Synthesis of Indoor Maps in Presence of Uncertainty

Andrea Fusiello, Bruno Caprile

Istituto per la Ricerca Scientifica e Tecnologica, Via Sommerive 18, I-38050 Povo, Trento, Italy

A robotic system is presented, which is able to autonomously explore unengineered indoor environments, thereby synthesising maps suitable for planning and navigation purposes. Map recovery takes place through interaction between the robot and the world, in which either sensing and acting are affected by uncertainty. Kalman filtering is applied to maintain position best estimates, which are then fused with data coming from the observation of landmarks.

The proposed method has been implemented on a robot equipped with ultrasonic range finders, and tested in a fairly simple, real environment.

Key words: Robot exploration, Map synthesis, Dead reckoning navigation, Sonar, Machine learning.

1 Introduction

Maps synthesis from unsupervided exploration is a well known and challenging problem in robotics. In this paper, a system is presented which has proven up to the task, at least when working in fairly simple indoor environments.

The main objective of the research reported here is to develop a robotic system able to successfully accomplish the so called *weekend experiment* [21]:

(i) a robot is left alone in an office building (let us suppose it is 5:30 pm on Friday night);
(ii) the robot has no a-priori map;
(iii) the command "travel at your leisure around the environment and build a map of accessible space (without harming yourself)" is issued to the robot;
(iv) on Monday morning the robot is ready to execute commands using the map it learned on the weekend.

If sensors readings and motor actions were uncertainty immune, recovering the map would be merely matter of time. Yet, when dealing with real environments, models should be considered which are able to tolerate the presence of uncertainty.

In modeling the environment, a distinction is typically made between *geometric* and *qualitative* approaches. In the first, the spatial structure of the environment is represented through geometric primitives – fixed (grid-based representations like Occupancy Grid, first introduced in [24,16], and the Histogram Grid [3]) or adaptive (Voronoi Diagrams [5], Generalised Cones [4], Segment Models [9], Convex Polygon Models [8] and Polygonal Region Model [23]). Albeit fairly popular, geometric techniques present a main weakness in the use they make of error-prone metrical information.

Qualitative models abstract "relevant" features of the environment, establishing relationships among them. Typical qualitative representations are graph-based models and feature maps, that is, collections of uncertain spatial relationships among landmarks [20,21,26]. Graph-based models give a qualitative-topological representation of the environment: the world is modeled as a graph whose nodes represent places that – on the basis of appropriate perceptual criteria – are regarded as distinctive, and the arcs express spatial or functional relationships between nodes. This approach, pioneered by Kuipers [18], was inspired to human cognitive maps, and has more recently led to the introduction of qualitative mapping schemes [19]. Other well known works on graph-based mapping have been authored by Mataric [22], Dean and Basye [2,11,1], and Dudek et al. [13]. To the work of the latter the present paper is particularly inspired. In [1], the environment is represented as a finite state automaton. The robot is endowed with a set of actions and perceptions restricting its interaction with the world in such a way that the world/robot system behaves like a finite state automaton. The robot infers the structure of the automaton by experimenting with it in the presence of noise.

When dealing with real, unengineered environments a number of problems arise concerning the *navigation abilities* (namely, self-localisation and sensory data interpretation) of the robot being employed. The model of environment should therefore be "robust" with respect to uncertainties afflicting both perception and motion. In the present work, Basye's approach is extended in the direction of a greater robustness. In particular, the dead reckoning process is treated in a statistical framework [26]. Errors and uncertainties in position are dealt with Kalman filtering techniques [17], which allow to match positions of distinctive places with greater reliability and flexibility.

Our mobile robot is equipped with sonar range finders and an odometer, which provide sensor measurements for a set of basic reactive navigation modules [7]. The system has been thoroughly tested in our Institute building. Left free to explore the first floor, the robot is able to accomplish the task, extracting topologically correct maps, which are also sufficiently accurate to be used for navigation purposes.

In Sec. 2 the model of interaction is introduced, while Secs. 4, 5 and 6, contain a rather detailed description of the main modules of the system. In Sec. 7 results obtained when exploring a simple environment are reported.

2 The Model

The model of environment that will be adopted is inspired to early works of Kuipers [18,19], and consists in a graph whose vertices correspond to *Local Distinctive Places* (LDP); the edges are elemental paths (the *Conduits*) connecting LDPs. To each LDP a *signature* is assigned which abstracts the information collected in sensing the LDP. In fact, neither connectivity, nor the distinctiveness of places are to be considered in absolute terms: it is the robot that, endowed with appropriate perceptual and sensory-motor abilities, labels the LDPs and probes whether connections exist between them.

Our environment consists of straight corridors of different width, meeting at right angles; corridors are associated to Conduits, and junctions to Local Distinctive Places. The signature of the LDP consists in the position and the (oriented) shape of the junction; Conduits are labeled with the *navigation commands* whose execution causes the robot to traverse them. In this simple model of the environment, corridors may have only four oriented directions; conventionally [1] these will be indicated with `N`, `E`, `S`, `W`.

While the model of the environment is deterministic, the interaction between the robot and the environment is stochastic, being affected by the uncertainty arising from either inaccuracies in the measurements and the interpretation of sensory data. We will assume that:

- outcome of navigation commands can be predicted only with a certain probability;
- it is only with a certain probability that junctions are correctly distinguished from one another.

In Sec. 6 it will be shown how, under these assumptions, the graph underlying the environment can be exactly reconstructed. The idea is to cumulate a series of observations in a small number of stochastic matrices (one matrix for each navigation command). Entry i, j of matrix M_α is the frequencies of observing place j after the execution of the navigation command α starting from

[1] It may be worth stressing the fact that no relationship exists, in absolute terms, between these directions and their geographic counterparts.

place i. Under mild assumptions [12], the adjacency matrix of the underlying undirected graph can be recovered from such stochastic matrices.

3 The Architecture

The Map Learning System is structured in three main layers (see Fig. 1): each layer executes commands coming from the layer above by employing the functionalities provided by the layers below. The bottom layer is the Behaviour Layer, which, by means of a set of sensory-motor functionalities, interfaces the system with the world. The map learning process takes place in the other two layers: at the top, the Cartographer builds the qualitative map, controls the learning process and maintains the position of the vehicle. Immediately below, the Navigator is in charge of building the metric map, and manages to execute the navigation commands coming from the Cartographer.

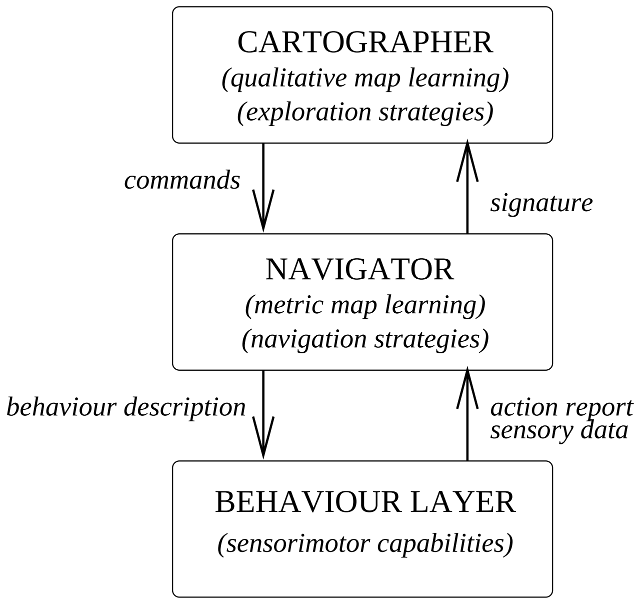

Fig. 1. The architecture of the Map Learning System.

4 The Behaviour Level

In this section, the Behaviour Layer is briefly illustrated (for more details, refer to [6,7]).

Its basic elements are (see Fig. 2):

– sensors and actuators;

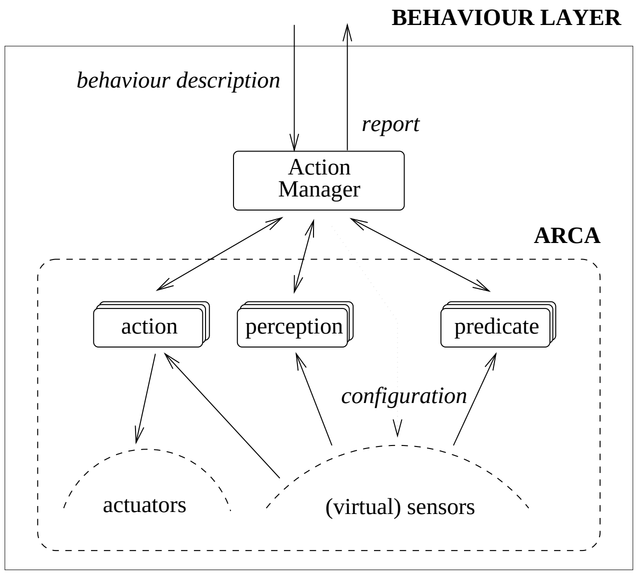

Fig. 2. The Behaviour Layer. Motor-actions, predicates and perceptions, as well as sensors and actuators, are executed as threads of a single process, called ARCA [7], which provides the runtime environment.

– motor-actions: modules implementing motor activities;
– predicates: modules implementing perceptual activities aimed at answering questions related to events relevant for the navigation;
– perceptions: modules implementing perceptual activities aimed at estimating interesting (physical or geometric) quantities.

A termination condition is a form composed by combining predicates. It typically triggers the disactivation of the motor action being executed or the activation of other motor actions, predicates or perceptions.

A basic behaviour is a pairing of a motor action and a termination condition (plus a set of optional fields); the former conveys the motor action that has to be performed and the latter the events upon which the motor action is to be ended. The syntax of a basic behaviour is the following:

```
(BASIC-BEHAVIOUR
    :motor-action <motor-action>
    :termination <termination>
    :report <perception> )
```

The interpretation and execution of basic behaviours is performed by the Action Manager. This receives the description of a basic behaviour from the Navigator; after verifying its correctness, the Action Manager handles the activation and disactivation of appropriate modules – thereby supervising their execution. Upon termination, the Action Manager reports on the execution back to the Navigator, adding the information corresponding to the activated perceptions.

The BEHAVIOUR LAYER consists of several modules. The motor actions relevant for the map learning process are:

`Follow-the-Corridor`: move along corridors;
`Move`: move straight ahead;
`Rotate`: rotate on place.

The predicates are:

`Rotated-Angle-Greater-Than`: it signals when the angle between the robot heading at the activation time and the current heading is greater than a given value;
`Left(Right)-Corridor-Detected`: it signals the presence of the left (right) lateral corridor;
`Frontal-Distance-Lower-Than`: it signals when the distance between the vehicle and a persisting frontal obstacle becomes lower than a given value.

Important perception for the exploration are:

`Get-Position`: it returns an estimate of the absolute position and orientation (p_x, p_y, θ) computed by the odometer. It also gives the 3×3 covariance matrix expressing the uncertainty of the position's estimate on the basis of the odometer's model (see appendix A);
`Get-Open-Space-Directions`: it returns a set of unobtructed directions as seen from the robot. When activated in a junction, it returns the directions of the departing corridors (see Appendix B);
`Store-Lateral-Sonar-View`: it provides the sequence of range measurements returned by the lateral sonars.

5 The Navigator

Two are the main tasks that the NAVIGATOR is called to accomplish: (1) to execute the requests coming from the CARTOGRAPHER ; (2) to build the metric map.

The NAVIGATOR receives from the CARTOGRAPHER high level commands described by one of the four directions `N`, `E`, `S`, `W` – their meaning being that of requesting the robot to navigate towards the indicated direction, until either a junction or the end of the corridor is met. In order to execute the commands the NAVIGATOR builds appropriate sequences of basic behaviours, and controls their execution by interacting with the ACTION MANAGER. When a failure occurs, the NAVIGATOR tries to apply simple recovery procedures. At the end, it reports back to the CARTOGRAPHER a data structure (called *signature*) containing information about the current place. The syntax is:

```
object SIGNATURE is
        shape: SHAPE_TYPE;
        position: ARRAY[3]{DOUBLE};
        covariance: MATRIX[3]{DOUBLE};
end;
```

The field `shape` encodes the shape of the junction as perceived by the robot; for example, the label *NSE* (to be intended as the set $\{N, S, E\}$) means that the robot has detected a junction with three corridors along the directions `N`, `S`, `E`.

On the basis of distances detected by the lateral sonars, the NAVIGATOR builds also the metric map – a gray level image representing a quantised top view of the environment. Each pixel corresponds to a square cell of side equal to 0.1 m; the grey value of each pixel (the *occupancy value*) reflects how many times the corresponding cell has been classified as obstructed or free during the exploration. At the beginning, all the pixels are given the same value (128); for each sonar return, the occupancy value of every cell is updated in agreement with a simple sonar model (see Appendix D), increasing the value of the cell when it is occupied, and decreasing it when it is free.

6 The Cartographer

As previously outlined, the main goal of the CARTOGRAPHER is to build the qualitative map while guiding the whole learning process. Moreover, the CARTOGRAPHER maintains an estimate of the current position of the robot, by applying the algorithm described in Sec. 6.2.

The CARTOGRAPHER sends the navigator commands like "move along the corridor in the direction of `N`". From the NAVIGATOR, it receives the signature of the distinctive places reached by the robot.

As explained in Sec. 1, the qualitative map contains both symbolic and metric information about the environment: it is structured as a labelled oriented graph whereby a node represents a *distinctive place* identified by the signature. In analogy with the Tour Model proposed by Kuipers [18], an arc between two nodes corresponds to a navigation command that has been observed to cause the robot go from the first to the second place.

Even though the adopted model is qualitative, metric information can be extracted from the graph: nodes contain the absolute position of distinctive places, and arcs are given the orientation of the corresponding corridors.

6.1 The Learning Algorithm

The main idea underlying the learning algorithm is to recover the graph by experimenting with it [1]: an "agent" executes actions, consisting in traveling along edges, and observes the resulting state – the node. Were the process completely deterministic, it would be sufficient, in order to recover the graph, that any one of the following conditions held:

(i) each node has a unique "signature";
(ii) the graph has a "distinguishing sequence" [11];
(iii) the agent is allowed to use a single marker that it can pick-up and leave in nodes [14].

At least in principle, the odometric position uniquely identifies each distinctive place. In practice, however, cumulative errors affecting dead reckoning may corrupt the position estimate to the point it becomes useless. Uncertainty in the position can be reduced whenever the robot visits an already visited place. Moreover, local sensory information can be exploited to identify places. The same sources of uncertainty affecting LDPs identification also affect the execution of commands. Since commands are based on sensory-motor loops, the uncertainty in sensing has an impact on the outcome of commands. It shall therefore be assumed that results of commands are not deterministic.

The algorithm described in [1], recovers the graph from noisy experiments, provided that the signature of every node is unique. The algorithm employs two basic data structure: a list, L, contains the signatures of the already visited places, while the adjacency information is stored in four matrices, M_α , α being equal to one (and only one) of the four direction `N`, `E`, `S`, `W`.

The strategy is fairly simple: for each pair of signatures (λ_i, λ_j) and for each command α, the algorithm cumulates in the i, j entry of matrix M_α the number of times that the observation of λ_i followed by the execution of α lead to the observation of λ_j. Provided that the frequency of visits at nodes is approximatively uniform, a number of visits exists that is sufficient to recover the graph from the stochastic matrices. In particular, if λ_j is the most frequently observed signature after the execution of α starting from λ_i, and λ_i is the most frequently observed signature after the execution of α^{-1}, starting from λ_j, then the labeled edge $(\lambda_i, \alpha, \lambda_j)$ is added to the graph. Notice how it is required that for each arc oriented along a given direction, another ought to exist between the same two nodes, but in the opposite way.

```
for each command $\alpha$
    INITIALIZE($M_\alpha$)
$L \leftarrow \emptyset$
$G \leftarrow \emptyset$
$\lambda_i \leftarrow NULL$
loop begin
        GETSIGNATURE($\lambda_p$);
        $\lambda_j$ = BESTMATCH($\lambda_p, L$)
        if $\lambda_j$ = NULL
          then
              INSERT ($\lambda_p, L$);
              $\lambda_j \leftarrow \lambda_p$
          else
              $\lambda_j \leftarrow$ Merge ($\lambda_j, \lambda_p$)
        INCREMENT($M_\alpha(\lambda_i, \lambda_j)$);
        INCREMENT($M_{\alpha^{-1}}(\lambda_j, \lambda_i)$);
        for each unfeasible command $\beta$
          do INCREMENT($M_{\beta^{-1}}(\lambda_j, \lambda_j)$;
        $\lambda_i \leftarrow \lambda_j$
        $\alpha \leftarrow$ CHOOSECOMMAND();
        EXECUTECOMMAND($\alpha$);
end
for each command $\alpha$
        for each $\lambda_i \in L$
          do begin
              $\lambda_j \leftarrow \max M_\alpha(\lambda_i, .)$)
              $\lambda_p \leftarrow \max M_{\alpha^{-1}}(\lambda_j, .)$)
              if $\lambda_i = \lambda_p$
                then ADDEDGE(($\lambda_i, \alpha, \lambda_j$), $G$)
          end
```

Fig. 3. The Learning Algorithm.

The core of the algorithm (see Fig. 3) is implemented by the following three procedures:

- BestMatch: it returns the signature in L that best matches signature λ_p according to their Mahalanobis distance (see Eq. B.1) and shapes. In particular, two signatures are defined as equivalent when the Mahalanobis distance between their respective positions is smaller than 7.81 (corresponding to 95% χ^2 confidence) and their shapes differ in no more than one element;
- Merge: it merges two matching signatures. It uses Kalman filtering (see Eq. B.4) to obtain a better position estimate and performs the intersection of the shapes.

– ChooseCommand: taking into account the current junction shape, it returns the less traversed direction among those along which the robot is expected to navigate successfully. This strategy makes the visiting frequencies asymptotically equalised.

6.2 Position Estimate

The process of learning the qualitative and metric maps heavily relies on the capability of maintaining good estimates of the position. Unfortunately, dead reckoning is prone to cumulative errors that cause position estimates to become quickly useless [27,15,20].

When two signatures are recognised as equivalent (BestMatch), their respective positions[2] and the two covariance matrices are merged into a new position and a new covariance matrix by using a Kalman filter (Merge). The uncertainty on the position estimate is therefore reduced in the new signature (see Appendix A). The special command `Set-Position` is then issued to the Behaviour Layer, thereby causing the odometer to be set to the new values of the position and covariance matrix.

7 Testing the System

The map learning scheme described in the previous sections has been implemented on a real robot and tested in an indoor, unengineered environment – the first floor of our Institute.

The robot is equipped with 8 Polaroid sonar range finders detecting reflecting surfaces in the range $[0.2, 4.0]$ m. The kinematics consists in two driving wheels plus a pivoting one in the rear. The computing power is provided by two on board Intel 486DX2 microprocessors, connected through a local Ethernet network.

The Cartographer and the Navigator are implemented as distinct processes running on the same processor, and exchanging data through *sockets*. The output of the Navigator is a grey-level image representing the occupancy grid as recovered from exploration of the environment (see Fig. 5); the output of the Cartographer is an undirected graph from which a map like that reported in Fig. 4b can be obtained by placing the nodes in the positions of the corresponding signatures.

[2] The position is a signature's field.

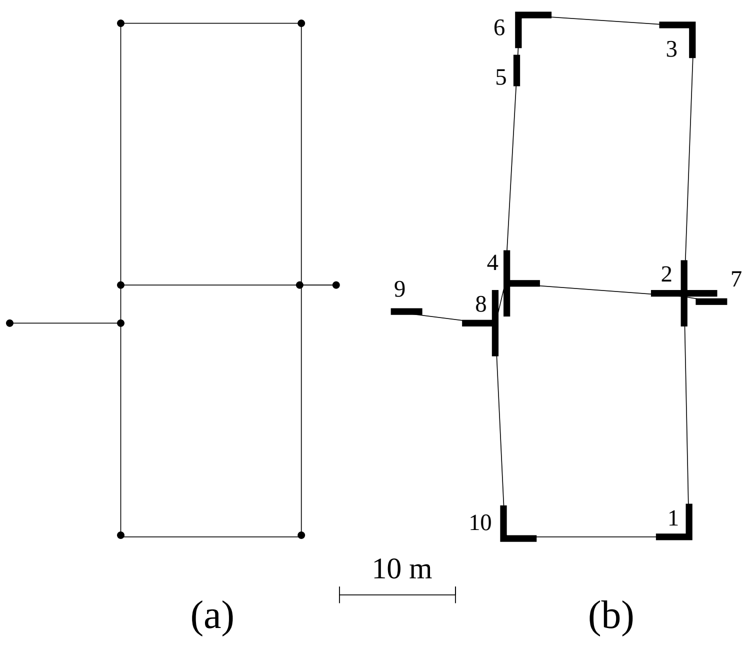

Fig. 4. The ideal (a) and the recovered map (b). LDPs are plotted at their respective positions, while the bold markers represent their shapes.

The experiment starts with the robot placed in an arbitrary position (in our case, the lower left corner of Fig. 5) with no *a-priori* representation of the environment. Under the control of the CARTOGRAPHER, the robot starts exploring the surrounding world by moving along the corridors. The exploration takes about two hours. A typical value for the average node visit frequency is 4, while the total length of the traveled path is about one kilometer. Notice how the resetting mechanism allows to reconstruct maps that are non only topologically correct but also metrically acceptable.

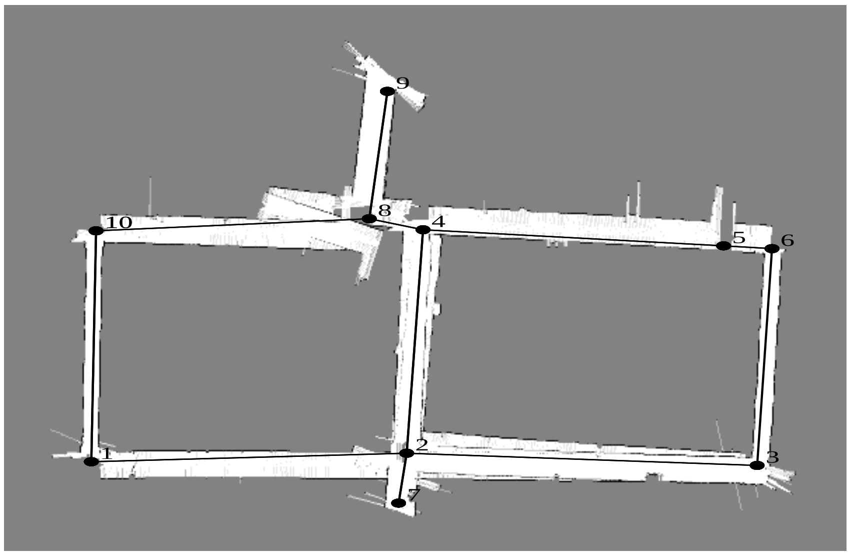

Fig. 5. The reconstructed grid-map. The grey level represent the occupancy value: the darker is the pixel, the more likely that the cell is occupied.

8 Conclusions

In this paper, a robotic system able to synthesise maps of unengineered indoor environments was presented. The robot autonomously navigates along the corridors, gathering data from the world. Starting from an exploration algorithm introduced by Basye, Kalman filtering was applied to reduce errors affecting position measurements.

The map that our system provides appears suitable for a systems needing only topologic information for planning and navigation, since any path on the map corresponds to a sequence of navigation commands. Our MAP LEARNING SYSTEM was developed in the framework of MAIA (Advanced Model of Artificial Intelligence) [25] – an integrated system aimed at executing fairly sophisticated missions in working environments. In order to evaluate the performance of our system in the light of the *week-end experiment*, it is therefore natural for us to compare its output with that needed by the navigation system of MAIA – an annotated bitmap encoding geometric *and* structural information. While either metric and topological information extracted by the MAP LEARNING SYSTEM are in this respect sufficiently accurate, methods to integrate the two are presently under investigation.

Acknowledgement

Authors wish to thank R. Cattoni for many helpful discussions. B. Crespi and S. Messelodi read early drafts of the paper and made useful comments. Remarks and suggestions by an anonymous referee helped improving the paper.

A Odometry

The position of the robot, $\mathbf{p}(t)$ can be computed as the integral over time of its speed:

$$\mathbf{p}(t) = \int_{t_0}^{t} K(\frac{\partial \mathbf{u}}{\partial t})dt, \qquad \text{(A.1)}$$

where K is the kinematics and $\mathbf{u}$ is the vector of shaft encoder readings. This process is called *dead reckoning.* A problem arises from the fact that the measurements from the encoders and the kinematic parameters are not error-free, so the computed position is affected by a cumulative error.

The vehicle has two opposed drive wheels, mounted on an axis of length L. Steering is obtained by imposing different speeds to the wheels. Main advantage of this configuration, called *differential drive*, is that of possessing a zero turning radius.

The kinematic state of the vehicle can be expressed as a triplet:

$$\mathbf{x}(k) = (p_x(k), p_y(k), \theta(k))^\top, \tag{A.2}$$

where $p_x(k)$ and $p_y(k))$ are the coordinates of the midpoint of the axis, and θ is the heading of the vehicle (that is, the angle between the perpendicular to the wheel axis and the x-axis taken counterclockwise).

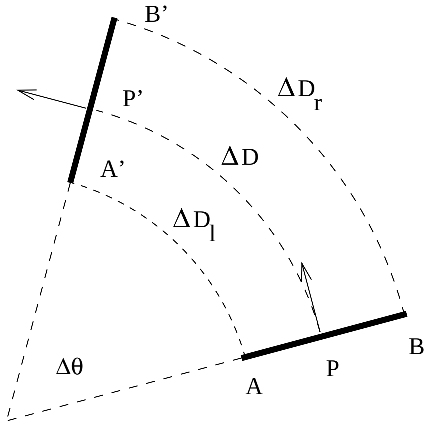

Fig. A.1. Vehicle's kinematics. The robot reference point is the midpoint of axis P; the right and left drive wheels are indicated by A and B, respectively. As the vehicle moves from P to P', an arc of length ΔD is covered while heading changes by $\Delta\theta$.

As shown in Fig. A.1, the traveled distance, ΔD, and the heading increment, $\Delta\theta$, can be computed in terms of the distances covered by the right and left wheels, which will be indicated with $\Delta D_r(k)$ $\Delta D_l(k)$, respectively:

$$\Delta D(k) = \frac{\Delta D_r(k) + \Delta D_l(k)}{2} \tag{A.3}$$

$$\Delta\theta(k) = \frac{\Delta D_r(k) - \Delta D_l(k)}{L}. \tag{A.4}$$

Simple geometric arguments give:

$$p_x(k+1) = p_x(k) + S\Delta D(k)\cos(\theta(k) + \Delta\theta(k)/2) \tag{A.5}$$

$$p_y(k+1) = p_y(k) + S\Delta D(k)\sin(\theta(k) + \Delta\theta(k)/2) \tag{A.6}$$
$$\theta(k+1) = \theta(k) + \Delta\theta(k) \tag{A.7}$$

where

$$S = \frac{\sin(\Delta\theta(k)/2)}{\Delta\theta(k)/2}.$$

In the first-order approximation this term vanishes, corresponding to consider PP' as a straight segment. A discrete dynamical system is then defined:

$$\mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{u}(k)), \tag{A.8}$$

where

$$\mathbf{u}(k) = (\Delta D_r(k), \Delta D_r(k))^\top$$

is the input.

A.2 Adding Noise to the Model

Due to various sources of error (weight distribution, the condition of the rubber of the wheels, the condition of the floor) the actual state of the robot may considerably differ form that obtained applying Eq. A.5. A way to account for these discrepancies is to assume that the system's input is perturbed additive, zero-mean noise, $\mathbf{v}(k)$, with known covariance matrix:

$$\mathbf{u}^*(k) = \mathbf{u}(k) + \mathbf{v}(k) \tag{A.9}$$

$$E[\mathbf{u}^*(k)] = \mathbf{u}(k) \tag{A.10}$$

$$C[\mathbf{u}^*(k)] = C[\mathbf{v}(k)]. \tag{A.11}$$

The covariance matrix, $C[\mathbf{v}(k)]$ of the input noise has the following form:

$$C[\mathbf{v}(k)] = \begin{pmatrix} \sigma^2\Delta D_r(k) & 0 \\ 0 & \sigma^2\Delta D_l(k) \end{pmatrix}, \tag{A.12}$$

where it has been assumed that: (1) relative errors on encoders readings are independent from one another and identically distributed; (2) their distribution has zero mean and variance σ^2, which can be estimated empirically; (3) the variance of the absolute error is proportional to the measured distance.

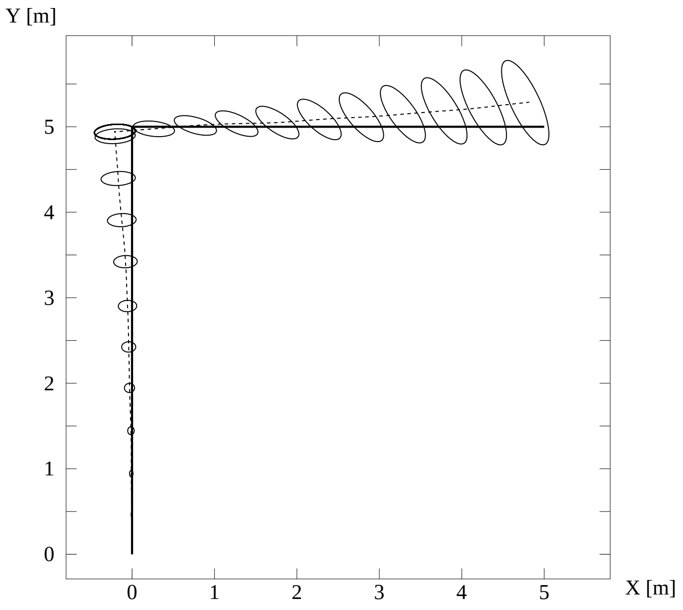

Fig. A.2. The position estimation error. The solid line is the real trajectory, while the dashed line is the trajectory as recovered from dead reckoning. 1% error on encoder readings was assumed. Ellipses representing 95% confidence contours for the position estimate are shown.

The state estimate $\hat{\mathbf{x}}$, is given by:

$$\hat{\mathbf{x}}(k+1) = F(\hat{\mathbf{x}}(k), \mathbf{u}^*(k)). \tag{A.13}$$

Linearising the system and assuming that noise and state are uncorrelated, the updating formula for the covariance matrix can be obtained in the form:

$$C[\hat{\mathbf{x}}(k+1)] = \mathbf{A}(k)C[\hat{\mathbf{x}}(k)]\mathbf{A}(k)^\top + \mathbf{B}(k)C[\mathbf{u}^*(k)]\mathbf{B}^\top, \tag{A.14}$$

where

$$\mathbf{A}(k) = \frac{\partial F}{\partial \mathbf{x}}|_{\hat{\mathbf{x}}(k)}; \qquad \mathbf{B}(k) = \frac{\partial F}{\partial \mathbf{u}}|_{\mathbf{u}(k)}. \tag{A.15}$$

Drawing a constant probability contour of the multidimensional gaussian distribution $N(\hat{\mathbf{x}}, C[\hat{\mathbf{x}}])$, an ellipsoidal uncertainty region for the estimate $\hat{\mathbf{x}}$ is defined whose boundary points satisfy the following equation:

$$(\mathbf{x} - \hat{\mathbf{x}})^\top C[\hat{\mathbf{x}}]^{-1}(\mathbf{x} - \hat{\mathbf{x}}) = c^2. \tag{A.16}$$

Fig. A.2 shows uncertainty regions for a simulated trajectory.

B Position Best Estimates

In the following $\hat{\mathbf{p}}$ will denote an estimate of the position $(p_x(k), p_y(k))$ of the vehicle, and $\Lambda = C[\hat{\mathbf{p}}]$ its 2×2 covariance matrix.

Let us assume that, given a position estimate $(\hat{\mathbf{p}}_o, \Lambda_o)$, we want to compare it with another estimate $(\hat{\mathbf{p}}_m, \Lambda_m)$. More precisely, in the assumption that each estimate is a sample from some (unknown) normal distribution, we want to test the hypothesis that both are drawn from the same distribution. A well known metric to compare gaussian distributions is the Mahalanobis distance [10]:

$$\|\hat{\mathbf{p}}_m - \hat{\mathbf{p}}_o\|_M = (\hat{\mathbf{p}}_m - \hat{\mathbf{p}}_o)^\top (\Lambda_m + \Lambda_o)^{-1} (\hat{\mathbf{p}}_m - \hat{\mathbf{p}}_o). \tag{B.1}$$

Since $(\hat{\mathbf{p}}_m - \hat{\mathbf{p}}_o)$ has a n-variate normal zero-mean distribution (with covariance equal to $(\Lambda_m + \Lambda_o)$), the Mahalanobis distance has a χ^2 distribution with n degrees of freedom. The hypothesis that $(\hat{\mathbf{p}}_o, \Lambda_o)$ and $(\hat{\mathbf{p}}_m, \Lambda_m)$ are taken from the same d-dimensional distribution at a α confidence level is rejected when their Mahalanobis distance is greater than $\chi^2_d(\alpha)$. For 2-dimensional normal distributions at 0.95 confidence level, $\chi^2_d(\alpha) \simeq 7.81$.

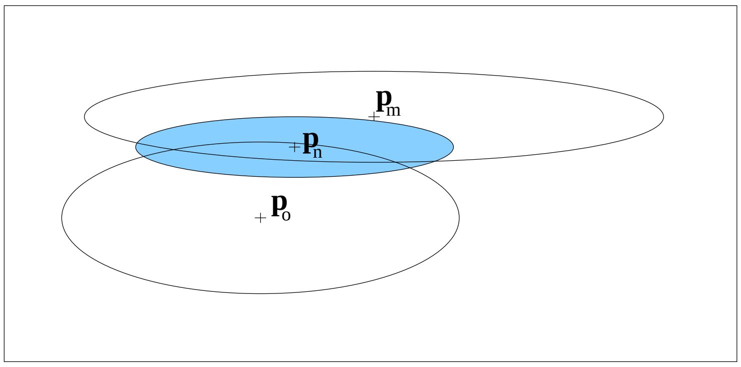

Fig. B.1. Merging two estimates: uncertainties regions for $\mathbf{p}_m$ and $\mathbf{p}_o$ ($\alpha = 0.95$). The shadowed ellipse represents the uncertainty region of $\mathbf{p}_n$, as resulting from the merging process.

When two estimates, $(\hat{\mathbf{p}}_m, \Lambda_m)$ and $(\hat{\mathbf{p}}_o, \Lambda_o)$, are recognised as corresponding to the same physical position, one may want to merge them into a new (and possibly better) estimate. The Kalman Filter [17] provides the resulting best unbiased linear estimate $(\hat{\mathbf{p}}_n, \Lambda_n)$, which can be computed from the following formulæ:

$$\mathbf{K} = \Lambda_m [\Lambda_m + \Lambda_o]^{-1} \tag{B.2}$$
$$\hat{\mathbf{p}}_n = \hat{\mathbf{p}}_m + \mathbf{K}(\hat{\mathbf{p}}_o - \hat{\mathbf{p}}_m) \tag{B.3}$$
$$\Lambda_n = (\mathbf{I} - \mathbf{K})\Lambda_m \ . \tag{B.4}$$

A slightly different version of above formula for Λ_n can be considered in order to force the covariance matrix to be positive definite:

$$\Lambda_n = (\mathbf{I} - \mathbf{K})\Lambda_m(\mathbf{I} - \mathbf{K})^\top + \mathbf{K}\Lambda_o\mathbf{K}^\top. \tag{B.5}$$

It is by construction that the two terms of the sum are positive definite (and symmetric), this implying that Λ_n is positive definite (and symmetric).

C Open Spaces Detection

When activated in a junction, the `Get-Open-Space-Directions` perception is expected to return the directions corresponding to the departing corridors.

Our robot is equipped with 8 Polaroid Ultrasonic Range Finder, which can measure distance from surfaces in the range [0.2, 4.0] meters. As it is well known, raw sonar readings are of limited accuracy and reliability. In our case the emission beam width is equal to 25.4^o and the wave length is 6.95 mm, hence angular resolution of the device is fairly poor (about 12.7^o) and most surfaces in the environment give rise to specular reflections.

The perception starts with an on-place rotation of the vehicle, during which a 360-view is obtained by combining two 180-views from taken from oppositely directed sonars. Referring sonar readings to a common, robot-centered reference frame, polar plots of the range profile can be obtained (see Fig. C.1). As it can be observed, measurements are fairly noisy; they are therefore filtered with a Gaussian kernel of semi amplitude approximatively equal to the angular resolution of the sonar. This process, filtering out high frequency artifacts, yields a smooth and more reliable depth profile, in which local maxima correspond to deep and broad obstruction-free areas. The filtered profile is then convolved with a simple derivative kernel: points of local maxima will correspond to decreasing derivative zero-crossings.

Some heuristics is now introduced. In particular:

- only sufficiently high maxima may correspond to corridors;
- maxima corresponding to corridors should be (approximately) spaced by multiples of 90^o.

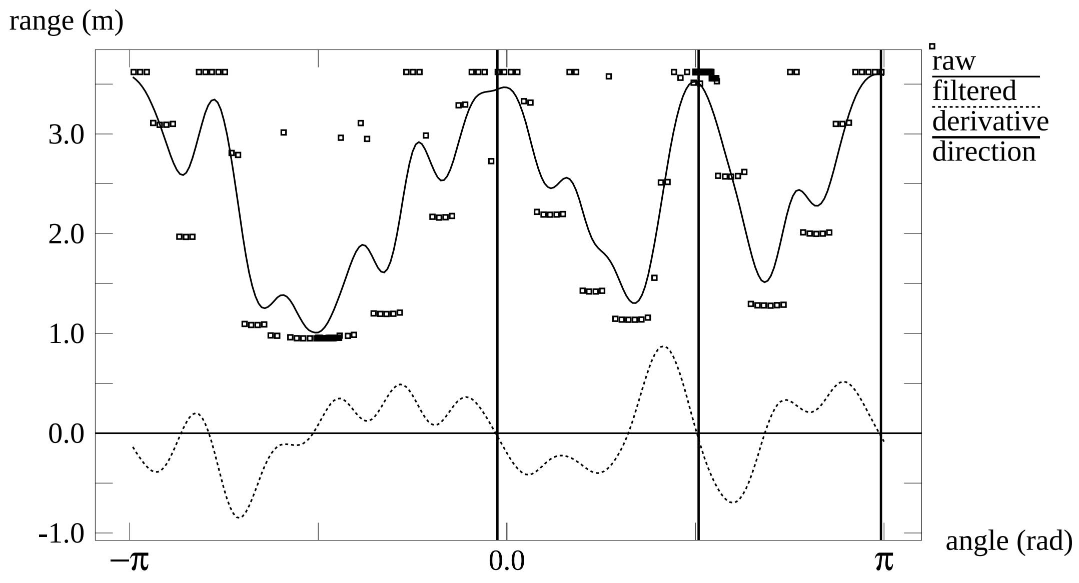

Fig. C.1. Processing of sonar readings as recorderd from a three-corridor junction. Raw measurements are reported as square dots. Solid line is the Gaussian filtering of raw data, while the dashed line correspond to its discrete derivative. Heavy vertical lines show the directions of corridors. Notice at far left a negative derivative zero-crossing which does not meet the orthogonality constraint.

D Occupancy Grid Update

Let us take a closer look to the computation of the occupancy-grid values. The vehicle position is given by a vector

$$\mathbf{x} = (p_x, p_y, \theta)^\top, \tag{D.1}$$

where p_x and p_y represent the vehicle position in a global reference frame and θ represents its orientation i.e. the angle between the robot heading and the x axis. Let us consider a robot-centered reference frame (x', y'), with the origin in the robot position and the x' axis aligned with the vehicle heading. The position and orientation of one sonar sensor, in the robot reference frame, are given by:

$$\mathbf{s}' = (s'_x, s'_y, \phi')^\top, \tag{D.2}$$

where ϕ' is the angle between x' axis and the sonar axis.

Trivially, the position and orientation of the same sensor in the global reference frame are given by:

$$\begin{pmatrix} s_x \\ s_y \end{pmatrix} = \begin{pmatrix} p_x \\ p_y \end{pmatrix} + \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} s'_x \\ s'_y \end{pmatrix} \tag{D.3}$$

$$\phi = \phi' + \theta. \tag{D.4}$$

If the sonar sensor returns a range reading r, in the simple model we adopt, the position of the reflecting target is computed as follows:

$$\begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} s_x \\ s_y \end{pmatrix} + r \begin{pmatrix} \cos\phi \\ \sin\phi \end{pmatrix} \tag{D.5}$$

The occupancy value of the discrete cell to which the target belongs is decremented (obstructed space), while the cells crossed by the sonar axis are incremented (they are free space). In the global reference frame this is the segment:

$$\begin{pmatrix} u_x \\ u_y \end{pmatrix} = \alpha \begin{pmatrix} s_x \\ s_y \end{pmatrix} + (1-\alpha) \begin{pmatrix} t_x \\ t_y \end{pmatrix}, \quad 0 < \alpha < 1. \tag{D.6}$$

This approach, inspired to the *Histogram Grid* [3], may appear to be oversimplified, when compared with others [16] which assign a probability distribution for the target location inside the sonar emission cone. Yet, a probability distribution is actually obtained by dense sampling of the sensor while the vehicle is moving.

References

[1] K. Basye. Graph-based mapping by mobile robots. In Wolfe and Chun, editors, *Mobile Robot VII*, pages 643–649. The International Society for Optical Engineering, November 1992.

[2] K. Basye and T. Dean. Map learning with indistinguishable locations. In Lemmer and Kanal, editors, *Uncertainty in Artificial Intelligence 5*, pages 331–341. Elsevier, 1990.

[3] J. Borenstein and Y. Koren. Histogramic in-motion mapping for mobile robot obstacle avoidance. *IEEE Transactions on Robotics and Automation*, 7(4):535–539, August 1991.

[4] R. A. Brooks. Solving the find-path problem by good representation of free space. In *Proceedings of the National Conference on Artificial Intelligence*, pages 381–386. AAAI, 1982.

[5] J. Canny and B. Donald. Simplified voronoi diagrams. *Computational Geometry*, 3(3):219–236, 1988.

[6] R. Cattoni, T. Coianiz, and B. Caprile. Planning and Reactivity for the Mobile Robot of MAIA. In *Proceedings of the 6th International Conference on Advanced Robotics*, Tokyo, Japan, November 1993.

[7] R. Cattoni, G. Di Caro, M. Aste, and B. Caprile. Bridging the gap between Planning and Reactivity: a Layered Architecture for Autonomous Indoor Navigation. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems IROS'94*, pages 878–885, Munich, Germany, September 1994.

[8] R. Chatila and J. P. Laumond. Position referencing and consistent world modeling for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 138–145, 1985.

[9] J. L. Crowley. Navigation for an intelligent mobile robot. *IEEE Journal of Robotics and Automation*, 1(1):31–41, March 1985.

[10] J. L. Crowley and F. Ramparany. Mathematical tools for representing uncertainty in perception. In A. Kak and S. Chen, editors, *Proceedings of the Workshop on Spatial Reasoning and Multi-sensor Fusion*, pages 293–302. Morgan Kaufmann, 1987.

[11] T. Dean, D. Angluin, K. Basye, S. Engelson, L. Kaelbling, E. Kokkevis, and O. Maron. Inferring finite automata with stochastic output functions and application to map learning. Technical Report CS-92-27, Brown University Department of Computer Science, 1992.

[12] T. Dean, K. Basye, and L. Kaelbling. Uncertainty in graph-based map learning. In J. Connel and S. Mahadevan, editors, *Robot Learning.* Kluwer Academic, 1993.

[13] G. Dudek, P. Freedman, and S. Hadjres. Using uncertain sensing data to create reliable maps: an algorithm for exploring/mapping unknown graph-like worlds. In Wolfe and Chun, editors, *Mobile Robot VII*, pages 650–660. The International Society for Optical Engineering, November 1982.

[14] G. Dudek, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation*, 7(6):859–865, December 1991.

[15] C. Durieu, J. Opderbecke, and Gilles Allegre. A data fusion application for location of a mobile robot using an odometer and a panoramic laser telemeter. In Groen, Hirose, and Thorpe, editors, *Intelligent Autonomuos Systems*, pages 519–529. IOS Press, 1982.

[16] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3(3):249–265, June 1987.

[17] Arthur Gelb, editor. *Applied Optimal Estimation.* The M.I.T. Press, 1974.

[18] B. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2:129–153, 1978.

[19] B. Kuipers and Y. T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8:47–63, 1991.

[20] J. J. Leonard and H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382, June 1991.

[21] J. J. Leonard and H. F. Durrant-Whyte. *Direct Sonar Sensing for Mobile Robot Navigation.* Kluver Academic, 1992.

[22] Maja. J. Mataric. A distributed model for mobile robot environment-learning and navigation. Technical Report AI-TR 1228, M.I.T., May 1990.

[23] D. Miller. A spatial representation system for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 122–127, 1985.

[24] H. P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 116–121, March 1985.

[25] T. Poggio and L. Stringa. A Project for an Intelligent System: Vision and Learning. *International Journal of Quantum Chemistry*, 42:727–739, 1992.

[26] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In Lemmer and Kanal, editors, *Uncertainty in Artificial Intelligence 2*, pages 435–461. Elsevier, 1988.

[27] C. M. Wang. Location estimation and uncertainty analysis for mobile robot. In Cox and Wilfong, editors, *Autonomous Robot Vehicles.* Springer-Verlag, 1990.