Global Localization in SLAM in Bilinear Time

Lina M. Paz, Pedro Piniés, José Neira, Juan D. Tardós
Computer Science Dept. of the University of Zaragoza,
María de Luna 1, 50018 Zaragoza (Spain).
E-mail: {linapaz, ppinies, jneira, tardos}@unizar.es

***Abstract*— In this paper we study the global localization problem in SLAM: the determination of the vehicle location in a previously mapped environment with no other prior information. We show that, using a grid sampling representation of the configuration space, it is possible to evaluate all vehicle location hypotheses in the environment (up to a certain resolution) with a computational cost that is bilinear: linear both in the number of map features and in the number of sensor measurements. We propose a *pairing-driven* algorithm that considers only individual measurement-feature pairings and thus, in contrast with current correspondence space algorithms, it avoids searching in the exponential correspondence space. It uses a voting strategy that accumulates evidence for each vehicle location hypothesis, assuring robustness to noise in the sensor measurements and environment models. The general nature of the proposed strategy allows the consideration of different types of features and sensor measurements. Using the popular Victoria Park dataset, we compare its performance with *location-driven* algorithms where the solution space is usually randomly sampled. We show that the proposed pairing-driven technique is computationally more efficient in proportion to the density of features in the environment.**

***Index Terms*— SLAM, Global Localization, Grid Sampling, Voting Algorithms**

I. Introduction

A problem of considerable attention in SLAM is the determination of the location of a vehicle in an environment, given a set of m local measurements and a map of the environment with n features, but no prior information about the vehicle location. A solution to this *global localization* problem allows to restart the SLAM algorithm when the vehicle is lost or when it arrives to a previously mapped area with no odometry or high odometry error. In environments or situations in which no GPS fix is possible, this constitutes the only alternative.

Most current global localization algorithms are based on some form of consensus, and therefore are *robust* to spurious sensor measurements, or the detection of features not present in the map. In current applications of interest, because large environments are being considered, the issue of global localization has shifted towards *efficiency*. Since sensors are typically local, and therefore the number of measurements will normally be bound, attention is drawn to how well these algorithms scale with the size of the map.

Global localization algorithms can be classified as *location-driven* or *pairing-driven*. Location-driven strategies explore the *configuration space*, where a set of s different vehicle localization hypotheses, or location samples, are considered. Each is ranked according to how well the local measurements match the environment map at the hypothesized location. Algorithms that use random sampling of the configuration space, such as Monte Carlo localization [1], and those that use grid sampling, such as Markov Localization [2], [3], belong to this category. If the environment map is represented as an occupancy grid, the computational complexity of these algorithms, called here `Loc_driven`, is $O(s \cdot m)$.

In contrast to location-driven techniques, pairing-driven techniques explore the *correspondence space*, where data association hypotheses are produced considering consistent combinations of measurement-feature pairings. For the most promising hypotheses (in terms of the number of pairings), the vehicle location can be computed and the hypothesis can be verified. Algorithms that belong to this category traverse the exponential Interpretation Tree [4] using techniques such as: hypothesize and test [5], branch and bound [6], maximum clique [7], or random sampling [8]. In this last work it is shown that introducing the concept of *locality*, any pairing-driven algorithm can be made linear with the size of the map, but there remains in the Interpretation Tree search a component exponential in the number of measurements.

In this work we propose a *pairing-driven* algorithm, `Pair_driven`, that uses a voting strategy, in the spirit of the Generalized Hough Transform [9]. In contrast to current correspondence space algorithms, it evaluates all location samples considering only individual measurement-feature pairings. This avoids the exponential computational cost of interpretation tree traversal, resulting in an algorithm that is $O(n \cdot m)$, linear with both the number of environment features and of sensor measurements. We show that this pairing-driven algorithm is computationally faster than location-driven algorithms in proportion to the density of features in the environment. The main reason is that our algorithm exploits the lattice structure offered by a grid sampled configuration space (another problem in which a lattice structure offers advantages is motion planning [10]). Markov Localization [2], [3] uses grid sampling, but does not take advantage of the lattice structure that the grid offers.

This paper is organized as follows: section II describes both location-driven algorithms that work in the configuration space, and pairing-driven algorithms that traverse the correspondence space. In section III we analyze the alternatives in the representation of both the environment and the configuration space, and their effect in the computational complexity of these algorithms. In this section we also carry out a probabilistic analysis of the robustness of these voting

algorithms. Section IV contains comparative experimental results of running both algorithms in a large outdoor environment, using the popular Victoria Park dataset obtained by Guivant and Nebot [11]. Finally, in section V we draw the main conclusions of our work and set future directions of research.

II. Algorithms for global localization

Assume that the environment map consists of n features $\mathbf{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_n\}$ distributed in a 2D environment. Without loss of generality, we will consider 2D point features, so that $\mathbf{f}_j = (x_j,\ y_j)^t$. Assume a sensor mounted on the vehicle obtains observations of m features $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$, gathered from the vehicle location to be determined. The information available from each measurement depends on the type of sensor. In the case we are considering, sensors can give range-only (sonar), bearing-only (a camera), or range and bearing (laser) measurements of 2D points.

The configuration space is the space of possible vehicle location hypotheses of the form $\mathbf{x} = (x,\, y,\, \phi)^t$, where $x \in [x_{min}, x_{max}]$, $y \in [y_{min}, y_{max}]$ and $\phi \in [\phi_{min}, \phi_{max}]$. Given that the configuration space is continuous although bound, the number of alternative vehicle location hypotheses is infinite. A limited number of hypotheses, or location samples, within the bounds, must then be considered. Monte Carlo methods [1] work by drawing random samples in this space, while Markov Localization [2], [3] draw grid samples. Both random sampling and grid sampling methods can be tuned to adequately represent the solution space for a given resolution.

Assume that a set $\mathbf{X}$ of s samples are obtained from the configuration space. We can either uniformly divide the configuration space in n_x, n_y and n_ϕ grid elements in x, y and ϕ, respectively, or equivalently, we can randomly sample the configuration space in $s = n_x \cdot n_y \cdot n_\phi$ location samples. The appropriate resolution depends on the distribution of features in the environment and on the precision of the local sensor.

Global localization algorithms work by evaluating the compatibility between sensor measurements and environment features considering their individual properties, such as length, radius, color, etc., but mainly analyze their relative geometry and the compatibility between the measurements predicted from the map and the actual measurements. Both Monte Carlo methods and Markov Localization methods compute the likelihood of each location sample given an occupancy grid map and the measurements.

In the case of feature-based maps, the correspondence space techniques that are usually applied try to maximize the number of observations that successfully match a feature in the map. The implicit assumption is that the likelihood of a location hypothesis increases with the number of matchings. In section III-D we perform a probabilistic analysis that supports this assumption.

Thus we consider global localization algorithms that evaluate each location sample by computing its corresponding number of observation-feature matches. There are two basic alternatives with respect to how to proceed in the evaluation of each sample in the set $\mathbf{X}$ of alternative vehicle locations: *location-driven* strategies, and *pairing-driven* strategies.

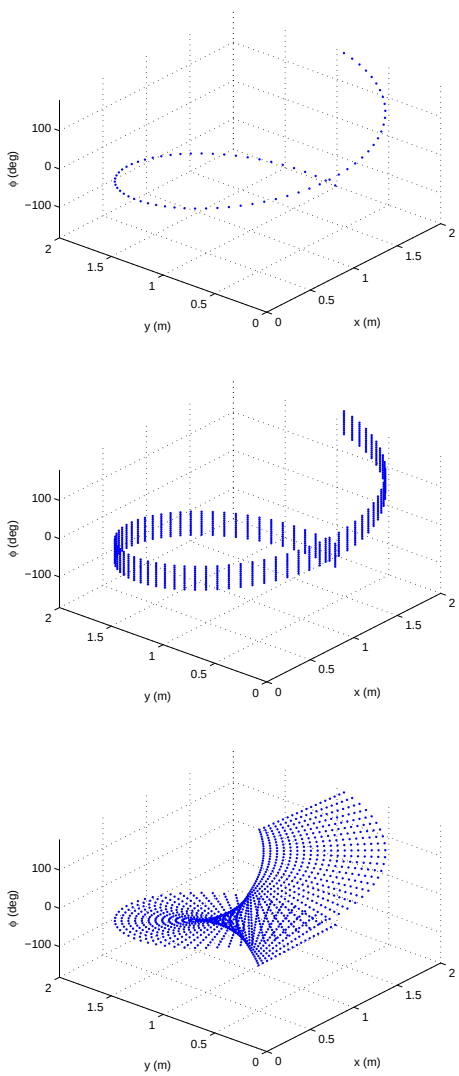
Algorithm 1 `Loc_driven`:
consider each alternative location hypothesis in turn

```
votes = 0
for each hypothesis x ∈ X do
  for each measurement z_i ∈ Z do
    F_i = predict_features(x, z_i)
    if any_compatible_feature(F_i, F) then
      votes(x) = votes(x) + 1
    end if
  end for
end for
```

Algorithm 2 `Pair_driven`:
consider each measurement-feature matching in turn

```
votes = 0
for each measurement z_i ∈ Z do
  for each feature f_j ∈ F do
    X_ij = hypothesize_locations(f_j, z_i)
    X_v = compute_compatible_locations(X_ij, X)
    votes(X_v) = votes(X_v) + 1
  end for
end for
```

- In a *location-driven* strategy, each vehicle location hypothesis $\mathbf{x}$ is considered in turn. Each of the m sensor measurements will agree with the location hypothesis if its hypothesized absolute location matches one or more of the n environment features. Algorithm 1, `Loc_driven`, implements this alternative. $\mathbf{F}_i$ is the set of hypothesized feature locations that would produce the measurement. In the case of range and bearing, it will be one 2D point, in the case of range, it will be a set of 2D points in a circle around the vehicle location sample at the measured range distance, etc. The validity of a measurement-feature association can be determined using error bounds; statistical validations using χ^2 tests can also be carried out. In this algorithm, each location hypothesis is ranked according to the number of measurements for which a feature association can be established. This strategy amounts to considering each candidate location and counting the number of votes cast by measurements that agree with that location.
- In a *pairing-driven* strategy, each measurement $\mathbf{z}_i$ is considered in turn. For every feature $\mathbf{f}_j$ in map $\mathbf{F}$, we compute the vehicle location(s) from where the feature could have produce the measurement. We then find the locations in $\mathbf{X}$ compatible with these hypothesized locations, and update their ranking (see algorithm 2, `Pair_driven`). You can consider this as a voting strategy in the spirit of the Hough transform [9]: each mea-



$\mathbf{X}_{ij} = \texttt{hypothesize\_locations}(\mathbf{f}_j,\ \mathbf{z}_i)$:
; $\mathbf{f}_j = (x_j,\ y_j)$

; For Range and Bearing:
; $\mathbf{z}_i = (\rho_i,\ \theta_i)$
;
$\mathbf{X}_{ij} = [\,]$
for ϕ = ϕ_{min} to ϕ_{max} step ϕ_{step} **do**
  $(x_i,\ y_i) = \texttt{pol2cart}(\theta_i + \phi,\ \rho_i)$
  $\mathbf{X}_{ij}$ = $[\mathbf{X}_{ij}; [(x_j - x_i)\ (y_j - y_i)\ (\phi)]]$
end for

; For Range Only:
; $\mathbf{z}_i = \rho_i$
;
$\mathbf{X}_{ij} = [\,]$
for ϕ = ϕ_{min} to ϕ_{max} step ϕ_{step} **do**
  for θ = θ_{min} to θ_{max} step θ_{step} **do**
    $(x_i,\ y_i) = \texttt{pol2cart}(\theta + \phi,\ \rho_i)$
    $\mathbf{X}_{ij}$ = $[\mathbf{X}_{ij}; [(x_j - x_i)\ (y_j - y_i)\ (\phi)]]$
  end for
end for

; For Bearing Only:
; $\mathbf{z}_i = \theta_i$
;
$\mathbf{X}_{ij} = [\,]$
for ϕ = ϕ_{min} to ϕ_{max} step ϕ_{step} **do**
  for ρ = ρ_{min} to ρ_{max} step ρ_{step} **do**
    $(x_i,\ y_i) = \texttt{pol2cart}(\theta_i + \phi,\ \rho)$
    $\mathbf{X}_{ij}$ = $[\mathbf{X}_{ij}; [(x_j - x_i)\ (y_j - y_i)\ (\phi)]]$
  end for
end for

Fig. 1. Hypothesized vehicle locations in the X space, generated by a measurement of a 2D point feature at $(1.0, 1.0)$, whose location relative to the vehicle is at (1.0, 0.0), when the measurement gives range and bearing (top), range only, such as polaroid sonars (center), and bearing only (bottom). Algorithms on the right show the actual implementation of function $\mathbf{X}_{ij} = \texttt{hypothesize\_locations}(\mathbf{f}_j,\ \mathbf{z}_i)$ for each case.

surement $\mathbf{z}_i$ will determine the set of vehicle locations $\mathbf{X}_{ij}$ from which it is feasible to detect the environment feature $\mathbf{f}_j$ in order to produce the given measurement, and vote for location samples $\mathbf{X}_v$ sufficiently close to those feasible locations. It amounts to consulting each voter (the measurements) in turn and adding a vote to each of the candidates (the vehicle location samples) it points out. The amount of votes the measurement casts depends on the type of map features and sensor measurements. Fig. 1 shows the votes that a measurement would cast when detecting a 2D point feature at coordinates $(1.0, 0.0)$ with respect to the vehicle, when the feature is at absolute coordinates $(1.0, 1.0)$, and the sensor is range and bearing (for example a laser scanner), range-only (sonar), and bearing-only (a camera).

Both strategies are bound to obtain roughly the same results, and so from the point of view of robustness, `Loc_driven` and `Pair_driven` can be considered basically equivalent. In the next section we discuss how the representation that we choose, both for the environment map and for the configuration space, will play a crucial role in the computational cost of these algorithms.

III. Computational complexity and robustness of global localization algorithms

A. Computational complexity of `Loc_driven`

The representation of the environment map has a very important effect in the complexity of `Loc_driven`. This algorithm traverses the whole configuration space, computing the support for each of the $s = n_x \cdot n_y \cdot n_\phi$ candidate locations. It gathers evidence by predicting the absolute location of each of the m measurements in trying to find any compatible

Sensor	`Loc_driven`	`Pair_driven`	`Pair_driven`/`Loc_driven`
Range and bearing	$O(n_x \cdot n_y \cdot n_\phi \cdot m)$	$O(n \cdot m \cdot n_\phi)$	$n/(n_x \cdot n_y)$
Range-only	$O(n_x \cdot n_y \cdot n_\phi \cdot m \cdot n_\theta)$	$O(n \cdot m \cdot n_\phi \cdot n_\theta)$	$n/(n_x \cdot n_y)$
Bearing-only	$O(n_x \cdot n_y \cdot n_\phi \cdot m \cdot n_r)$	$O(n \cdot m \cdot n_\phi \cdot n_r)$	$n/(n_x \cdot n_y)$

Fig. 2. Comparative computational cost of `Loc_driven` and `Pair_driven`

feature. If the absolute map feature coordinates are maintained *explicitly*, in a location vector $(\mathbf{f}_1 \cdots \mathbf{f}_n)^t$, a naive implementation of `any_compatible_feature` would sequentially consider all the n alternative features in $\mathbf{F}$. The computational complexity would then be $O(n_x \cdot n_y \cdot n_\phi \cdot m \cdot n)$.

In some cases, such as when features are 2D points and the sensor gives range and bearing measurements, the map can be preprocessed into a tree, and then finding a match for a predicted measurement at a hypothesized location can be computed in $O(\log n)$ instead of $O(n)$.

Alternatively, an *implicit*, *indexed* representation of the absolute feature locations can be maintained, for example using an occupancy grid or binary image. Using such a representation, determining whether a predicted measurement is present in the map amounts to consulting the corresponding element(s) in the occupancy grid or image. If we have a range and bearing sensor, instead of incurring in a $O(n)$ cost, we incur in $O(1)$. For a range-only sensor, we have to consult elements of the occupancy grid in an arc of a circle around the hypothesized vehicle location at the given range. Discretizing this arc in n_θ angles, the cost will be $O(n_\theta)$[1]. In the case of bearing-only sensors, and discretizing the viewing direction in n_r steps, we will have to consult $O(n_r)$ elements. Table 2 contains a summary of the resulting computational complexities.

B. Computational complexity of `Pair_driven`

Algorithm 2, `Pair_driven`, considers each of the m measurements in turn, and then each of the n map features in turn. It hypothesizes the set $\mathbf{X}_{ij}$ of vehicle locations from where feature $\mathbf{f}_j$ would produce measurement $\mathbf{z}_i$. If the s location samples are randomly drawn, $\mathbf{X}$ will be represented a set of location hypotheses with *explicit* coordinates. Again, a first implementation of function `compute_compatible_locations`, would cost $O(s)$; it would sequentially consider each alternative location sample $\mathbf{x}$ in $\mathbf{X}$ to decide whether it is compatible with any of $\mathbf{X}_{ij}$. The total computational complexity of `Pair_driven` would then be $O(m \cdot n \cdot s) = O(m \cdot n \cdot n_x \cdot n_y \cdot n_\phi)$.

Alternatively, if grid sampling is used, samples of the configuration space are drawn in a systematic way, evenly placed in the center of equally-sized *tiles* that fully cover the configuration space and exhibit a lattice structure, and can easily be represented by a location grid. This allows to compute the closest sample to a given location hypothesis, required by function `compute_compatible_locations` in constant time. Thus, using a grid sampling method, `Pair_driven` can be bilinear in both the number of features and the number of measurements, $O(n \cdot m)$. The number of hypothesized locations will be $O(n_\phi)$ for range and bearing sensors, in $O(n_\phi \cdot n_\theta)$ for range-only sensors, and in $O(n_\phi \cdot n_r)$ for bearing-only sensors (see fig. 1). Since sensors are local, n_ϕ and n_r are constants.

[1] In the case of indoor sonar, the span of θ is around $30deg$. In the pure range-only case, the span of θ will be $360deg$. In this case there is no need to sample ϕ in neither algorithm because the vehicle orientation cannot be recovered from the observations.

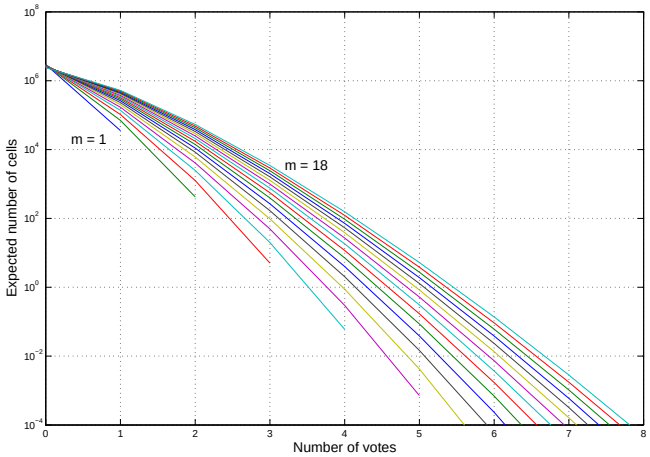


Fig. 3. Number of location hypotheses expected to get a certain number of votes, for $m = 1$ to $m = 18$ observations.

C. `Loc_driven` *.vs.* `Pair_driven`

How do both algorithms compare? Consider the case of range and bearing sensors: the computational cost of `Loc_driven` will be $O(n_x \cdot n_y \cdot n_\phi \cdot m)$, while `Pair_driven` will be $O(n \cdot m \cdot n_\phi)$. Their ratio then will be equal to ρ, the density of features per grid cell unit:

$$\frac{\texttt{Pair\_driven}}{\texttt{Loc\_driven}} = \frac{n \cdot m \cdot n_\phi}{n_x \cdot n_y \cdot n_\phi \cdot m} = \frac{n}{n_x \cdot n_y} = \rho$$

This will be the case for any type of sensor (see table 2). This means that the computational cost of algorithm `Pair_driven` will be a fraction of the cost of `Loc_driven` proportional to feature density. This makes it especially efficient in sparse environments, like Victoria Park, Sydney, where in an area of around $197m \times 93m$ there are 99 trees. If you decide to discretize the configuration space every $1.5m$, you will obtain 132×63, 8316 grid elements in position. If $n = 99$ then you can expect `Pair_driven` to run about 82 times faster.

D. A probabilistic analysis of the robustness of voting strategies

Global localization problems in fact pose a twofold question: (1) is the vehicle in the map? (2) if so, where? Both `Loc_driven` and `Pair_driven` compute the number of observation-feature pairings for each element in the set of location samples, so we further need to decide when the hypothesis with the highest number of votes can be accepted and thus the vehicle can be considered to be in the map. In [8], an empirical threshold $t = 6$ of six matchings was used to prevent false positives for the Victoria Park dataset.

In the following, we carry out a probabilistic analysis of voting strategies, that allows to compute the probability of accepting a location hypothesis formed with matchings occurring at random.

Consider the `Pair_driven` algorithm in the case of range and bearing. Each observation casts $n \cdot n_\phi$ votes, that are to be distributed among some of the $n_x \cdot n_y \cdot n_\phi$ grid cells, or candidate locations. Assuming that features are randomly distributed in the environment, the probability that a given candidate randomly gets a vote from an observation is equal to the density of features in the map:

$$\frac{n \cdot n_\phi}{n_x \cdot n_y \cdot n_\phi} = \rho$$

Thus, the probability that a given candidate is randomly voted by k of the m observations follows the binomial distribution [12] as follows:

$$p_m(k) = \frac{k}{k!(m-k)!}\, \rho^k\, (1-\rho)^{m-k}$$

As a result, the expected number of candidates in the configuration space having k votes will be:

$$r_{k,m} = n_x \cdot n_y \cdot n_\phi \cdot p_m(k)$$

Similar results can be obtained by considering the `Loc_driven` algorithm, as well as other sensors.

This analysis can be used to derive an adequate threshold for the acceptance of a location hypothesis. For the Victoria Park dataset, figure 3 shows, for different values of m, the number of cells expected to get k of the m votes. We can see that a fixed $t = 6$ pairings criteria is not uniformly restrictive: for a small number of observations, say $m = 6$, the expected number of cells with 6 random pairings is $8.5220e{-}6$, but for $m = 18$, this expected number climbs up to 0.1370.

We can limit the probability of accepting random false positives by setting a fix bound to $r_{k,m}$. For example, in order to have $r_{k,m} \leq 10^{-2}$ random false positives, for $m = 5$ the threshold t should be set to 5 , while for $m = 18$, t should be set to 7.

IV. Experiments

An experimental comparison of both algorithms previously detailed is carried out using the dataset obtained by Guivant and Nebot [11]. We used 2500 steps of the trajectory of an outdoor vehicle equipped with a laser sensor along Victoria Park, Sydney. Point features, corresponding to trees are segmented from the scan using the `find_trees` algorithm [11]. A stochastic map of $n = 99$ point features was generated with the first 1000 steps (fig. 4). The remaining steps (1001 to 2500) were used in the relocation algorithms, `Loc_driven` and `Pair_driven`. In this way, the statistical independence between the scans and the stochastic map is guaranteed. To verify the vehicle locations calculated by our algorithms, we obtained a reference solution running continuous SLAM until step 2500. The number of observations gathered from each position of the vehicle ranges between $m = 3$ and $m = 18$. Figure 5 shows the segmented trees for a scan of $m = 16$ measurements corresponding to step 1888.

Both algorithms are executed using a grid sampled configuration space of resolution $1.5m$ for x and y, and $1deg$ for ϕ. Fig. 6 shows a summarized table of votes corresponding

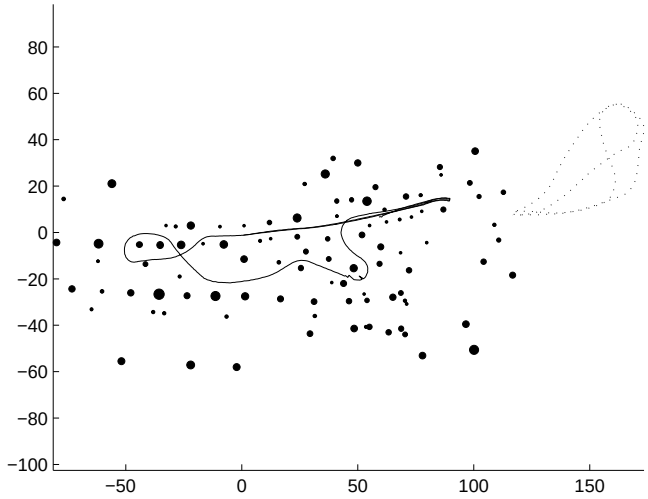


Fig. 4. Stochastic map of 2D points built until step 1000 with n = 99 features. Reference vehicle trajectory inside the map (solid line) and outside of map (dashed line). Tree radiuses × 5; distances are in meters.

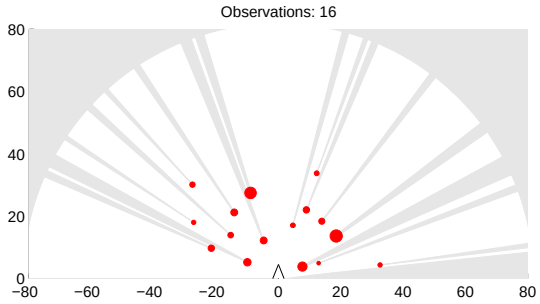


Fig. 5. Segmented trees by the algorithm at step 1888. Tree radiuses × 5; distances are in meters.

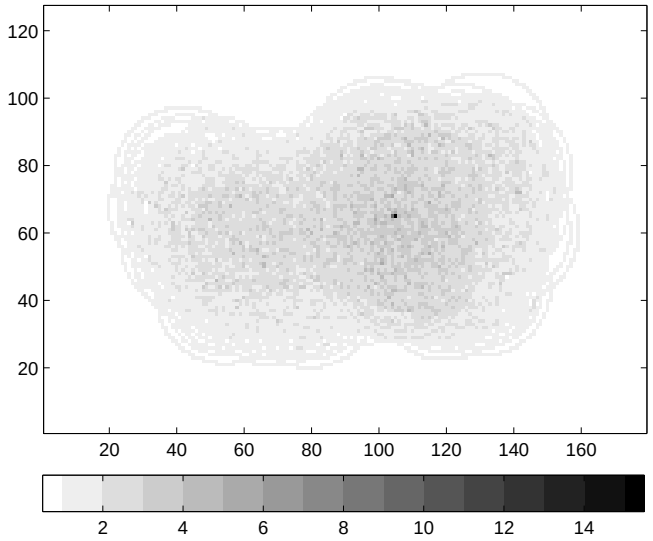


Fig. 6. Cumulative voting table for step 1888 (m = 16) with dark pixel representing an unique hypothesis solution with votes = 16.

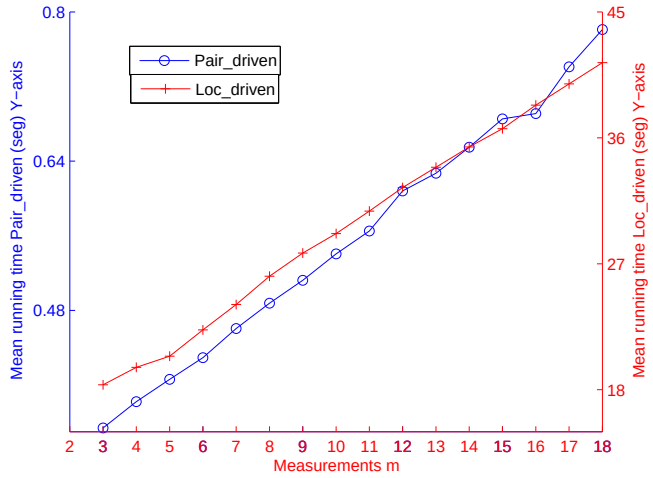


Fig. 7. Mean running time of each algorithm versus the number of measurements.

to the result of executing `Pair_driven` at step 1888. For each grid cell in position, the corresponding image pixel depicts the maximum number of votes of all orientations corresponding to that position. For this example there is only one cell solution with 16 votes, represented by a black pixel in the image, and only one cell with 7 votes, a neighbor of the most voted cell (a predictable tessellation effect). The remaining cells contain less than 6 votes, as predicted by our probabilistic analysis (fig. 3).

In order to compare the computational complexity of `Loc_driven` and `Pair_driven` we have executed the algorithms for all the test steps. `Loc_driven` has been implemented representing the map using an occupancy grid. Figure 7 shows the mean running time of each algorithm versus the number of measurements. Both algorithms were implemented in MATLAB, and executed on a Pentium IV, at 2.8GHz. Algorithm `Pair_driven` is faster than `Loc_driven` by

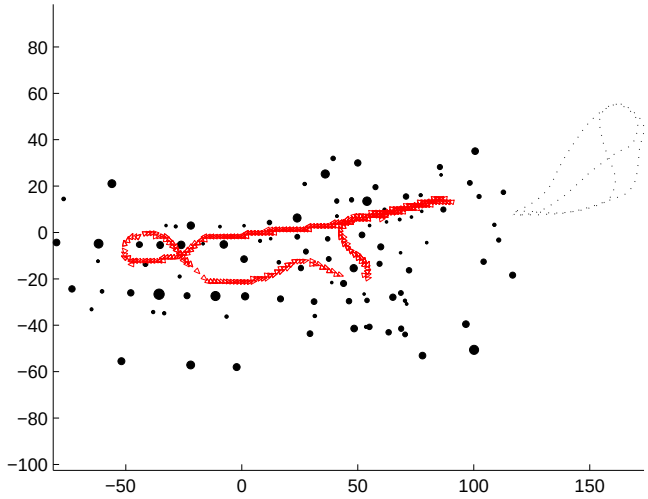


Fig. 8. True positive solutions: 565 out of 737 steps. Missing steps (false negatives) are due to insufficient number of measurements, or insufficient number of pairings.

a factor of around 60. The differences of this result with a predicted 80 times gain in efficiency for this data are probably due to slightly different implementations of the operations in each algorithm, as well as memory management issues in MATLAB array operations.

In [8] it was shown that, for the same Victoria Park dataset, with less than 6 pairings there is insufficient evidence to assert that the vehicle is within map limits without having false positives. In order to compare the results of this work with the RS algorithm in [8], we first consider the criteria of $t = 6$ pairings as a threshold for accepting the solution of the algorithms.

Of the 1500 test steps, we consider 737 steps within map limits (see fig. 4, continuous line). In 74 steps of those (10.1%), the number of segmented trees was less than six. In 98 steps (13.3%) there are six or more measurements, but our algorithm finds less than six pairings. Thus, there is a total of 172 false negatives (23.4%). In the 565 remaining cases (76.6%), the algorithm found six or more pairings, and the solution obtained was always consistent with the reference solution, without false positives (fig. 8). These results show that the `Pair_driven` algorithm is slightly more prone to *false negatives* than the RS algorithm reported in [8]. In that work, in 604 cases (82%), RS finds six or more pairings, in 39 more cases. The difference is expected, since it is well known that voting algorithms using strategies of the type of the Hough Transform are sensitive to tessellation effects [13]. In some cases, some of the votes for the correct solution may fall in bordering cells if the solution is close to the border. Further work will be necessary to refine the determination of the resolution of the grid.

In these results, the false negative rate is rather high. The reason is that, in steps in which there are less than 6 observations, the answer is always negative, because of insufficient data. We can improve the results if we use the

variable threshold described in subsection III-D. For steps with $m = 4$ observations, we consider a threshold $t_4 = 4$; the expected number of random hypotheses will be 0.0601. For larger values of m, we will set t_m so that $r_{k,m}$, the expected number of false cells, will be less than 10^{-2}: $t_m = 5$ for $m = 5 \ldots 6$, $t_m = 6$ for $m = 7 \ldots 12$, and $t_m = 7$ for $m = 13 \ldots 18$. Using this threshold, we obtain true positive solutions in 609 cases, around 82.6%. We still consider steps with $m = 1 \ldots 3$ observations as having insufficient information, 22 cases.

We also tested the 580 steps in which the vehicle is outside the map limits (fig. 4, dashed line). Using the new threshold, there are no false positive answers.

V. Discussion

In this paper we have shown that the global localization problem in SLAM can be solved in time linear with both the size of the map and the number of sensor measurements. The representation of the vehicle configuration space via grid sampling allows to use a more efficient pairing-driven voting strategy than the location-driven voting strategy that Monte Carlo style algorithms use. Recent works [14] suggest that, for motion planning, there is no advantage in the use of random sampling techniques. In this paper we point out an advantage in the use of grid sampling for global localization in SLAM.

Surprisingly, in contrast to most of the global localization literature, in our experiments we are able to localize the vehicle in *one-shot*. The reasons are that the environment is sparse, without symmetries, and the laser scanner is very precise. More complex scenarios will probably require to move the vehicle in order to acquire sufficient information for localization.

Incorporating vehicle motion to consider measurements obtained at more than one location can be easily considered in the algorithm, and thus constitutes immediate future work. Additional future work will also include considering incremental sampling strategies [10], that will allow to incorporate the decision on the grid resolution into the voting algorithm. This will allow the algorithm to quickly identify the most promising regions of the configuration space using low resolution, and then concentrating on these regions using high resolution, further improving the computational cost of global localization.

We feel that this technique may also be useful as a bootstrapping step for Monte Carlo methods. With no prior information on vehicle location in a very large environment, our algorithm could be used to determine promising areas within the environment. A Monte Carlo algorithm could then take control focusing on those areas with a smaller number of random particles.

Acknowledgment

This research has been funded in part by the Dirección General de Investigación of Spain under project DPI2003-07986.

References

[1] D. Fox, W. Burgard, D. F., and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," in *Proc. of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, 1999.

[2] D. Fox, W. Burgard, and S. Thrun, "Active markov localization for mobile robots," *Robotics and Autonomous Systems*, vol. 25, no. 3-4, pp. 195–207, 1998.

[3] W. Burgard, D. Fox, D. Hennig, and T. Schmidt, "Estimating the absolute position of a mobile robot using position probability grids," in *Proc. of the Fourteenth National Conference on Artificial Intelligence (AAAi-96)*, 1996.

[4] W. E. L. Grimson, *Object Recognition by Computer: The Role of Geometric Constraints*. Cambridge, Mass.: The MIT Press, 1990.

[5] J. H. Lim and J. J. Leonard, "Mobile robot relocation from echolocation constraints," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 9, pp. 1035–1041, September 2000.

[6] J. A. Castellanos and J. D. Tardós, *Mobile Robot Localization and Map Building: A Multisensor Fusion Approach*. Boston, Mass.: Kluwer Academic Publishers, 1999.

[7] T. Bailey, E. M. Nebot, J. K. Rosenblatt, and H. F. Durrant-Whyte, "Data association for mobile robot navigation: A graph theoretic approach," in *IEEE Int. Conf. Robotics and Automation*, San Francisco, California, 2000, pp. 2512–2517.

[8] J. Neira, J. D. Tardós, and J. A. Castellanos, "Linear time vehicle relocation in SLAM," in *IEEE Int. Conf. on Robotics and Automation*, Taipei, Taiwan, September 2003, pp. 427–433.

[9] D. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[10] S. R. Lindemann, A. Yershova, and S. M. LaValle, "Incremental grid sampling strategies in robotics," in *Workshop on Algorithmic Foundations of Robotics (WAFR-2004)*, 2004.

[11] J. E. Guivant, F. R. Masson, and E. M. Nebot, "Simultaneous localization and map building using natural features and absolute information," *Robotics and Autonomous Systems*, vol. 40, pp. 79–90, 2002.

[12] A. Papoulis and S. Pillai, *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, 2002.

[13] W. E. L. Grimson and D. P. Huttenlocher, "On the sensitivity of the hough transform for object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 3, pp. 255–274, 1990.

[14] S. M. LaValle, M. S. Braincky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *Int. J. of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.