MUREA: A MUlti-Resolution Evidence Accumulation Method for Robot Localization in Known Environments

Marcello Restelli[1], Domenico G. Sorrenti[2], and Fabio M. Marchese[3]

[1] Politecnico di Milano
Dept. Elettronica e Informazione
Piazza Leonardo da Vinci 32, 20135 Milano, Italy
restelli@elet.polimi.it
[2] Universitá degli Studi di Milano - Bicocca
Dept. Informatica, Sistemistica e Comunicazione
Via Bicocca degli Arcimboldi 8, 20126 Milano, Italy
sorrenti@disco.unimib.it
[3] Universitá degli Studi di Milano - Bicocca
Dept. Informatica, Sistemistica e Comunicazione
Via Bicocca degli Arcimboldi 8, 20126 Milano, Italy
marchese@disco.unimib.it

Abstract. We present MUREA (MUlti-Resolution Evidence Accumulation): a mobile robot localization method for known 2D environments. It is an evidence accumulation method where the complexity is reduced by means of a multi-resolution scheme. The added value of the contribution, in the authors opinion, are 1) the method per sé; 2) the capability of the system to accept both raw sensor data as well as independently generated localization estimates; 3) the capability of the system to give out a (less) accurate estimate whenever asked to do so (e.g. before its regular completion), which could be called *any-time localization*.

Our experience in robotic research (specifically RoboCup competitions) allows us to assert that a localization system cannot rely on a reduced set of workspace features. It often happens that some of the features, expected to be detectable, are not perceived. The reasons for this can range from occlusion, to noise on the sensors, to imperfect algorithms processing the sensor data. Although we use omnidirectional vision as the main sensing, we do feel the convenience of a localization algorithm independent from a specific sensory system. Moreover, we wanted a system capable to accept raw sensor data, without any intermediate interpretation.

We aimed also at what we call *any-time localization*, i.e. the capability of the system to provide its best estimate whenever a timeout expires. This results very useful in a realistic real-time robot system; on the other hand, the usual behavior of other known robot localization algorithms is not to have an available output until their completion.

The remainder of the paper is organized as follows: in section 1 we very briefly review a very reduced set of localization approaches, the ones more related to

G.A. Kaminka, P.U. Lima, and R. Rojas (Eds.): RoboCup 2002, LNAI 2752, pp. 351–358, 2003.

our approach. In section 2 we describe our proposal. Some results are presented and discussed in section 3, while in section 4 we draw some conclusions.

1 Localization Methods

The approaches more related to the one here presented are all based on the use of sensors which provide dense data. They accomplish a match between dense sensor scans and the map of the environment without the need for extracting landmark features. Most of them base on probabilistic approaches.

Scan matching techniques [1] use *Kalman Filtering* and assume that both the movement and measurements are affected by White Gaussian Noise. These techniques being local methods, cannot recover from bad matches and/or errors in the model. *Grid-based Markov Localization* [2] operates on raw sensor measurements. This approach, when applied to fine-grained grids, could turn into a very expensive computation. In order to overcome the huge search-space, these techniques use to include several optimizations [3]; the most frequent being to update only the data in a small area around the robot. Recently, *Monte Carlo Localization* [4] gained increasing interest. This approach bases on randomly generating localization hypotheses. In contrast with Kalman filtering techniques, these methods can globally localize a robot; with respect to grid-based Markov localization, the Monte Carlo methods require less memory and are more accurate. Another interesting method is due to Olson [5]; in order to match the map generated by the robot sensors with the a priori known map of the environment, the method bases on a maximum-likelihood similarity measure. In order to find the pose with the best match, it imposes a grid decomposition of the search-space and applies a branch-and-bound technique to limit the travel. The main drawback of this approach is the high computational cost required to compare the maps.

2 System Architecture and Localization Algorithm

The system has three main components: the map of the environment, the perceptions and the localization engine. The environment is represented by a 2D geometrical map that can be inserted in the system "manually" (i.e. through a configuration file) or can be built by an automatic system, as e.g. in [6]. The map is made up of simple geometrical primitives, i.e. points, lines, circles, etc., that we call *types*. For each primitive a list of attributes must be specified, which describe its main characteristics. Moreover, for each map element we have to specify the sensors able to sense it.

On the other hand, we have sensor data and, possibly, other localization applications. Both produce perceptions, intended to be everything that provides information (even partial) about the the robot pose. Each perception is characterized by type, attributes, and the sensor that perceived it; these data are useful in order to reduce the number of comparisons between perceptions and map elements, as shown later in section 2.2.

The localization engine takes in input the map of the environment as well as the perceptions, and outputs the estimated pose(s) of the robot (if the environment and/or the perceptions have inherent ambiguities).

As mentioned above, we want an evidence accumulation method, where the difficulty is in accumulating the evidence while working at high resolution in order to get an accurate estimate. We divide the search-space in subregions (hereafter cells, section 2.1), to which we associate a counter, as usual in evidence accumulation methods. Since we deal with a localization problem for a mobile robot in 2D, the search-space is a 3D space, i.e. (X, Y, Θ), the coordinates of the robot pose.

Each perception increases the counter associated to a cell if some cell point is compatible with both the perception and the model (see section 2.2 for compatibility). Then, on the basis of the votes collected by each cell, the system selects (section 2.3) the ones which are more likely to contain the correct robot pose. This process is further iterated on the selected cells (section 2.4) until at least one termination condition is matched (section 2.5).

2.1 The Cells

The localization task is to find the point (x, y, θ) in the 3D search space (X, Y, Θ) that gives the best fit between the perceptions and the world model. In order to achieve this goal, the search-space is divided into cells. We currently define a cell as a convex connected subregion of the search space. In particular, unlike other approaches, we decided to use cylindrical cells, characterized by a circle in the plane (X, Y) and a segment on the Θ axis. The reason for this is the simplification in the subsequent section 2.2, on the voting phase.

The use of cylindrical cells implies that we have to cope with the overlapping of adjacent cells, in the (X, Y) subspace. We distributed the elements (circles in (X, Y)) of our decomposition as if we had an hexagonal tessellation (see Figure 1 on the right). This choice, with respect to other 2D tessellations, allows for a minimum overlapping between the elements of our decomposition. Each circle is therefore centered on an hexagon and its radius is equal to the length of an edge of it.

2.2 The Voting Phase

The system determines, for each cell, how many perceptions are compatible with both the map and a robot pose inside the cell. We define a function Γ that, given a perception, a map element and a pose returns *true* if, in that pose, the specified map element can generate that perception, otherwise it returns $false$:

$$\Gamma : P \times M \times L^n \rightarrow \{true, false\} \tag{1}$$

where P, M and L^n are respectively the set of all the possible perceptions, the set of all the map elements and the n-dimensional space of the robot pose.

We say that a perception p *votes* a cell C iff exists a map element m so that: $Attrib(p) \subseteq Attrib(m)$, $Sensor(p) \subseteq Sensor(m)$ and $\exists\ \bar{l} \in C \mid \Gamma(p, m, \bar{l})$,

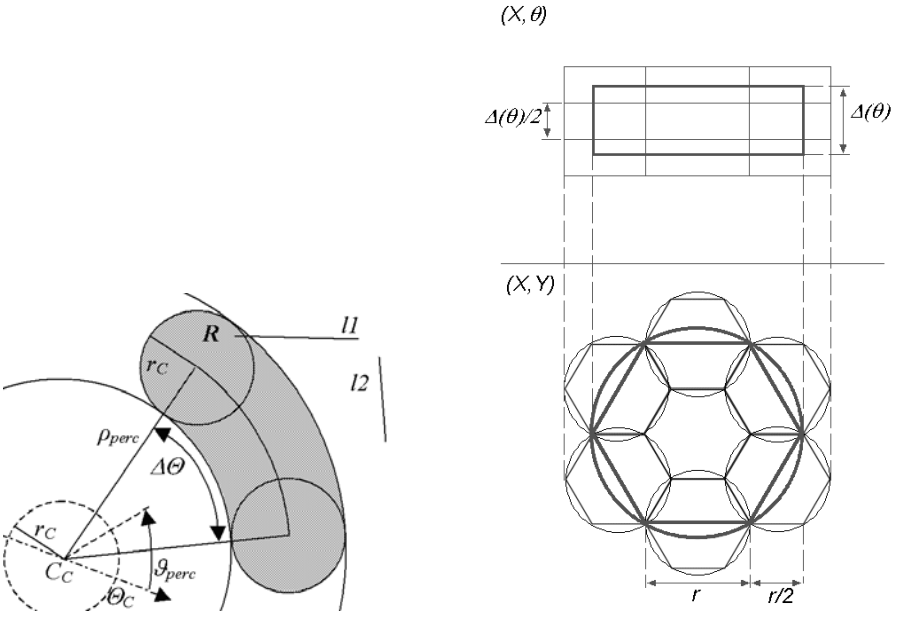

Fig. 1. On the left: Verification of the compatibility between a perception and a cell. On the right: The generation of the refined hypotheses

where *Attrib* returns the attributes of its argument and *Sensor* returns the sensor(s) associated to its argument.

In other words, the attributes of the perception must be one of those associated to the map element, the map element must be perceivable by the sensor that has produced the perception, and a pose inside the cell must exists that is *compatible* with both the perception and the map element. It is worth noting that Γ is in charge of checking whether a perception, e.g. a point perception, can be an observation of a certain map element for the pose associated to the cell, e.g. a line. The possible lack of homogeneity of the two items, i.e. perception and map-element, is therefore confined inside Γ.

See Figure 1 on the left for an example of what happens in this phase. Here we have to evaluate whether the point perception $(\rho_{perc}, \vartheta_{perc})$ votes the cell $C \equiv (C_C, \Theta_C, r_C, \Delta\Theta)$, where we called: C_C the point (X_C, Y_C), Θ_C the central value of the robot orientation interval for the cell, r_C the radius of the cell in the (X, Y) plane and $\Delta\Theta$ is the robot orientation interval.

In order to get the vote, a map element should be found inside the depicted region **R**. In the example we have two map elements, of type *line*; this type is obviously compatible with a *point* perception. In this case, the cell C gets the vote because the map element $l1$ intersects the region **R**, not because of the map element $l2$.

The more the votes of a cell, the more reliable is the match between the perceived world and its representation, and the more are the chances that the robot pose falls inside that cell.

2.3 The Selection Phase

After the voting phase, we select the most promising cells, which will be subdivided and considered in the next voting phase. By most promising we mean most compatible with the input sensor data, where the compatibility is as described above. Therefore the aim is now to search for the maxima in the vote accumulator. It should be now clear that a selection takes place at each level of resolution, which increases at each iteration of the whole process. At the each level we look for the absolute maximum, and we select all the cells which took more votes than a given percentage of the absolute maximum.

2.4 The Generation of the Refined Hypotheses

Given a cell C (father) of radius r and height $\Delta(\theta)$ its refinement consists in producing 21 cells (sons) with radius $r/2$ and height $\Delta(\theta)/2$, disposed as shown in Figure 1 on the right. In practice, we create a son with halved dimensions with respect to the father and put it at the center of the father. Then we surround it with other cells of the same size by respecting an hexagonal distribution of our tessellation elements (circles) in the (X, Y) plane. The union of the sons is a region larger than the region occupied by the father. The reason for this choice (different from those used in similar approaches) is in what we call *the phase problem*, i.e. in the misalignment of tessellation boundaries and the correct localization. Actually, if the correct pose falls near a border of a cell, because of the noise affecting the sensor data, some votes could scatter among contiguous cells.

2.5 The Termination Condition

The process sketched above terminates when any of the following condition holds.

1. The size of the cells gets smaller than the precision required by the application. This is the most favorable situation, since in this case the noise is not perceivable, at the precision required by the application.
2. The size of the cells gets so small that the noise on the sensor data creates a vote dispersion. This is an unfavorable situation because, due to the noise, the process comes to a stop before reaching the required precision.
3. The most voted cell did not receive more than a given percentage of the available votes. This is an even worse situation, because the support from the sensor data is really low.
4. A timeout expires. In this situation the precision of the solution is not relevant.

2.6 The Selection of the Solution

When one of the termination condition takes place, the system has to provide its currently best solution(s). In order to accomplish this task we cluster the

adjacent cells in the vote accumulator as long as voted cells are encountered. Then we compute the barycenter of the cluster as the weighted average of the cell centers. The weights are the votes received by each cell. This is the output of the system. It is supplied with an associated estimate of its precision. This estimate is based upon the size of the cluster, which is useful for discriminating the case in which the solution has been required too early (e.g. for a timeout) or when there is not enough support from the data. Moreover, the solution is output with an associated reliability, which is correlated to the received votes.

3 Localization Experiments

We made some experiments with a robot equipped with an omnidirectional vision system, based on a multi-part mirror [7]. This mirror allows the vision system to have a resolution of about 40 mm, for distances under 6 m. We put the robot into a partially built RoboCup F2000 field. In the experiments we did not use either the flag-posts nor the poles that should surround the field, so the experiments are quite general. We defined three map elements of type line and attribute "blue" to describe the blue goal, seven lines and one circle with attribute "white" to describe the borders of the field. We also specified that every map element can be perceived by the vision sensor. The vision sensor produces just two types of perceptions: white point perception and blue direction perception. Since the goals are not flat objects, some of the green-blue transitions generates blue point perceptions which are affected by large radial errors; therefore we used only the direction for the blue perceptions.

We placed the robot inside the field at known positions and then we ran the localization procedure. The required accuracy was set to 100 mm and 1^o; the system will stop when the cell size will be smaller than the setting. The reference system is located in the center of the field, with the x axis directed toward the blue goal. The computation ran on a PC Pentium III 800 MHz under Linux Mandrake 8.1.

In order to clarify the functioning of the localization process, in the following we show how the number of selected cells changes at each step of the process. Two experiments are reported in Figure 2 and in table 1.

The data in the tables are organized as follows: each row is associated to a level, i.e. a set of cells that have the same size, and contains the data for the selected cells. In the first two columns there are the dimensions of the cells, expressed by their radius and height. The third column contains the ratio between the number of cells that has been selected and the number of cells tested for compatibility, while the last column contains the percentage of votes (with respect to the maximum number of votes available) obtained by the most voted cell.

In the first experiment 58 perceptions were collected by the vision system. The real robot pose, manually measured, was $x = 20.5\ cm$, $y = -150\ cm$, $\theta = 0^o$. The localization process returned the pose $\widehat{x} = 17.07\ cm$, $\widehat{y} = -146.19\ cm$, $\widehat{\theta} = 1.04^o$. The whole process took 367 ms.

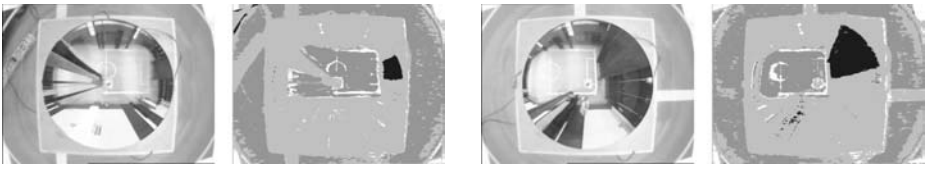

Fig. 2. Experiments: the first couple of images is related to experiment 1, while the second one is related to experiment 2

Table 1. Experiment 1 on the left, Experiment 2 on the right; data of the cells which have been selected

$\rho(cm)$	$\Delta(\theta)(^o)$	cells	% max votes
320	90.00	4/4	100%
160	45.00	22/56	100%
80	22.50	28/209	100%
40	11.25	7/305	99%
20	5.63	3/64	91%
10	2.81	1/43	81%
5	1.41	2/21	67%

$\rho(cm)$	$\Delta(\theta)(^o)$	cells	% max votes
320	90.00	4/4	100%
160	45.00	23/56	100%
80	22.50	17/207	100%
40	11.25	5/229	95%
20	5.63	3/62	90%
10	2.81	2/49	83%
5	1.41	3/35	71%

In the second experiment 163 perceptions were collected by the vision system. The real robot pose, manually measured, was $x = 350\ cm$, $y = -150\ cm$, $\theta = 0^o$. The localization process returned the pose $\widehat{x} = 345.87\ cm$, $\widehat{y} = -154.81\ cm$, $\widehat{\theta} = -1.38^o$. The whole process took 483 ms.

Including the results from other experiments, the evaluation of the errors are summarized in the following: average position error $= 5.63\ cm$, maximum position error $= 9.32\ cm$, average orientation error $= 0.72^o$, maximum orientation error $= 1.41^o$.

We have not yet introduced any optimization to the code, neither on the algorithmic side neither on the compiling. Especially for the first reason, we are confident to be able to drastically reduce computation.

We can draw some preliminary considerations from these experiments. By using the hierarchical search technique, the system had to test 702 cells in the first experiment and 642 in the second one. If we had used an a priori grid made up of cells with radius $= 5\ cm$ and height $= 1.4^o$, we should have had to test about 10^6 cells. This would have implied several minutes of computation before returning any solution.

The data in table 1 show the typical evolution of the search process: at the beginning we have large cells that collect several votes and there are a number of maxima, then, as long as the cell size decreases, we can notice a reduction in the votes and in the number of the selected cells.

4 Conclusions

We presented a mobile robot localization method for known 2D environments. We claim that evidence accumulation methods provide robustness against noisy

data; henceforth we devised a grid-based method where the complexity is reduced by means of a multi-resolution scheme. The added values of the contribution are its multi-resolution scheme, the capability to deal both with raw sensor data as well as other localization estimates, and what we call *any-time localization*, which is the capability of the system to give out a pose estimate whenever asked to do so. We designed the system in order to have it independent from the geometric primitives used in the model definition as well as from the sensory system.

The experimentation confirm the ideas behind the approach, even though no code optimization has been carried out, that allows to expect large speedups.

References

1. Gutmann, J., Burgard, W., Fox, D., Konolige, K.: An experimental comparison of localization methods. In: proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS98). (1998)
2. Burgard, W., Fox, D., Hennig, D., Schmidt, T.: Estimating the absolute position of a mobile robot using position probability grids. In: AAAI/IAAI, Vol. 2. (1996) 896–901
3. Burgard, W., Derr, A., Fox, D., Cremers, A.: Integrating global position estimation and position tracking for mobile robots: the dynamic markov localization approach. In: proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (1998)
4. Dellaert, F., Fox, D., Burgard, W., Thrun, S.: Monte carlo localization for mobile robots. In: proceedings of the IEEE International Conference on Robotics and Automation (ICRA99). (1999)
5. Olson, C.F.: Probabilistic self-localization for mobile robots. IEEE Transactions on Robotics and Automation **16** (2000) 55–66
6. Lu, F., Milios, E.: Globally consistent range scan alignment for environment mapping. Autonomous Robots **4** (1997) 333–349
7. Marchese, F.M., Sorrenti, D.G.: Omni-directional vision with a multi-part mirror. In: proceedings of the International Workshop on RoboCup. (2000) 289–298