Efficient Occupancy Grids for Variable Resolution Map Building

Eric Dedieu, José del R. Millán[*]

Institute for Systems, Informatics and Safety
Joint Research Centre, European Commission
21020 Ispra (VA), Italy

1 Introduction

Map learning is a crucial issue in autonomous robotics, where robots must acquire appropriate models of their environment from their actual sensory perceptions of and interactions with the real world. The two main approaches are the *geometrical* and the *topological* ones. In the former, the map quantitatively reproduces the metric and spatial features of the environment (e.g., [Moravec, 88], [Borenstein & Koren, 91], [Schiele & Crowley, 94]). Such methods are computationally expensive and very vulnerable to errors that affect the metric information (i.e., odometry and range sensors). The alternative topological approach is a qualitative one, less vulnerable to sensory errors. A topological model consists of a graph where nodes represent perceptually distinct places (typically, visual landmarks) and arcs indicate the spatial relations between them (e.g., [Kuipers & Byun, 91], [Engelson & McDermott, 92], [Matarić, 92], [Borghi & Brugali, 95]). Such models involve more compact representations and allow fast planning. However they rely heavily on the existence of ever recognisable landmarks—a strong limitation. For a discussion of both approaches see [Thrun, 98] or [Borenstein et al., 96, ch. 8]. As the latter concludes, getting beyond laboratory settings remains an active research topic for both approaches.

Recently, [Thrun, 98] has developed a hybrid approach that integrates the geometrical and topological ones. It is characterised by the use of accurate metric maps (occupancy grids) as the basic level of representation. Then, this approach abstracts a topological graph from the global grid-based map in a second, off-line step. This involves computationally expensive representations and processing. Achieving real-time performance requires considerable algorithmic work and some simplifying assumptions (as well as the use of several computers).

Starting from the same basic concerns (hybrid map building) and tools (the use of local occupancy grids as basic primitives), we propose a very different approach. We are not interested in building a global and accurate metric map of the environment. It is too hard and heavy to build and maintain such a type of maps for complex and dynamic environments. We propose that topological and metric aspects are not to be viewed hierarchically but collaboratively, and should be worked out in parallel. In particular:

- *Spatially integrated metric information* is only used locally, in "local maps" that have been designed to allow fast robot position estimation. Local maps are built while the robot is following straight trajectories, thus they may have different sizes, adapted to the environment's complexity (i.e., constraints on trajectories).
- *Topological information* consists of connections between local maps, which overlap around connection points. It is adequate for global planning, and is used to propagate robot position information between local maps. However, such propagation is reduced to *adjacent* maps: we do not seek (nor need) a global geometric consistency between connected local maps.
- *Immediate metric information* is used for reactive navigation control, using a private local occupancy grid having a finer resolution than the one used for building maps. So the latter may use a resolution adapted to its purpose, leaving out fine details to be handled by the reactive controller.

[*] Email: {eric.dedieu, jose.millan}@jrc.it

The basic concept supporting each aspect of this work is the *variable resolution* adaptive strategy, built on previous work [Millán & Arleo, 97]. Different scales being adequate for different tasks, they should be able to cooperate harmoniously.

This paper presents two parts of the whole system we are developing. The first one is a *new method for integrating metric information into local occupancy grids* that makes them more appropriate for learning variable resolution hybrid maps. It has two major appealing features:

– A simple refinement naturally solves the problems arising from the discrete nature of occupancy grids. Such problems are responsible for the high computational cost of managing occupancy grids.
– The different "levels" of processing are not organised in a hierarchical way, but as equally available dynamic sensory resources. Each of them, however, represents invariant information characterised by different time scales—even if all the resources are updated at every cycle. This yields more and more "abstract" sensory invariants, which do not arise out of an explicit abstraction structure. This scheme allows a very fast and robust distributed sensory processing. The sensory resources that have been developed so far are: the immediate sensory readings, a local occupancy grid, a local dominant wall orientation, and a local grid of obstacle edges.

As a result, this new method makes fewer assumptions than our previous work [Millán & Arleo, 97] and it is considerably faster to process than any previous occupancy grid used for map building. Indeed, it takes just 5 ms to update all four sensory resources on a Sparc Ultra 1 Sun workstation, using non-optimised C++ code.

The second part reported is the *local map representation* used to model the environment by spatially integrating the local grids. It is characterised by:

– The use of edge information (the above mentioned "edge grid") as the main mapping primitive, occupancy information being but secondary. Edges are robust to dynamic objects of the environment and to spurious sensor readings.
– The use of a one-dimensional structure to represent a local 2D map. It allows fast matching and relocalisation, and therefore it may be used continuously as the robot moves.

We will only sketch how such local maps are connected into a global representation of the environment.

The paper reports experimental results obtained with TESEO, a Nomad 200 mobile robot. It is equipped with a bottom ring of 16 infrared sensors and a top ring of 16 sonar sensors. The proposed method has been systematically validated in natural, non-engineered environments: rooms with varied and moving furniture, a workshop (see Fig. 2), and a long corridor with furniture and metallic walls generating many spurious sonar readings. The time cycle at which resources are updated is 200 ms (the limiting factor is not computation time, but communication delay with the robot via radio-Ethernet).

2 Exploiting Continuity in Occupancy Grids

We use the variety of occupancy grids known as *histogram grids* [Borenstein & Koren, 91]. Every time a sensor reading falls within a cell of coordinates (i, j) within the grid—let us call that a "hit"—, the occupancy $C(i, j)$ of this cell is increased by 1. Then, all the cells are updated with a decay rate d in order to privilege more recent information. In this way, $C(i, j) * (1 - d)$ converges toward a "hit frequency"[2]. This method is much faster than the classical integration of hits using Bayes' rule. The latter is in a sense more exact, interpreting sonar hits as cone arcs rather than points. However, this interpretation assumes an accurate sensor model. Histogram grids have proven very efficient altogether, and parameter tuning to control the dynamics of the grid is much easier.

On the Nomad robot, the turret supporting the range sensors (sonars and infrareds) can rotate independently of the wheels. We exploit this feature to enhance the quality of the local occupancy grid. The turret is constantly turning left and right so as to ensure a good coverage over 360 degrees even when the robot is stopped.

[2] Given the sampling method, a hit frequency should not be interpreted as a probability of occupancy. This probabilistic interpretation would require to actually check that a cell is free in order to assign it a "no hit". On the contrary, here "no hit" only means that the cell was not sampled. A high certainty of occupancy is actually reached with quite low hit frequencies.

2.1 Two Usual Shortcomings

Histogram grids suffer from stability problems due to their discrete nature [Borenstein & Koren, 91]. When the robot changes cell, quantities tied to its position (e.g., discretised distances or speeds) change abruptly. This makes a high resolution (i.e., small cell size) necessary, and also requires filtering techniques.

Grid discreteness also raises the critical need for efficient geometric transforms (rotations and translations). A local grid is usually centered on the robot and must move with it. This involves expensive interpolation methods [Thrun, 98], or advanced mathematical analysis or their learning by neural networks [Van Dam, 98]. This is computationally heavy, and is a well-known shortcoming of grid techniques.

2.2 Including Continuous Information

These two shortcomings disappear when one removes two usual assumptions that are in fact unnecessary. The first one is that the robot should be perfectly centered on the grid: this implies that cells of the grid at time $t-1$ do not correspond to cells of the grid at time t. The second assumption is that the grid should be seen as a discretisation of the space: this is why any computation involving distances or speed suffers from stability problems.

We still demand the robot to be in the central cell, but *its position within this cell may vary*. Let us use for simplicity a scale of 1 unit per cell (in our case, 1 unit is typically 15 cm). The cell centre has coordinates $(0,0)$. Let x and y, taking values in $]-0.5, 0.5]$, be the coordinates of the robot within the central cell. When the robot moves without leaving the cell, x and y are simply updated. When the robot leaves the cell, the new cell it goes into becomes the central cell, and x and y are hereafter relative to the centre of the new cell. Algorithmically, this involves no global change (e.g., copying values) of the array representing the grid. For instance, in the case of a $15*15$ grid encoded in an array `occ[15][15]`, the central cell is at the outset element `[7][7]`. Should robot motion drive it one cell up, the apparent grid translation is coded by making the central cell point at `occ[7][6]`. In addition, the grid is coded as a torus. Thus, the "north edge" of the grid, which was initially at the row `occ[][0]`, is now at the row `occ[][14]`—whose previous values are deleted since it now represents an unknown area. Updating the grid after any translation in any direction is thus instantaneous.

Note that the grid does not rotate with the robot. But this doesn't imply that the grid's orientation is assumed to be constant relative to the environment: odometric errors can make it drift. However, the local grid itself is always consistent, because its rate of updating is much higher than the rate of accumulation of odometric errors.

In addition, we also keep precise information regarding hits. For cell (i,j), let $x(i,j)$ and $y(i,j)$ be the coordinates of the last hit in the cell, relative to its centre. Then, every distance computation over the grid will use those precise values (see examples below) as well as the precise robot position. The continuity of distance or speed manipulation, and therefore smooth transitions, are ensured *independently of the resolution of the grid*. For example, global rotations of the grid are made very easy and with limited information loss. An important point is that we do not keep the precise values because we *believe* in their accurateness, but only because the resulting continuity entails smooth and easy processing. We do not rely on the precision of metric values for any particular computation.

In this approach, *the grid resolution is not seen any longer as a discretisation* of the sensory space. Instead, *the grid resolution determines the density of information retained from sensory data*. For the sake of illustration, we have obtained the same qualitative results using cells of $60*60$ cm, $15*15$ cm or $7*7$ cm.

3 Finding and Integrating Edges

In this section, we present an algorithm to find edges in the local occupancy grid. Such edges provide instantaneous information: they are drawn at time T_i taking into account the occupancy grid at time T_i only. Edge information will thus have to be integrated over time and space. It is done by checking for a *dominant wall orientation* (and aligning with it if any), and by maintaining a local *edge grid* that handles hard problems of spurious sonar readings.

1. Take the hits that have fallen within the grid on the last sonar scan. Previously recorded hits are only used in steps 4 and 5.
2. Select candidate segments. If three or more consecutive hits are roughly aligned, they provide a candidate segment.
3. Enhance candidate segments by using information in the occupancy grid. The extremities of a segment are extended while they cross occupied cells. Inversely, an extremity that is preceded by a "hole" (i.e., a certain number of consecutive unoccupied cells) is shrunk.
4. Gather every recorded hit within 1 unit (i.e., 15 cm in our implementation) of the candidate segment. Segments may be discarded after this step if they have large "holes", if they are too short, or if they didn't gather enough hits.
5. Use gathered hits of a candidate segment for line fitting. Instead of using a standard method involving mathematical functions and iterations (e.g., the χ^2 method), we assume that the candidate segment's initial orientation is a good first approximation that needs just a slight correction. The least square fitting line can be shown to pass through the barycentre of the points, making with the initial orientation a corrected angle α such as:

   $$\frac{1}{2}\text{tg } 2\alpha = \frac{\sum_i x_i y_i}{\sum_i (x_i^2 - y_i^2)}$$

   x_i and y_i being the hits' coordinates in the frame centered on the barycentre and taking the initial segment as x-axis. Only one pass through the gathered hits is thus enough to compute the best fitted line.

Fig. 1.: The line fitting algorithm.

3.1 Line Fitting Algorithm

The algorithm does not work on the grid representation as hierarchically "higher" than immediate sonar readings. On the contrary, current readings are used to initiate the algorithm.

Fig. 1 shows the algorithm. Some minor details and parameters have been left out for simplicity. The parameters have to be tuned empirically, but this may be done incrementally, one at a time. Note that this algorithm does not require the grid to be aligned with the walls (see Fig. 2 for an example).

The processing time is proportional to the grid resolution but is independent of its total size, since the direct use of sensor readings make any search step unnecessary. Also, it does not seek to extract all edges in the grid; rather it aims to find very quickly those that are easy to determine. Uncertainties are not directly handled by the algorithm. Rather, we rely on temporal integration onto the "edge grid" (see Sec. 3.3) to cope with them, in the same way we do with the normal occupancy grid.

These are the main differences with the method of [Schiele & Crowley, 94], who use an extension of a Hough-transform to find segments in an occupancy grid and compute variances for each estimate. Note that their method would highly benefit from our extension to retain continuous information.

3.2 Dominant Wall Orientation

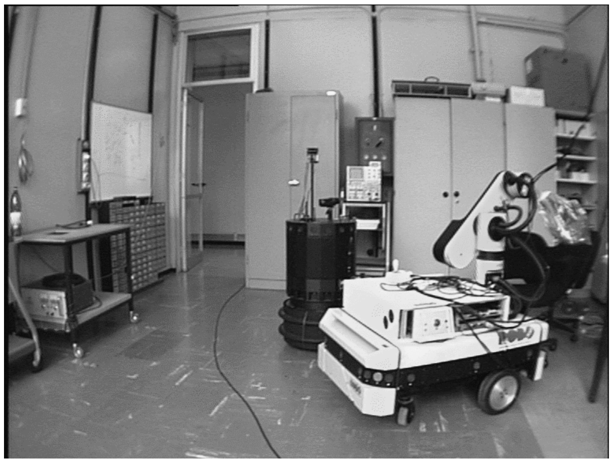
In human-made rooms, walls and furniture often make square angles with each other. In order to find the *dominant wall orientation* we maintain an histogram (with time decay and interpolation properties) of each edge's direction modulo $\frac{\pi}{2}$. This histogram is made of 8 sectors. If the histogram shows a strong peak, this is assumed to be the dominant wall orientation. The contents of the grid is then rotated to get aligned with this orientation. This automatically resets the robot's encoders. Once such an alignment is done, on-line checking of the dominant wall orientation allows to detect drifts in the odometry system, or to characterise areas that are oriented in a different way.

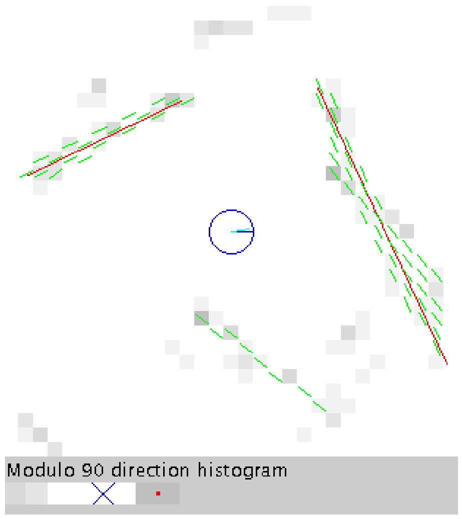
3.3 Edge Grid

The grid concept (involving the principles of Sec. 2.2) can be used with other kind of information than occupancy values. This section describes a local grid keeping edge orientation, namely the *edge grid*.

This sensory resource was developed after spurious sonar readings were systematically observed with metallic walls. While, in this case, occupancy information is unreliable, edge information has been demonstrated a relevant alternative resource. It is worth noting that edge information need not depend on occupancy values. For example, open doors are free space but may have a dominant orientation—that of the surrounding walls. Spurious "holes" generated by metallic walls can be filled up that way, too.

The edge grid building algorithm is very simple. Each cell contains a histogram of orientations modulo π, discretised in $\frac{\pi}{8}$ degrees steps. The edge and occupancy grids have the same resolution. Every time an edge is detected in the occupancy grid that goes through the corresponding cell in the edge grid, its orientation is integrated into the cell's histogram. When a strong peak emerges in the orientation distribution, it is taken as the cell's dominant orientation (a cell may have no dominant direction). Since orientation information is quite stable and not influenced by objects with high dynamics (e.g., walking people), the dynamics of the edge grid may be significantly lower than that of the occupancy grid.


(a)



(b)

Fig. 2.: (a) The office: Our robot is the cylindrical black one and is looking at the shelves in the background. (b) The corresponding local grid. Lines in a cell indicate its dominant edge orientation (edge grid). The two longer segments are the current edges found on the occupancy grid. These two edges approximate quite well the left wall and the set of shelves (even if the robot is not aligned with them). The doorway is the upper opening at the end of the right edge. Note that the local grid also represents the second robot and its dominant orientation, as well as a couple of chairs (bottom right). Below the local grid, there is the histogram of the dominant wall orientations. The point indicates the angle by which to rotate the map to get aligned with the wall (the 0 degree reference being shown by a cross).

3.4 Results

The local occupancy and edge grids (hereafter referred together as "the local grids") have been extensively tested in offices and corridors of our building, teleoperating the robot throughout or letting it move autonomously in the following way. Initially, a reflex behaviour makes it wander until the dominant wall orientation is found. Once the robot is aligned with it, a wall-following behaviour takes over. Wall orientation is constantly checked and is corrected on-line if the deviation gets between 5.5 and 11 degrees[3].

Fig. 2 shows the robot in a cluttered room and the corresponding local grids (details in the caption).

[3] This prevents frequent useless corrections, as well as large ones that are always caused by confusing areas, provided the first alignment was correct.

4 Building Maps

The local grids are used to build maps and localise the robot. In a map-building step, the robot moves along straight trajectories, and for each of them it records an "edge trace" by spatially integrating the successive local edge grids generated. Such edge traces are connected as the robot moves. Each edge trace is considered as a local map. Later on, local grids are matched against edge traces for position correction.

In this section we describe the edge trace structure and recording, as well as how matching and position correction are done. Then, we sketch our global localisation method using the whole graph of connected edge traces.

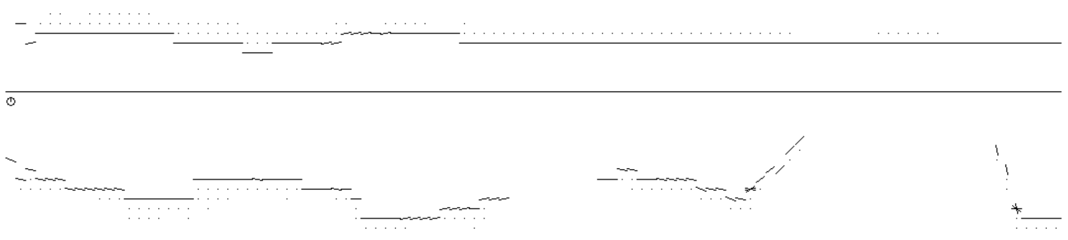


Fig. 3.: An edge trace of a long corridor. The central line is the robot trajectory. Small dots at each side show the distance intervals where edges can be located. Only large intervals of edge orientation are shown (there are two at the bottom-right). The initial grid orientation was good enough to record this long trace without lateral drifts. This was recorded with people passing by (and often stopping to watch the robot). Also, this corridor has metallic walls responsible for many sonar errors. Both problems are robustly handled in building edge traces.

4.1 Recording Edge Traces

An edge trace (hereafter simply "trace") is a vector of "features", indexed with a coordinate z. This z-axis represents the straight trajectory followed by the robot while recording the trace (discretised in grid units). The trajectories are parallel to the grid reference frame. Initially the edge trace is empty and z is arbitrarily set to 0. As the robot moves, the complete (robot-centered) edge grid is superposed onto the trace around the robot's current position at every cycle, and trace information is updated. All edge information is not recorded, but only a few features—located left and right of the trajectory. Such features must be sufficient for further localisation processes, and they are currently: the interval of possible edge orientations found in the grid, the interval of possible distances of such edges relatively to the robot's trajectory, and whether the edges (if any) were occupied or not. Such information represents reliably a 2D structure of that trace (fig. 3).

Traces are recorded until an obstacle is found in the way and remains for more than 30 s (this allows to ignore most spurious readings and moving persons), or until a hint from the user is received. Then, a new trajectory is begun, turning either 90 left or right, depending on the occupancy grid configuration and on what traces have been already recorded. Two successive traces are *connected* at the turning point (other connections may be added later or off-line).

There is another kind of edge traces, called *immediate traces*. They represent the current local grids in trace format. Moreover, immediate traces include two new features, which are the closest occupied cells to the left and right of the direction of travel. Even if, in general, such features are not discriminant by themselves, they are relevant for matching when there is no edge information.

4.2 Matching Local Grids with Traces

The main reason for representing a local map by a one-dimensional structure is to develop fast algorithms for matching traces. Fast processing is a primary concern, since our localisation strategy relies on the frequency of position corrections.

Let us suppose that the robot is positioned on a recorded trace T_r at position (z_0, u_0) in T_r's frame (the u-axis is 90 degrees left of the z-axis). The position correction problem is, having built an immediate trace

T_i out of the local grids, to seek in the neighbourhood of (z_0, u_0) at what position it matches T_r best. For this we must devise a way to *compare* the two traces.

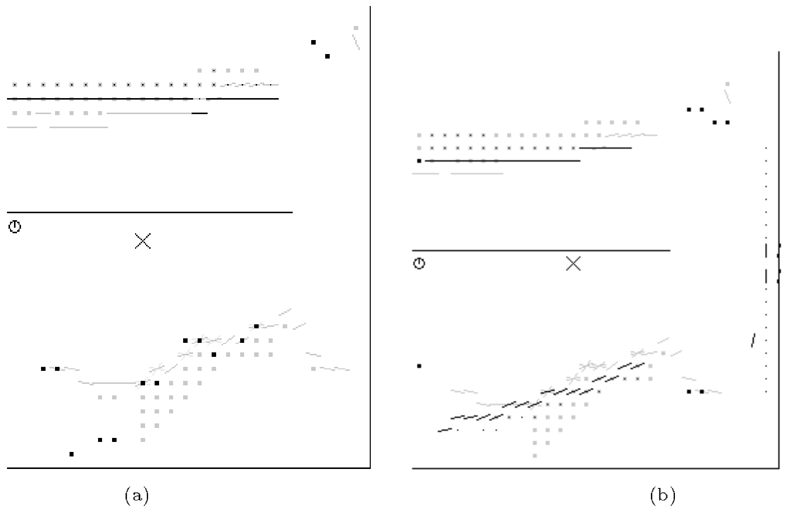


Fig. 4.: Two examples of an immediate trace (black) matched with a recorded trace (gray). In (a) no edge information (shown as segments) was available on the right of the immediate trace, so matching used occupancy information (shown as black dots). In (b) more edge information was available. The cross is the robot position, and the frame shows the limit of the local grids. The room is the same as in fig. 2a, the other robot being pushed against the right wall (at the bottom of the trace). Wall realignments occurred between the original trace recording and (a), and also between (a) and (b). The original pictures use colours for clarity.

If we superpose the robot-centered trace T_i at the supposed robot location on T_r, as done in fig. 4, it defines a correspondence between feature indexes. In our approach, comparison is done in two steps. First, a complete vector of *feature-to-feature similarities* is built. Then in a second step, the global similarity value is computed as the average (over the immediate trace) of the feature-to-feature similarities. Those may be negative or positive, a zero similarity meaning "I don't know". Actually, *three* different feature-to-feature similarities are independently computed and averaged: the *edge* similarity (concerned only with edge orientations), the *width* similarity (concerned with the left-to-right distances), and the *sides* similarity (concerned with the precise positioning of edges or occupied cells). The latter is the only one to be sensitive to the u_0 robot position. The current algorithms for computing feature-to-feature similarities are being incorporated better measures (see below), so we do not detail those algorithms here[4].

The advantages of computing a vector of similarities as an intermediary step are twofold. First, the possible visualisation of this vector helps to tune the individual parameters in the feature-to-feature comparison formulas. The second advantage is that other schemes than averaging may be tried to compute the global similarity from which corrections will be decided. For example, if the left side of T_i matches relevantly that of T_r but there is an incompatible area on the right side, then this may indicate a modification of the environment rather than poor matching. Indeed, the averaging scheme—our current one—is very vulnerable to e.g. furniture displacement.

There is an important piece of information missing in our positioning variables: we do not account for the displacements in rotation of T_i relatively to T_r. They are considered aligned[5]. Thus, correcting the robot position only means to translate it. Avoiding rotation concerns is another key for simple and fast processing. The rationale of such an assumption is:

[4] Let us only note that whenever T_i's occupancy information has to be used in width or sides similarities, any negative result becomes zero. This is because occupancy information is much unreliable. If it happens to coincide with T_r's edges, it is considered a positive hint, but if it does not, it it ignored because there may be too many reasons for having it there.

[5] Remember that the local grids do not rotate with the robot, so positioning has nothing to do with robot *heading*.

– We rely on the wall orientation alignment mechanism. Information about the dominant wall orientation is available often enough to prevent odometric drifts to exceed about 10 degrees.
– If we were to match long traces, even a small rotation uncertainty would yield large discrepancies between distance features at the extremities. However, the size of an immediate trace is limited: the maximum distance to the centre is 15 grid units.

All this limits the error done in considering both traces as aligned. So, given a range of possible corrections around (z_0, u_0), the best robot localisation is currently found in two steps. First, the z_0 position is corrected to maximise a combination of edge and width similarities. Then u_0 is maximised over the sides similarities. The proposed correction is only accepted if it improves significantly edge and sides similarities over the initial position. Given the nature of traces, the z-axis is the most appropriate for corrections. So, a natural extension will be to use pairs of two orthogonal connected traces, whenever possible, to yield the best combined localisation.

Though our initial tests are quite satisfactory, a deeper study and parameter tuning has still to be made to finalise the method (and make performance comparison with others). Its main interest is to be very fast and to avoid the heavy process of matching whole occupancy grids, which is the normal procedure in the field. Building an immediate trace and matching it with a given trace takes less than 20 ms.

A closely related work is that of [Schiele & Crowley, 94], who match grids through matching segments extracted from them. However, they use a gaussian uncertainty hypothesis, while we prefer to keep the full interval of possible values without privileging the mean. This is because edges found on occupancy grids often depend on where the robot is in the environment, and this source of variability cannot be considered as gaussian noise. Also, our comparison algorithm is not symmetrical, since the immediate trace is more versatile than the trace it is matched with. Also, we compare fixed-size segments (the size of the edge grid cells), while they use segments of maximal size. To our disadvantage, on the contrary, our method relies on a correct alignment with the dominant wall orientation.

4.3 Localising The Robot (sketch of the method)

Edge traces are connected into a graph (overlapping around connection points). The graph may be cyclic. This graph constitutes the “global map” of the environment.

We are not interested in maintaining a global, coherent geometric frame. Metric information is handled at the local level: *every edge trace within the map keeps its own robot position* in its private frame. The position is assigned a certain range of possible corrections. Should the robot leave the area where checking by matching is possible, that correction range increases automatically to model the accumulation of odometric uncertainties.

Positions may be corrected in two ways. Within a trace, corrections are tried at every time cycle[6]. Positions are also updated when the robot, which is on a trace T_r, enters a connected trace T_s. The resulting position undergoes correction, with a range depending on the quality of the connection. It may then override T_s private position (if any) if matching is better.

Loops in the graph are easy to handle since (1) we ask only for consistency between adjacent traces, and (2) the quality of a connection may be quantified. Reentering a trace after following a loop is simply modeled by adding a weak connection (this is currently only done off-line).

Preliminary result are encouraging. In short tests the robot has remained consistently located while moving. However we did not study stability over long-run tests, nor vary parameters, nor study the recovery from large localisation errors; and important details are still in development. Though this method is not mature yet for real comparisons, let us point two conceptual differences with other ones ([Engelson & McDermott, 92], [Borghi & Brugali, 95]) that also combine local metric frames into a global topological graph. First, the nodes of the graph are not distinctive places but complete maps on which the robot position may be checked constantly. Also, trace areas may overlap: several traces in the map may be active at the same time, and they may even show differences in their common area (this is a cause of robustness, not a cause of incoherence).

[6] This allows position information to be considered as another resource available to the robot, together with the local grids.

5 Perspectives and Conclusion

Our ultimate motivations are the usual map-based tasks: path planning, autonomous exploration for active map learning, labeling and recognition of places of interest (as an interface for human control). A good test application would be to use the robot as a guide for visitors of the laboratory. This will not be tackled until the complete map building and localisation processes are available. We have concentrated so far on efficient and robust primitives that have been constantly developed and tested in a bottom-up way, using a real robot and difficult environments. However, this bottom-up design has been guided by how complex tasks will be tackled. We have used either teleoperation or simple grid-based reactive behaviours to test several parts of our system.

The global coherence of all parts of our system is given by the *variable resolution* strategy first developed in this group in [Millán & Arleo, 97]. The current approach consists in using metric information only at local levels, with the resolution being adapted to the task. Moreover, all levels cooperate in a non-hierarchical way, relying on the frequency of simple and very fast processes, which we test from the beginning in natural environments.

The original contributions of this work for map building are:

- An improvement of occupancy grids (or similar grids) to use continuous information. This solves problems due to the discreteness of classical grids, and makes processing efficiency much less sensitive to grid resolution.
- The use of an "edge grid" as a mapping primitive resource, the occupancy grid being but secondary. Edges are robust to dynamic objects of the scene and to inaccuracies in sensor readings. They are commonly found in human-made environments, even in quite complex ones.

References

[Borenstein et al., 96] J. Borenstein, H.R. Everett, and L. Feng (1996). Where am I? Sensors and methods for mobile robot positioning. TR, Mobile Robotics Laboratory, University of Michigan.

[Borenstein & Koren, 91] J. Borenstein and Y. Koren (1991). The vector field histogram—fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, **7**(3):278–288, 1991.

[Borghi & Brugali, 95] G. Borghi and D. Brugali (1995). Autonomous map learning for a multi-sensor mobile robot using diktiometric representation and negotiation mechanism. *Proc. of Int. Conf. on Advanced Robotics.*

[Engelson & McDermott, 92] S.P. Engelson and D.V. McDermott (1992). Error correction in mobile robot map learning. *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2555–2560.

[Kuipers & Byun, 91] B.J. Kuipers and Y.T. Byun (1991). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, **8**:47–63.

[Matarić, 92] M.J. Matarić (1992). Integration of representation into goal-driven behavior-based robots. *IEEE Trans. on Robotics and Automation*, **8**:304–312.

[Millán & Arleo, 97] J. del R. Millán and A. Arleo (1997). Neural network learning of variable grid-based maps for the autonomous navigation of robots. *Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation*, 40–45.

[Moravec, 88] H.P. Moravec (1988). Sensor fusion in certainty grids for mobile robots. *AI Magazine*, **9**:61–74.

[Schiele & Crowley, 94] B. Schiele and J. Crowley (1994). A comparison of position estimation techniques using occupancy grids. *Proc. of the Int. IEEE Conf. on Robotics and Automation*, 1628–1634.

[Thrun, 98] S. Thrun (1998). Learning maps for indoor mobile robot navigation. *Artificial Intelligence*, to appear.

[Van Dam, 98] J. Van Dam (1998). *Environment Modelling for Mobile Robots: Neural Learning for Sensor Fusion.* PhD Thesis, Dept. of Computer Science, University of Amsterdam.

This article was processed using the TEX macro package with SIRS98 style