Proceedings of the 2003 IEEE
International Conference on Robotics & Automation
Taipei, Taiwan, September 14-19, 2003

Linear time vehicle relocation in SLAM

José Neira, Juan D. Tardós, José A. Castellanos
Dept. Informática e Ingeniería de Sistemas
Universidad de Zaragoza
c/María de Luna 1, 50018 Zaragoza, Spain
{jneira, tardos, jacaste}@unizar.es

Abstract— **In this paper we propose an algorithm to determine the location of a vehicle in an environment represented by a stochastic map, given a set of environment measurements obtained by a sensor mounted on the vehicle. We show that the combined use of (1) geometric constraints considering feature correlation, (2) joint compatibility, (3) random sampling and (4) locality, make this algorithm linear with both the size of the stochastic map and the number of measurements. We demonstrate the practicality and robustness of our approach with experiments in an outdoor environment.**

I. Introduction

The objective of simultaneous localization and mapping (SLAM) is to use the information obtained by sensors mounted on a vehicle to build and update a map of the environment and compute the vehicle location in that map. The most critical point in obtaining a robust SLAM solution is data association, i.e. relating sensor measurements with the elements included in the map. There are two basic approaches to address the data association problem:

- Search in *pose space*: where a set of candidate vehicle locations is generated and analyzed looking for consistency between sensor measurements and the previous map. This idea, that can be used with raw sensor data, is the base of the Monte Carlo localization approach to SLAM [16].
- Search in *correspondence space*: where sensor measurements are processed to obtain discrete features (points, lines, etc.) that are matched against the features stored in the map. This is the approach used in all feature based approaches to SLAM [4], [6], [8].

During continuous SLAM, the uncertainty in the location of the map elements relative to the vehicle is usually small, and both techniques have proven able to solve the problem. In this work, we concentrate on the more difficult vehicle relocation problem, also known as first-location, global localization, or "kidnapped" robot problem. It can be stated as follows:

given a vehicle in an unknown location, and a map of the environment, use a set of measurements taken by onboard sensors to determine the vehicle location within the map.

Several works have addressed the problem using an *a priori* map of the environment (see [5] for a review). In SLAM, solving this problem is essential to be able to re-start the robot in a previously learned environment, to recover from localization errors, or to safely close big loops.

Monte Carlo localization addresses the problem by simply generating vehicle pose hypotheses covering all possible locations and computing the likelihood of each pose by looking for consistency with the map. However, results about the computing time required and the speed of convergence of such an extensive technique have not been reported.

On the other hand, feature based approaches to SLAM usually rely on the gated nearest neighbor (NN) algorithm, that can only solve data association when a good vehicle estimation exists. Our Joint Compatibility technique [14] is much more robust, but it is still restricted to some meters and less than 30 degrees of vehicle error. When there is no vehicle estimation, previous work on object recognition suggests that simple geometric constraints can be used to limit the complexity of searching the correspondence space [10]. Recent implementations of this idea include the work of Bailey et al. [1] using graph theory, the work of Lim and Leonard [13] using a hypothesize and test technique, and our own work [5] using Grimson's interpretation tree [10].

In this paper we further elaborate on the interpretation tree approach, presenting in section II alternative algorithms able to solve the relocation problem. We introduce in section III the idea of *locality* that allows to obtain search algorithms linear in time with the size of the map. Finally, we provide experimental results comparing the different algorithms, using the datasets obtained by Guivant and Nebot [11] with an outdoor vehicle. The winner is a novel algorithm that, combining random sampling and hypothesis verification, solves the relocation problem in a very robust and efficient way.

II. Relocation in SLAM

In classical stochastic mapping, the environment information related to a set of elements $\mathcal{F} = \{B, F_0, F_1, \ldots, F_n\}$ is represented by a global map $\mathcal{M}^B_{\mathcal{F}} = (\hat{\mathbf{x}}^B_{\mathcal{F}}, \mathbf{P}^B_{\mathcal{F}})$, where:

$$\hat{\mathbf{x}}^B_{\mathcal{F}} = \begin{bmatrix} \hat{\mathbf{x}}^B_{F_0} \\ \vdots \\ \hat{\mathbf{x}}^B_{F_n} \end{bmatrix} ; \; \mathbf{P}^B_{\mathcal{F}} = \begin{bmatrix} \mathbf{P}^B_{F_0F_0} & \cdots & \mathbf{P}^B_{F_0F_n} \\ \vdots & \ddots & \vdots \\ \mathbf{P}^B_{F_nF_0} & \cdots & \mathbf{P}^B_{F_nF_n} \end{bmatrix} \quad (1)$$

The state vector $\hat{\mathbf{x}}^B_{\mathcal{F}}$ contains the estimated location of the vehicle F_0 and of n environment features $F_1 \ldots F_n$, all with respect to a base reference B. Matrix $\mathbf{P}^B_{\mathcal{F}}$ is the estimated error covariance of $\hat{\mathbf{x}}^B_{\mathcal{F}}$. In the case of the vehicle, its location vector $\hat{\mathbf{x}}^B_{F_0}$ describes the transformation from B to F_0. In the case of an environment feature F_j, the parameters that compose its location vector $\hat{\mathbf{x}}^B_{F_j}$ depend on the feature type.

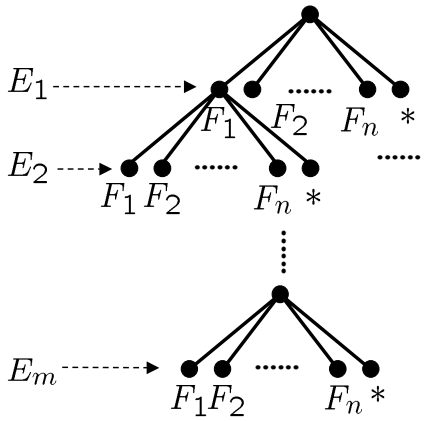

Fig. 1. Interpretation tree of measurements $E_1 \cdots E_m$ in terms of map features $F_1 \cdots F_n$.

Given a new set of m measurements $\hat{\mathbf{z}} = \{\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_m\}$ of the environment features $\mathcal{E} = \{E_1 \ldots E_m\}$ obtained by a sensor mounted on the vehicle, with $\mathbf{R}$ being the estimated error covariance of $\mathbf{z}$, the purpose of data association is to produce a hypothesis:

$$\texttt{H = [j}_1 \ \texttt{j}_2 \ \cdots \ \texttt{j}_m\texttt{]}$$

associating each measurement E_i with its corresponding map feature F_{j_i} ($\texttt{j}_i = 0$ indicates that $\hat{\mathbf{z}}_i$ is considered spurious). This hypothesis can be used to determine or refine the vehicle and environment feature locations.

The space of measurement-feature correspondences can be represented by an *interpretation tree* of m levels [10] (see fig. 1). Each node of the tree at level i has $n+1$ branches, corresponding to the n alternative feature pairings for the measurement E_i, and an extra branch (star-branch) to account for the measurement being spurious. The size of this correspondence space, (i.e. the number of alternative hypotheses) is exponential with the number of measurements, $N_h = (n+1)^m$.

Next we describe several algorithms to perform data association by searching in the interpretation tree for the most plausible hypothesis in a depth-first branch and bound manner. Each algorithm implements a different technique to validate the set of pairings in a hypothesis.

A. Geometric Constraints

Given no estimation of the vehicle location, *location independent* constraints must be used to efficiently traverse the interpretation tree in search for the best hypothesis. Grimson [10] proposed a branch and bound algorithm for model based geometric object recognition that uses unary and binary geometric constraints.

Given a pairing $p_{ij} = (E_i, F_j)$, let $\mathbf{u}_i = (\hat{\mathbf{u}}_i, \mathbf{P}_i)$ and $\mathbf{u}_j = (\hat{\mathbf{u}}_j, \mathbf{P}_j)$ be d-dimensional stochastic parameter vectors (e.g. length for segments, angle for corners, or radius for circular features) related to the measurement E_i and the feature F_j respectively. Assuming independence between measurements and the stochastic map, the unary constraint is satisfied when:

$$\begin{aligned} D^2_{ij} &= (\hat{\mathbf{u}}_i - \hat{\mathbf{u}}_j)^T (\mathbf{P}_i + \mathbf{P}_j)^{-1} (\hat{\mathbf{u}}_i - \hat{\mathbf{u}}_j) \\ &< \chi^2_{d,\alpha} \end{aligned} \tag{2}$$

Given two pairings $p_{ij} = (E_i, F_j)$ and $p_{kl} = (E_k, F_l)$, a binary geometric constraint is a geometric relation between measurements E_i and E_k that must also be satisfied between their corresponding map features F_j and F_l (e.g., distance between two points, angle between two segments). In the case of map features, let $\mathbf{f}_{jl}$ be a d-dimensional function to compute a geometric relation between features F_j and F_l:

$$\mathbf{b}_{jl} = \mathbf{f}_{jl}\left(\mathbf{x}^B_{\mathcal{F}}\right) \tag{3}$$

with estimated mean and covariance:

$$\begin{aligned} \hat{\mathbf{b}}_{jl} &= \mathbf{f}_{jl}\left(\hat{\mathbf{x}}^B_{\mathcal{F}}\right) \\ \mathbf{P}_{jl} &= \mathbf{J}_j \, \mathbf{P}^B_{F_j F_j} \, \mathbf{J}_j^T + \mathbf{J}_l \, \mathbf{P}^B_{F_l F_l} \, \mathbf{J}_l^T \\ &+ \mathbf{J}_j \, \mathbf{P}^B_{F_j F_l} \, \mathbf{J}_l^T + \mathbf{J}_l \, \mathbf{P}^B_{F_l F_j} \, \mathbf{J}_j^T \\ \mathbf{J}_j = \left.\frac{\partial \mathbf{f}_{jl}}{\partial \mathbf{x}^B_{F_j}}\right|_{\hat{\mathbf{x}}^B_{\mathcal{F}}} &\quad ; \quad \mathbf{J}_l = \left.\frac{\partial \mathbf{f}_{jl}}{\partial \mathbf{x}^B_{F_l}}\right|_{\hat{\mathbf{x}}^B_{\mathcal{F}}} \end{aligned}$$

The influence of the correlations between the features in the computation of the geometric relation is important in the case where the features are distant from the map base reference. Their estimated location may be very imprecise, but their correlation will allow to estimate their geometric relations with sensor precision. Similar equations correspond to measurements E_i and E_k. Assuming measurements independent from map features, the binary constraint is satisfied when:

$$D^2_{ikjl} = (\hat{\mathbf{b}}_{ik} - \hat{\mathbf{b}}_{jl})^T (\mathbf{P}_{ik} + \mathbf{P}_{jl})^{-1} (\hat{\mathbf{b}}_{ik} - \hat{\mathbf{b}}_{jl}) < \chi^2_{d,\alpha}$$

Figure 2 describes the recursive branch and bound algorithm (`GCBB`) which computes the best hypothesis from the available measurements and the features of the stochastic map using unary and binary location independent geometric constraints. Starting with an empty hypothesis, the algorithm proceeds in a depth-first branch and bound manner. At the leaf-level of the interpretation tree, the algorithm checks whether it has come up with a hypothesis having more consistent pairings than the current best. Given that satisfying binary geometric constraints does not guarantee that a hypothesis is globally consistent [10], the vehicle location is estimated and the joint compatibility test [14] is used to verify global consistency of the matching hypothesis. Note that the algorithm performs a bounded search, it evaluates the potential benefit of considering the star-branch before it recurs. Branching can be done by pre-ordering the measurements so that, for example, the most precise (usually closer to the sensor) are considered first. Both unary and binary constraints, being location independent, can be pre-computed before recursion starts.

```
procedure relocation_GCBB:
Best = []
GCBB([], 1)
return Best

procedure GCBB (H, i):
-- H : current hypothesis
-- i : observation to be matched
if i > m -- leaf node?
   if pairings(H) > pairings(Best) -- did better?
      estimate_location_(H)
      if joint_compatibility(H)
         Best = H
      fi
   fi
else
   for j in {1...n}
      if unary(i, j) ∧ binary(i, j, H)
         GCBB([H j], i + 1) -- (E_i, F_j) accepted
      fi
   rof
   if pairings(H) + m - i > pairings(Best)
      GCBB([H 0], i + 1) -- try star node
   fi
fi
```

Fig. 2. Geometric Constraints Branch and Bound

```
procedure relocation_GV:
Best = []
GV([], 1)
return Best

procedure GV (H, i):
-- H : current hypothesis
-- i : observation to be matched
if i > m
   if pairings(H) > pairings(Best)
      Best = H
   fi
elseif pairings(H) == 3
   estimate_location_(H)
   if joint_compatibility(H)
      JCBB(H, i) -- hypohtesis verification
   fi
else
   for j in {1...n}
      if unary(i, j) ∧ binary(i, j, H)
         GV([H j], i + 1)
      fi
   rof
   if pairings(H) + m - i > pairings(Best)
      GV([H 0], i + 1)
   fi
fi
```

Fig. 3. Generation-verification

B. Maximum clique

A closely related technique also used in object recognition consists in building a compatibility graph whose nodes are unary compatible matchings and whose arcs represent pairs of binary compatible matchings. Finding the largest hypothesis consistent with unary and binary constraints is equivalent to finding the maximum clique in the compatibility graph (see [10] for a discussion and references). This idea has been applied recently by Bailey et al. [1] to the problem of robot relocation with an *a priori* map.

To compare with GCBB, we have also implemented a clique search algorithm, MAXCLI. Given that the compatibility graph is very sparse, we have chosen to implement the branch and bound clique algorithm of Carraghan and Pardalos [3] considered one of the best for sparse graphs (it is also three to four times faster than the clique algorithm used in [1]). We have found that its pruning of the correspondence space is more effective, and it can be programmed very efficiently in MATLAB using sparse matrices, giving as result a faster algorithm than GCBB.

C. Generation-verification

An alternative group of algorithms, originally developed for geometric object recognition [2], [7], follow a hypothesis generation-verification scheme. Based on this idea, we have developed algorithm GV (fig. 3). The generation of candidate hypotheses is carried out using location independent unary and binary geometric constraints, just as in GCBB. But as soon as the number of pairings is adequate to determine vehicle location (i.e. two points, two non-parallel segments), the hypothesis verification step takes place, predicting the location of map features and searching for pairings with the remaining measurements. For this purpose, we use the Joint Compatibility Branch and Bound (JCBB) algorithm (see fig. 4), proven to be very robust when an estimated vehicle location is available [14].

The justification of this approach lies in the fact that joint compatibility is a tighter consistency criterion than geometric constraints, and thus branch pruning is more effective. The potential drawback of this approach is that hypothesis verification is *location dependent*, and thus, the constraints to be used for validation cannot be precomputed. To limit the amount of location dependent constraints to apply, in our implementation verification takes place when a hypothesis contains at least *three* consistent pairings. The third pairing, being redundant, reduces the amount of incorrect hypothesis that arrive at the verification stage.

D. Random sampling

Branch and bound algorithms are forced to traverse the whole correspondence space until a good bound is found. In the SLAM relocation problem, when the vehicle is not within the mapped area, a good bound is never found. Since the correspondence space is exponential with the number of measurements, in this worst case the execution times of GCBB, MAXCLI and GV are very long.

```
procedure JCBB (H, i):
-- H : current hypothesis
-- i : observation to be matched
if i > m
   if pairings(H) > pairings(Best)
      Best = H
   fi
else
   for j in {1...n}
      if unary(i, j)
            ∧ joint_compatibility([H j])
         JCBB([H j], i + 1)
      fi
   rof
   if pairings(H) + m - i > pairings(Best)
      JCBB([H 0], i + 1)
   fi
fi
```

Fig. 4. Joint Compatibility Branch and Bound

```
procedure relocation_RS:
P_fail = 0.05, p = 3, P_g = 0.5
Best = []
i = 0
repeat
   ẑ = random_permutation(ẑ)
   RS([], 1)
   P_g = max(P_g, pairings(Best) / m)
   t = log P_fail / log (1 − P_g^p)
   i = i + 1
until i >= t
return Best

procedure RS (H, i):
-- H : current hypothesis
-- i : observation to be matched
if i > m
   if pairings(H) > pairings(Best)
      Best = H
   fi
elseif pairings(H) == 3
   estimate_location_(H)
   if joint_compatibility(H)
      JCBB(H, i) -- hypohtesis verification
   fi
else -- branch and bound without star node
   for j in {1...n}
      if unary(i, j) ∧ binary(i, j, H)
         RS([H j], i + 1)
      fi
   rof
fi
```

Fig. 5. Random Sampling

This fact led us to consider a relocation algorithm where the hypothesis generation process is done using random sampling (RS) instead of by full traversal of the interpretation tree. Our RS algorithm (fig. 5) is an adaptation of the RANSAC algorithm [9] for the relocation problem. The fundamental idea is to randomly select p of the m measurements to try to generate vehicle localization hypotheses, and verify them with all m measurements using joint compatibility. If P_g is the probability that a randomly selected measurement corresponds to a mapped feature (not spurious) and P_{fail} is the acceptable probability of not finding a good solution when it exists, the required number of tries is:

$$t = \left\lceil \frac{\log P_{fail}}{\log\left(1 - {P_g}^p\right)} \right\rceil \tag{4}$$

For the reason explained above, we use $p = 3$. Choosing $P_{fail} = 0.05$ and considering *a priori* that only half of the measurements are present in the map $P_g = 0.5$, the maximum number of tries is $t = 23$. If you can consider that at least 90% of the measurements correspond to a map feature, the number of required tries is only 3. The RS algorithm randomly permutes the measurements and performs hypothesis generation considering the first three measurements not spurious (without star branch). The number of tries is re-calculated to adapt to the current best hypothesis, so that no unnecessary tries are carried out [12].

Notice that the maximum number of tries does *not* depend on the number of measurements. Experiments will show that this fact is crucial in reducing the computational complexity of the RS algorithm.

III. Locality

As it has been explained above, the main problem of the interpretation tree approach is the exponential number of possible hypotheses (tree leaves): $N_h = (n+1)^m$. The use of geometric constraints and branch and bound search dramatically reduces the number of nodes explored, by cutting down entire branches of the tree. However, Grimson [10] has shown that in the general case, where spurious measurements can arise, the amount of search needed to find the best interpretation is still exponential. In these conditions, the interpretation tree approach seems impracticable except for very small maps.

To overcome this difficulty we introduce the concept of *locality*: given that the set of measurements has been obtained from a unique vehicle location (or from a set of nearby locations), it is sufficient to try to find matchings with *local* sets of features in the map. Given a map feature F_j, we define its *locality* $L(F_j)$ as the set of map features that are *close enough* to it, such that they could be seen from the same vehicle location. For a given mapping problem, the maximum cardinality of the locality sets will be a constant c that depends on the sensor range and the maximum density of features in the environment.

During the interpretation tree search, once a matching has been established with a map feature, the search can

be restricted to its locality set. For the first measurement, there are n possible feature matchings. Since there are at most c features covisible with the first one, for the remaining $m-1$ measurements there are only c possible matches, giving a maximum of $n(c+1)^{m-1}$ hypotheses. If the first measurement is not matched, a similar analysis can be done for the second tree level. Thus, the total number of hypotheses N_h will be:

$$\begin{aligned} N_h &\leq n(c+1)^{m-1} + n(c+1)^{m-2} + \cdots + n + 1 \\ &= n\frac{(c+1)^m - 1}{c} + 1 \end{aligned}$$

This implies that, using locality, the complexity of searching the interpretation tree will be *linear* with the size of the map.

There are several ways of implementing locallity:

- SLAM can be implemented by building sequences of independent local maps [15]. If the local maps are stored, the search for matchings can be performed in time linear with the number of local maps. In this case, the locality of a feature is the set of features belonging to the same local map. A drawback of this technique is that global localization may fail around the borders between two local maps.
- Alternatively, the locality of a feature can be computed as the set of map features within a distance less than the maximum sensor range. There are two drawbacks in this approach: first, this will require $O(n^2)$ distance computations, and second, in some cases features that are close cannot be seen simultaneously (for example because they are in different rooms), and thus should not be considered local.
- For these reasons, we have chosen to obtain the locality of a feature as the set of features that have been seen simultaneously with it at least once, which can be done during map building without extra cost.

Figure 6 (top) shows the covisibility matrix obtained during map building for the first 1000 steps of the Victoria Park dataset used in the experiments. As features are added incrementally during map building, the typical form of the covisibility matrix is band diagonal. Elements far from the diagonal appear when a loop is closed, because recently added features become covisible with previously mapped features. In any case, the number of elements per row or column only depends on the density of features and the sensor reach. Using a sparse matrix representation, the amount of memory needed to store the covisibility matrix (or any other locality matrix) is $O(n)$.

An important property is that the covisibility matrix is closely related to the information matrix of the map (the inverse of the map covariance matrix). Figure 6 (bottom) shows the normalized information matrix, where each row and column has been divided by the square root of the corresponding diagonal element. It is clear that the information matrix allows to determine which features have been seen form the vehicle location during map building. The

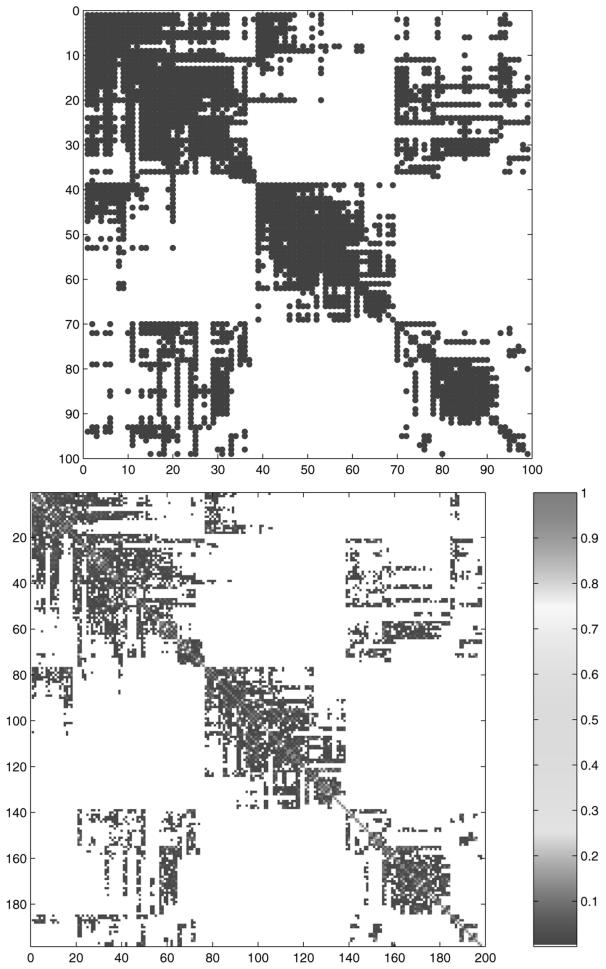

Fig. 6. Covisibility matrix and normalized information matrix

intuitive explanation is that as the uncertainty in the absolute vehicle location grows, the information about the features that are seen from the same location becomes highly coupled.

This gives further insight on the structure of the SLAM problem: while the map covariance matrix is a full matrix with $O(n^2)$ elements, the normalized information matrix tends to be sparse, with $O(n)$ elements. We believe that this fact can be used to obtain more efficient SLAM algorithms. Some preliminary results in this line have been obtained by Thrun [17].

IV. Experiments

To compare the relocation algorithms described in section II, we have used the dataset obtained by Guivant and Nebot [11], using a vehicle equipped with a SICK laser scanner in Victoria Park, Sydney. Wheel encoders give an odometric measure of the vehicle location. The laser scans are processed using Guivant's algorithm to detect tree trunks and estimate their radiuses (fig. 7).

We ran continuous SLAM until step 1000, obtaining a

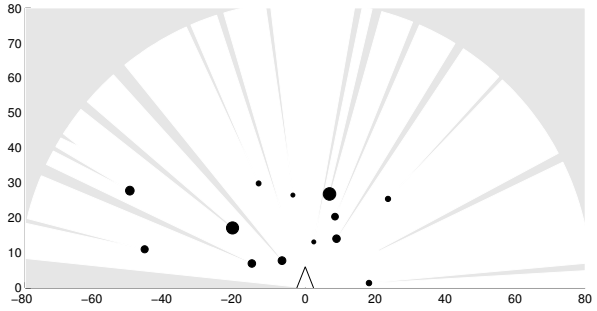

Fig. 7. Segmentation of scan 120, with $m = 13$ tree trunks detected. Radiuses are magnified $\times$ 5.

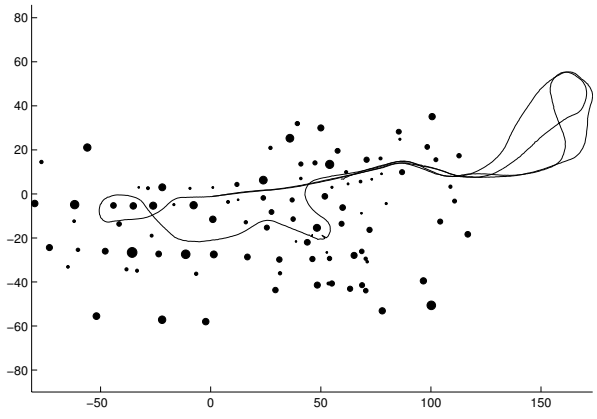

Fig. 8. Stochastic map of 2D points built until step 1000. There are $n = 99$ features. Reference vehicle trajectory for steps 1001 to 2500. Radiuses are magnified $\times$ 5.

map of $n = 99$ point features (see fig. 8). The relocation algorithms GCBB, GV, MAXCLI and RS were executed on scans 1001 to 2500. This guarantees that we use scans statistically independent from the stochastic map. The radiuses of the trunks are used as unary constraints, and the distance between the centers as binary constraints. To verify the vehicle locations obtained by our algorithms, we obtained a reference solution running continuous SLAM until step 2500. Fig. 8 shows the reference vehicle location for steps 1001 to 2500.

No significant difference was detected in the solutions obtained by the four algorithms. They agree most of the time because the techniques that they use to validate a hypothesis are substantially equivalent. In this experiment, when six or more measurements are paired, the algorithms return the true vehicle location, with no false positives. Otherwise, the solution must be discarded as being non reliable.

In the experiment, of the 1500 steps of the trajectory considered, we have selected 737 steps where the vehicle was within the map limits. In 604 (82%) cases, the algorithms found six or more pairings, and the location obtained was consistent with the reference solution (within 2σ bounds). In the remaining cases, either less than six points were segmented from the scan, (74 case, 10%), or the algorithms

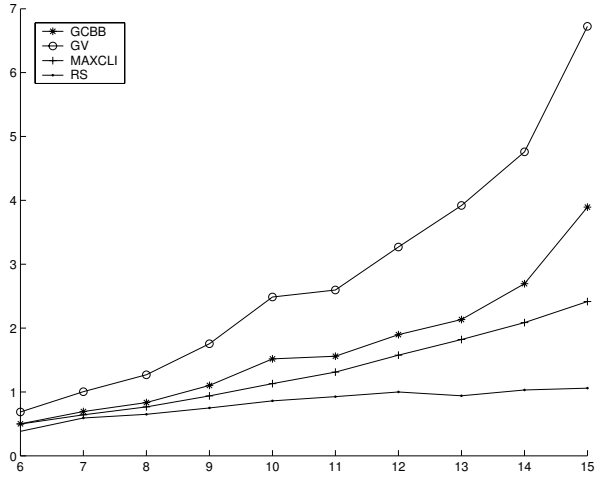

Fig. 9. Mean execution time .vs. m (number of measurements) for steps 1001 to 2500 of the vehicle trajectory.

could not find six matchings (59 cases, 8%). In these cases, more sensor information is necessary to reliably determine the vehicle location.

We found a significant difference in efficiency. The algorithms where implemented in MATLAB, and executed on a Pentium IV, at 1.7GHz. Figure 9 shows the mean running time of each algorithm versus the number of measurements in cases where the relocation was successful. GV turns out to be the less efficient. This is due to the fact that hypotheses must be verified using location dependent constraints, which cannot be pre-computed. The best algorithm is RS, with mean execution time of less than $1s$.

The worst case for all algorithms happens when the vehicle is not in the mapped area. In this case a good bound cannot be found, and the whole correspondence space must be explored. To compare the worst case performance of each algorithm, we generated random measurements for $m = 3 \ldots 30$, 100 times for each value of m, and executed the algorithms. Only MAXCLI and RS are considered, because both GCBB and GV are highly exponential on m, and their performance is extremely poor. Figure 10 shows the mean time of both algorithms versus m. In this experiment, the running time of MAXCLI appears to grow quadratically with m. This is because the dominant calculation is the determination of binary constraints, which is $O(cnm^2)$. In the case of RS, time grows steeply until $m = 7$. In these cases, all combinations of three measurements selected from m are tried, which is less than the maximum number of tries $t = 23$ given by eq. (4). After that, RS appears to grow linearly with m, because the dominant calculation is the individual innovation test of the hypothesis verification step, which is $O(cm)$. In both algorithms the search part is an exponential component, but it accounts for a very small part of the total running time for these values of m. RS shows to be superior to MAXCLI for more than 15 measurements.

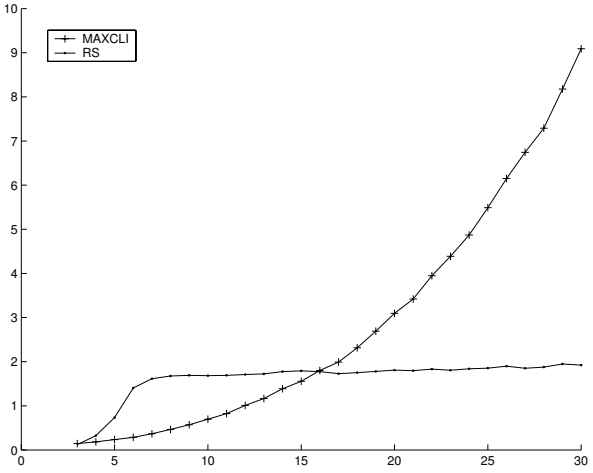

Fig. 10. Mean execution time .vs. m for random measurements

V. Conclusions

In this paper we have proposed an algorithm that determines the location of a vehicle in a stochastic map, given a set of measurements obtained by an onboard sensor. This algorithm follows a generation-verification scheme, whose main novelties are: (1) hypothesis generation is carried out using geometric constraint validation under feature correlation, described here for the first time; (2) the correspondence space is sampled instead of completely traversed, which results in a great reduction of complexity at the reasonable expense of a small probability of failure; and (3) the concept of locality is introduced, which makes constant the number of map features that the algorithm must consider simultaneously. We have shown how these properties make this algorithm linear with both the size of the stochastic map, and the number of sensor measurements.

This algorithm is potentially useful also in multi-vehicle SLAM. If a group of vehicles independently build maps of regions of an environment, identifying the common area mapped will allow to join the information in a single stochastic map. This constitutes the subject of future work.

Acknowledgements

Enormous thanks to José Guivant and Eduardo Nebot, from the University of Sydney, for making public the Victoria Park dataset, as well as the MATLAB tree segmentation code, at http://www.acfr.usyd.edu.au. This research has been funded in part by the Dirección General de Investigación of Spain under project DPI2000-1265.

References

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

[2] R.C. Bolles and P. Horaud. 3DPO: A three-dimensional part orientation system. *Int. J. Robotics Research*, 5(2):3–26, 1986.

[3] R. Carraghan and P.M. Pardalos. An exact algorithm for the maximum clique problem. *Operations Research Letters*, 9:375–382, 1990.

[4] J. A. Castellanos, J. M. M. Montiel, J. Neira, and J. D. Tardós. The SPmap: A probabilistic framework for simultaneous localization and map building. *IEEE Trans. Robot. Automat.*, 15(5):948–953, 1999.

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

[6] M. W. M. G. Dissanayake, P. Newman, H. F. Durrant-Whyte, S. Clark, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Trans. Robot. Automat.*, 17(3):229–241, June 2001.

[7] O. D. Faugeras and M. Hebert. The representation recognition, and locating of 3D objects. *Int. J. of Robotics Research*, 5(3):27–52, 1986.

[8] H. J. S. Feder, J. J. Leonard, and C. M. Smith. Adaptive mobile robot navigation and mapping. *Int. J. Robotics Research*, 18(7):650–668, July 1999.

[9] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. Assoc. Comp. Mach.*, 24(6):381–395, 1981.

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

[11] J. E. Guivant and E. M. Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Trans. Robot. Automat.*, 17(3):242–257, June 2001.

[12] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, U. K., 2000.

[13] J. H. Lim and J. J. Leonard. Mobile robot relocation from echolocation constraints. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(9):1035–1041, September 2000.

[14] J. Neira and J. D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Trans. Robot. Automat.*, 17(6):890–897, 2001.

[15] J.D. Tardós, J. Neira, P.M. Newman, and J.J. Leonard. Robust mapping and localization in indoor environments using sonar data. *Int. J. Robotics Research*, 21(4):311–330, 2002.

[16] S. Thrun. An online mapping algorithm for teams of mobile robots. *Int. J. Robotics Research*, 20(5):335–363, May 2001.

[17] S. Thrun. Simultaneous mapping and localization with sparse extended information filters: theory and initial results. Technical Report CMU-CS-02-112, Carnegie Mellon University, Computer Science Department, Pittsburgh, PA, 2002.