Markov or not" question, RouteLoc differs from other localization approaches with respect to some aspects that are gathered in table 1. As reference algorithms the topological-metric approach used for the office delivery robot Xavier by Simmons and Koenig [23] and the Mixture-MCL algorithm (an improved version of the common Monte Carlo Localization approaches) by Thrun *et al.* [26] are chosen.

Updating and propagating the probabilities is of linear complexity with respect to the number of junctions representing the environment. Since the number of junctions is usually related sublinearly (or linearly at most) to the size of the environment, the approach scales very well.

6 Results

In order to evaluate the performance of an approach for the global self-localization of a mobile robot, a reliable reference is required that delivers the *correct* actual position of the robot. Then, this reference can be used to compare it with the location computed by the new approach, and thus allows assessing the performance of the new method. RouteLoc uses a mixture of a topological and a metric representation. In fact, a typical position estimate would be "the wheelchair is

Table 1. Comparison between RouteLoc and two other localization approaches

Aspect	**RouteLoc**	**Simmons & Koenig [23]**	**Thrun *et al.* [26]**
sensor input	odometry (+ 2 sonars for generalization)	odometry + sonars	odometry + camera or laser range finder
setting	campus (in-/outdoor)	indoor office environment	indoor museum
complexity	144 junctions for 46 nodes and 100 edges, depends on number of decision points	3348 Markov states for 95 nodes and 180 edges, depends on extent of environment	About 1000 samples for an indoor environment, number of samples adaptable
memory	very low	very low	huge
precision	Position estimate given by junction and metric offset in the corresponding corridor	Topological map is represented by a set of Markov states (resolution 1m, $90°$ orientation steps)	Samples indicate position, only small errors

in the segment between junctions J_i and $J_{i'}$ in a distance of, e. g., 256 cm from J_i".

A metric self-localization method is used as a reference. To be able to compare the metric positions determined by the reference locator with the junction/distance pair returned by RouteLoc, the real-world position of each junction is determined in advance. Thus, it is possible to compute an (x, y, θ) triple from the junction/distance representation that can be compared to the metric position returned by the reference locator.

6.1 Scan Matching

The method used as a reference was developed by Röfer [18] and is based on earlier work by Kollmann and Röfer [10]. They improved the method of Weiß *et al.* [28] to build maps from measurements of laser range sensors (laser scanners) using a histogram-based correlation technique to relate the individual scans. They introduced state-of-the-art techniques to the original approach, namely the use of projection filters [13], line-segmentation, and multi-resolution matching. The line-segmentation was implemented employing the same approach that was already used for route generalization presented in Sect. 2.1. It runs in linear time with respect to the number of scan points and is therefore faster than other approaches, e. g. the one used by Gutmann and Nebel [8].

The generation of maps is performed in real-time while the robot moves. An important problem in real-time mapping is consistency [13], because even mapping by scan-matching accumulates metric errors. They become visible when a loop is closed. Röfer [18, 19] presented an approach to self-localize and to map in real-time while keeping the generated map consistent.

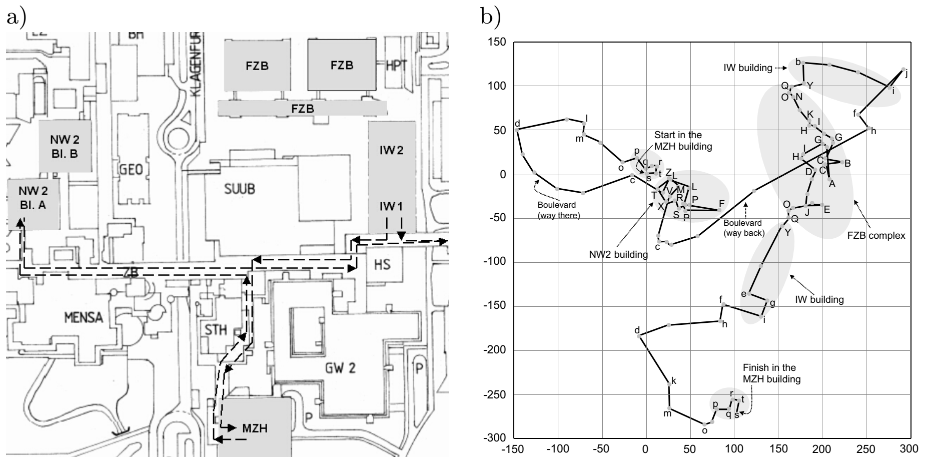

Fig. 12. a) The campus of the Universität Bremen (380m × 322m). b) Route generalization of odometry data recorded on the campus.

6.2 Experimental Setup

Experiments with the Bremen Autonomous Wheelchair "Rolland" have been carried out on the campus of the Universität Bremen (cf. Fig. 12a) . The wheelchair was driven indoors and outdoors along the dashed line shown in Fig. 12a, visited seven different buildings and passes the boulevard which connects the buildings. The traveled distance amounts to 2,176m. Traveling along this route with a maximum speed of 84cm/s takes about 75min. While traveling, the wheelchair generated a log file which recorded one state vector every 32ms. Such a state vector contains all the information available for the wheelchair: current speed and steering angle, joystick position, current sonar measurements, and complete laser scans. As mentioned, only locomotion data and the measurements of two sonar sensors are used for the self-localization approach presented here. Feeding the log file (192MB) into the simulator SimRobot [16], it is possible to test the algorithm with *real* data in a simulated world. Note that the simulator works in real-time, i. e. it also delivers the recorded data in 32ms intervals to the connected software modules, one of which is the self-localization module.

For the evaluation of the approach, a laser-scan map of the whole route was generated, using the scan matching method presented in [18]. For such a large scene, the laser map deviates from the original layout of the environment in that the relative locations of the buildings are not 100% correct. Therefore, the route-graph was embedded into the laser scan map making it possible to compare both localization results on a metric basis while traveling through the route with simultaneously active scan matching and route localization modules[1] It consists

[1] That is the reason why the layout of the route graph depicted in Fig. 13b differs from the map shown in Fig. 12a.

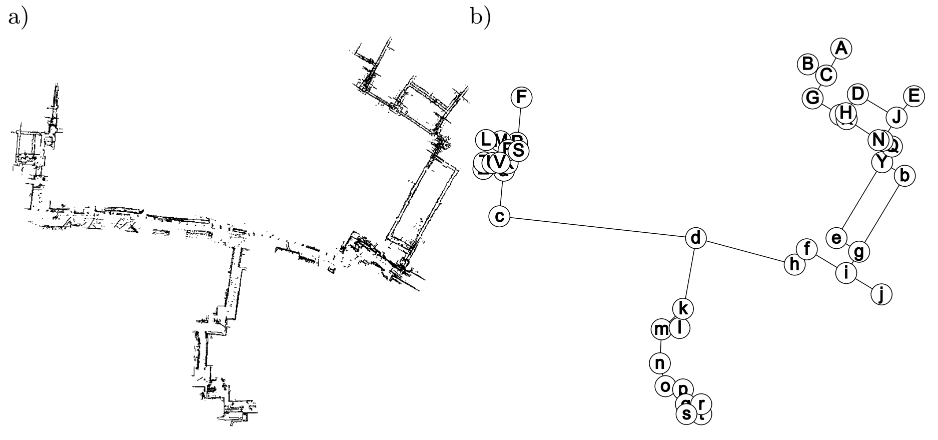

Fig. 13. a) Laser map generated along the route depicted in Fig. 12a. b) Route graph representing the relevant part of the campus.

of 46 graph nodes and 144 junctions. The represented corridors range in length from 4.3m to 179m.

The deviations between the metric positions determined by the reference locator and the locations calculated by RouteLoc are depicted in Fig. 14. Note that the horizontal axis corresponds to the travel *time* along the route and not to travel *distance*, i. e. the wheelchair stopped several times and also had to shunt sometimes, so that distances along this axis do not directly correspond to metric distances along the route.

As RouteLoc represents the environment as edges of a graph, its metric precision is limited. The edges of the route graph are not always centered in the corridors; therefore, deviations perpendicular to a corridor can reach its width, which can be more than 10 m outdoors (e. g. corridor *dc*). There are three reasons for deviations along a corridor: first, they can result from the location at which the current corridor was entered (see Sect. 3.3). The bandwidth of possibilities depends on the width of the previous corridor. Second, deviations can be due to odometry errors, because the wheelchair can only correct its position when it drives around a corner. In case of the boulevard (corridor *cdh*), the wheelchair has covered approximately 300 m without the chance of re-localization. Third, deviations can also result from a certain delay before a turn is detected (e. g. the peak after JE in Fig. 14). Such generalization delays are discussed in Sect. 3.3 and are also the reason for some peaks such as the one at the end of the boulevard (*dc*).

Even though the odometry data turned out to be very bad (see Fig. 12b), the approach presented here is able to robustly localize the wheelchair. It takes a while before the initial uniform distribution adapts in such a way that there is sufficient confidence to pose a reliable hypothesis about the current position

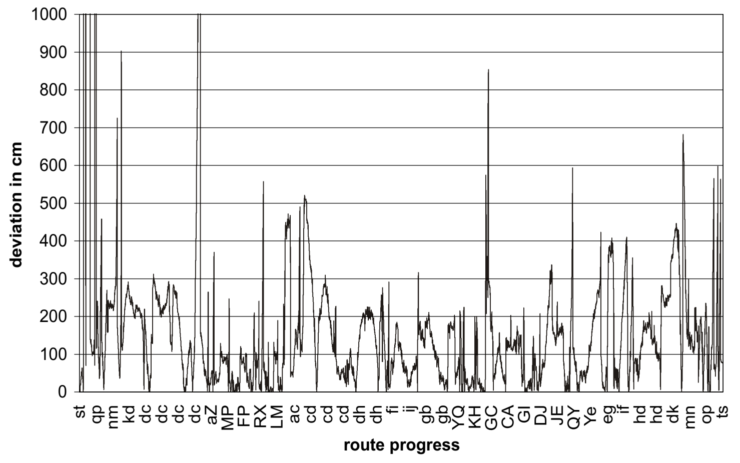

Fig. 14. Deviations of RouteLoc's position estimates from those made by the laser scan based localization. The letters correspond to segments between the junction labels used in Fig. 13b, but due to the lack of space, some are missing.

of the robot. But if this confidence is once established, the position is correctly tracked.

7 Conclusion and Future Work

Self-Localization of mobile robots in large-scale environments can be efficiently realized if a hybrid representation of the environment is used. The probabilistic approach presented here matches an incremental generalization of the traveled route with an integrated topological-metric map, the *route graph*. Real-world experiments at the Universität Bremen showed the robustness and efficiency of the algorithm. RouteLoc needs only very little input (only odometry data). It is fast and well-scaling, but is sometimes not as precise as other (metric) approaches. Therefore, it should be regarded as a basic method for absolute self-localization that can be extended on demand. In the first place, a disambiguation of situations and the resulting reduced time for the initial localization can be obtained if the route generalization and the route graph were augmented by feature vectors. Additional sensors to detect the features as well as dialogs with the human driver will help here.

RouteLoc will be extended such that self-localizing becomes possible even in a-priori unknown environments (SLAM). For this purpose, the robot has to build the route graph from scratch during runtime and, subsequently, it has to solve the problem of place integration. That means, it has to find out whether its current position is already represented in the route graph, or whether it is located in a corridor that is so far unknown.

Acknowledgements

The Deutsche Forschungsgemeinschaft supports this work through the priority program “Spatial Cognition”.

References

1. J. Borenstein, H. R. Everett, and L. Feng. *Navigating Mobile Robots – Systems and Techniques.* A.K. Peters, Ltd., USA, 1996.
2. W. Burgard, D. Fox, and D. Henning. Fast grid-based position tracking for mobile robots. In G. Brewka, Ch. Habel, and B. Nebel, editors, *KI-97: Advances in Artificial Intelligence*, Lecture Notes in Artificial Intelligence, pages 289–300, Berlin, Heidelberg, New York, 1997. Springer.
3. H. Choset and K. Nagatani. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17(2):125 – 136, April 2001.
4. A. Elfes. Occupancy grids: A stochastic spatial representation for active robot perception. In S. S. Iyengar and A. Elfes, editors, *Autonomous Mobile Robots*, volume 1, pages 60–70, Los Alamitos, California, 1991. IEEE Computer Society Press.
5. S. P. Engelson and D. V. McDermott. Error correction in mobile robot map learning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2555–2560. IEEE, May 1992.
6. C. Eschenbach, C. Habel, L. Kulik, and A. Leßmöllmann. *Shape Nouns and Shape Concepts: A Geometry for ‘Corner’*, volume 1404 of *Lecture Notes in Artificial Intelligence*, pages 177–201. Springer, Berlin, Heidelberg, New York, 1998.
7. D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo localization: Efficient position estimation for mobile robots. In *Proc. of the National Conference on Artificial Intelligence*, 1999.
8. J.-S. Gutmann and B. Nebel. Navigation mobiler Roboter mit Laserscans. In P. Levi, Th. Bräunl, and N. Oswald, editors, *Autonome Mobile Systeme*, Informatik aktuell, pages 36–47, Berlin, Heidelberg New York, 1997. Springer.
9. J.-S. Gutmann, T. Weigel, and B. Nebel. A fast, accurate, and robust method for self-localization in polygonial environments using laser-range-finders. *Advanced Robotics*, 14(8):651 – 668, 2001.
10. J. Kollmann and T. Röfer. Echtzeitkartenaufbau mit einem 180°-Laser-Entfernungssensor. In R. Dillmann, H. Wörn, and M. von Ehr, editors, *Autonome Mobile Systeme 2000*, Informatik aktuell, pages 121–128. Springer, 2000.
11. B. Kuipers, R. Froom, Y. W. Lee, and D. Pierce. The semantic hierarchy in robot learning. In J. Connell and S. Mahadevan, editors, *Robot Learning*, pages 141–170. Kluwer Academic Publishers, 1993.
12. A. Lankenau and T. Röfer. The Bremen Autonomous Wheelchair – a versatile and safe mobility assistant. *IEEE Robotics and Automation Magazine, “Reinventing the Wheelchair”*, 7(1):29 – 37, Mar. 2001.
13. F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.
14. A. Mojaev and A. Zell. Online-Positionskorrektur für mobile Roboter durch Korrelation lokaler Gitterkarten. In H. Wörn, R. Dillmann, and D. Henrich, editors, *Autonome Mobile Systeme*, Informatik aktuell, pages 93–99, Berlin, Heidelberg, New York, 1998. Springer.

15. I. Nourbakhsh, R. Powers, and S. Birchfield. Dervish: An office-navigating robot. *AI Magazine*, 16:53–60, 1995.
16. T. Röfer. Strategies for using a simulation in the development of the Bremen Autonomous Wheelchair. In R. Zobel and D. Moeller, editors, *Simulation-Past, Present and Future*, pages 460–464. Society for Computer Simulation International, 1998.
17. T. Röfer. Route navigation using motion analysis. In *Proc. Conf. on Spatial Information Theory '99*, volume 1661 of *Lecture Notes in Artificial Intelligence*, pages 21–36, Berlin, Heidelberg, New York, 1999. Springer.
18. T. Röfer. Building consistent laser scan maps. In *Proc. of the 4th European Workshop on Advanced Mobile Robots (Eurobot 2001)*, volume 86 of *Lund University Cognitive Studies*, pages 83 – 90, 2001.
19. T. Röfer. Konsistente Karten aus Laser Scans. In *Autonome Mobile Systeme 2001*, Informatik aktuell, pages 171–177. Springer, 2001.
20. T. Röfer and A. Lankenau. Ensuring safe obstacle avoidance in a shared-control system. In J. M. Fuertes, editor, *Proc. of the 7th Int. Conf. on Emergent Technologies and Factory Automation*, pages 1405 – 1414, 1999.
21. T. Röfer and A. Lankenau. Architecture and applications of the Bremen Autonomous Wheelchair. *Information Sciences*, 126(1-4):1 – 20, Jul. 2000.
22. J.S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, New Jersey, USA, 1995.
23. R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proc. of the Int. Joint Conf. on Artificial Intelligence, IJCAI-95*, pages 1080–1087, 1995.
24. S. Thrun. Learning maps for indoor mobile robot navigation. *Artificial Intelligence*, 99:21 – 71, 1998.
25. S. Thrun, W. Burgard, and D. Fox. A Real-Time Algorithm for Mobile Robot Mapping With Applications to Multi-Robot and 3D Mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 2000.
26. S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 101:99 – 141, 2000.
27. N. Tomatis, I. Nourbakhsh, and R. Siegwart. Simultaneous localization and map building: A global topological model with local metric maps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001)*, Maui, Hawaii, October/November 2001.
28. G. Weiß, C. Wetzler, and E. von Puttkamer. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *Proc. Int. Conf. on Intelligent Robots and Systems 1994 (IROS-94)*, pages 595–601, 1994.
29. S. Werner, B. Krieg-Brückner, and Th. Herrmann. *Modelling Navigational Knowledge by Route Graphs*, volume 1849 of *Lecture Notes in Artificial Intelligence*, pages 295–316. Springer, Berlin, Heidelberg, New York, 2000.
30. D. van Zwynsvoorde, T. Simeon, and R. Alami. Incremental topological modeling using local Voronoï-like graphs. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and System (IROS 2000)*, volume 2, pages 897 – 902, Takamatsu, Japan, October 2000.
31. D. van Zwynsvoorde, T. Simeon, and R. Alami. Building topological models for navigation in large scale environments. In *Proc. of IEEE Int. Conf. on Robotics and Automation ICRA 2001*, pages 4256 – 4261, Seoul, Korea, May 2001.