Divide and Conquer: EKF SLAM in $O(n)$

Lina M. Paz
Instituto de Investigación
en Ingeniería en Aragón
Universidad de Zaragoza, Spain
Email: linapaz@unizar.es

José Guivant
Australian Centre for Field Robotics
The University of Sydney, Australia
Email: jguivant@acfr.usyd.edu.au

Juan D. Tardós and José Neira
Instituto de Investigación
en Ingeniería en Aragón
Universidad de Zaragoza, Spain
Email: tardos,jneira@unizar.es

***Abstract*—In this paper we show that *all* processes associated to the move-sense-update cycle of EKF SLAM can be carried out in time *linear* in the number of map features. We describe Divide and Conquer SLAM, an EKF SLAM algorithm where the computational complexity per step is reduced from $O(n^2)$ to $O(n)$ (the total cost of SLAM is reduced from $O(n^3)$ to $O(n^2)$). In addition, the resulting vehicle and map estimates have better consistency properties than standard EKF SLAM in the sense that the computed state covariance more adequately represents the real error in the estimation. Both simulated experiments and the Victoria Park Dataset are used to provide evidence of the advantages of this algorithm.**

***Index Terms*—SLAM, Computational Complexity, Consistency, Linear Time.**

I. Introduction

The Simultaneous Localization and Mapping (SLAM) problem deals with the construction of a model of the environment being traversed with an onboard sensor, while at the same time maintaining an estimation of the sensor location within the model [1], [2]. Solving SLAM is central to the effort of conferring real autonomy to robots and vehicles, but also opens possibilities in applications where the sensor moves with six degrees of freedom, such as egomotion and augmented reality. SLAM has been the subject of much attention since the seminal work in the late 80s [3], [4], [5], [6].

The most popular solution to SLAM considers it a stochastic process in which the Extended Kalman Filter (EKF) is used to compute an estimation of a state vector $\mathbf{x}$ representing the sensor and environment feature locations, together with the covariance matrix $\mathbf{P}$ representing the error in the estimation. Currently, most of the processes associated to the move-sense-update cycle of EKF SLAM are linear in the number of map features n: vehicle prediction and inclusion of new features [7], [8], continuous data association [9], global localization [10]. The exception is the update of the covariance matrix of the stochastic state vector that represents the vehicle and map states, which is $O(n^2)$. The EKF solution to SLAM has been used successfully in small scale environments, however the $O(n^2)$ computational complexity limits the use EKF-SLAM in large environments. This has been a subject of much interest in research. Postponement [11], the Compressed EKF filter [8], and Local Map Sequencing [12] are alternatives that work on local areas of the stochastic map and are essentially constant time most of the time, although they require periodical $O(n^2)$ updates (given a certain environment and sensor characteristics, an optimal local map size can be derived to minimize the total computational cost [13]). More recently, researchers have pointed out the approximate sparseness of the Information matrix $\mathbf{Y}$, the inverse of the full covariance matrix $\mathbf{P}$. This suggests using the Extended Information Filter, the dual of the Extended Kalman Filter, for SLAM updates. The Sparse Extended Information Filter (SEIF) algorithm [14] approximates the Information matrix by a sparse form that allows $O(1)$ updates on the information vector, and $O(n)$ computations of the state vector $\mathbf{x}$. Nonetheless, data association becomes more difficult when the state and covariance matrix are not available, and the approximation can yield overconfident estimations of the state [15]. This overconfidence is overcome by the Exactly Sparse Extended Information Filter (ESEIF) [16] with a strategy that produces an exactly sparse Information matrix with no introduction of inaccuracies through sparsification.

The Thin Junction Tree Filter algorithm [17] works on the Gaussian graphical model represented by the Information matrix, and achieves high scalability by working on an *approximation*, where weak links are broken. The Treemap algorithm [18] is a closely related technique, which also uses a weak link breakage policy. Recently $\sqrt{SAM}$ [19] provided the insight that the full SLAM problem, the complete vehicle trajectory plus the map, is sparse in information form (although ever increasing). Sparse linear algebra techniques allow to compute the state, without the covariance, in time linear with the whole trajectory and map size. The T-SAM algorithm [20] provides a local mapping version to reduce the computational cost. However, the method remains a batch algorithm and covariance is not available to solve data association.

A second important limitation of standard EKF SLAM is the effect that linearizations have in the consistency of the final vehicle and feature estimates. Linearizations introduce errors in the estimation process that can render the result inconsistent, in the sense that the computed state covariance does not represent the real error in the estimation [21], [22], [23]. Among other things, this shuts down data association, which is based on contrasting predicted feature locations with observations made by the sensor. Thus, important processes in SLAM like loop closing are crippled. The Unscented Kalman Filter [24] avoids linearization via a parametrization of means and covariances through selected points to which the nonlinear transformation is applied. Unscented SLAM has been shown

to have improved consistency properties [25]. These solutions however ignore the computational complexity problem. All algorithms for EKF SLAM based on efficiently computing an approximation of the EKF solution [17], [18] will inevitably suffer from this problem.

In this paper we describe Divide and Conquer SLAM (D&C SLAM), an EKF SLAM algorithm that overcomes these two fundamental limitations:

1) The computational cost per step is reduced from $O(n^2)$ to $O(n)$; the total cost of SLAM is reduced from $O(n^3)$ to $O(n^2)$;
2) the resulting vehicle and map estimates have better consistency properties than standard EKF SLAM in the sense that the computed state covariance adequately represents the real error in the estimation.

Unlike many current large scale EKF SLAM techniques, this algorithm computes an exact solution, without relying on approximations or simplifications to reduce computational complexity. Also, estimates and covariances are available when needed by data association without any further computation. Empirical results show that, as a by-product of reduced computations, and without losing precision because of approximations, D&C SLAM has better consistency properties than standard EKF SLAM.

This paper is organized as follows: in section II we briefly review the standard EKF SLAM algorithm and its computational properties. Section III contains a description of the proposed algorithm. We study of its computational cost in comparison with EKF SLAM, as well as its consistency properties. In section IV we describe an algorithm for carrying out data association in D&C SLAM also in linear time. In section V we use the Victoria Park dataset to carry out an experimental comparison between EKF SLAM and D&C SLAM. Finally in section VI we draw the main conclusions of this work.

II. The EKF SLAM algorithm

The EKF SLAM algorithm (see alg. 1) has been widely used for mapping. Several authors have described the computational complexity of this algorithm [7], [8]. With the purpose of comparing EKF SLAM with the proposed D&C SLAM algorithm, in this section we briefly analyze its computational complexity.

A. Computational complexity of EKF SLAM per step

For simplicity, assume that in the environment being mapped features are distributed more or less uniformly. If the vehicle is equipped with a sensor of limited range and bearing, the amount of measurements obtained at any location will be more or less constant. Assume that at some step k the map contains n features, and the sensor provides m measurements, r of which correspond to re-observed features, and $s = m - r$ which correspond to new features.

The computational complexity of carrying out the move-sense-update cycle of algorithm 1 at step k involves the computation of the *predicted map* $\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}$, which requires

Algorithm 1 : `ekf_slam`

$$\begin{aligned}
\mathbf{z}_0, \mathbf{R}_0 &= get_measurements \\
\hat{\mathbf{x}}_0, \mathbf{P}_0 &= new_map(\mathbf{z}_0, \mathbf{R}_0)
\end{aligned}$$

for $k = 1$ to steps **do**

$$\begin{aligned}
\hat{\mathbf{x}}_{R_k}^{R_{k-1}}, \mathbf{Q}_k &= get_odometry \\
\hat{\mathbf{x}}_{k|k-1}, \mathbf{F}_k, \mathbf{G}_k &= prediction(\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{x}}_{R_k}^{R_{k-1}}) \\
\mathbf{P}_{k|k-1} &= \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^T \qquad (1)
\end{aligned}$$

$$\begin{aligned}
\mathbf{z}_k, \mathbf{R}_k &= get_measurements \\
\mathcal{H}_k, \mathbf{H}_{\mathcal{H}_k} &= data_assoc(\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}, \mathbf{z}_k, \mathbf{R}_k)
\end{aligned}$$

$$\begin{aligned}
\mathbf{S}_{\mathcal{H}_k} &= \mathbf{H}_{\mathcal{H}_k} \mathbf{P}_{k|k-1} \mathbf{H}_{\mathcal{H}_k}^T + \mathbf{R}_{\mathcal{H}_k} \qquad (2) \\
\mathbf{K}_{\mathcal{H}_k} &= \mathbf{P}_{k|k-1} \mathbf{H}_{\mathcal{H}_k}^T / \mathbf{S}_{\mathcal{H}_k} \qquad (3) \\
\mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_{\mathcal{H}_k} \mathbf{H}_{\mathcal{H}_k}) \mathbf{P}_{k|k-1} \qquad (4) \\
\nu_{\mathcal{H}_k} &= \mathbf{z}_k - \mathbf{h}_{\mathcal{H}_k}(\hat{\mathbf{x}}_{k|k-1}) \qquad (5) \\
\hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_{\mathcal{H}_k} \nu_{\mathcal{H}_k} \qquad (6) \\
\hat{\mathbf{x}}_k, \mathbf{P}_k &= add_feat(\hat{\mathbf{x}}, \mathbf{P}_k, \mathbf{z}_k, \mathbf{R}_k, \mathcal{H}_k)
\end{aligned}$$

end for

`return` $\mathbf{m} = (\mathbf{x}_k, \mathbf{P}_k)$

obtaining also the computation of the corresponding jacobians $\mathbf{F}_k, \mathbf{G}_k$, and the *updated map* $\mathbf{x}_k, \mathbf{P}_k$, which requires the computation of the corresponding jacobian $\mathbf{H}_{\mathcal{H}_k}$, the Kalman gain matrix $\mathbf{K}_{\mathcal{H}_k}$, as well as the innovation $\nu_{\mathcal{H}_k}$, and its covariance $\mathbf{S}_k$ (the complexity of data association is analyzed in section IV).

The fundamental issue regarding computational complexity is that all jacobians are *sparse* matrices [7], [8], [19]. Thus, their computation is $O(1)$, but more importantly, since they take part in the computation of both the predicted and updated map, the computational cost of eqs. (1) to (6) can also be reduced. Consider as an example the innovation covariance matrix $\mathbf{S}_k$ in eq. (2). Normally, the computation of this $r \times r$ matrix would require $rn^2 + r^2 n$ multiplications and $rn^2 + r^2 n + r^2$ sums, that is, $O(n^2)$ operations (see fig. 1). But given that matrix $\mathbf{H}_k$ is sparse, with an effective size of $r \times c$, the computation requires $rcn + r^2 c$ multiplications and $rcn + r^2 c + r^2$ sums, that is, $O(n)$ operations. Similar analysis leads to the conclusion that the cost of computing both the predicted covariance $\mathbf{P}_{k|k-1}$ and the Kalman gain matrix $\mathbf{K}_{\mathcal{H}_k}$ is $O(n)$, and that the greatest cost in an EKF SLAM update is the computation of the covariance matrix $\mathbf{P}_k$, which is $O(n^2)$. Thus, the computational cost per step of EKF SLAM is quadratic on the size of the map:

$$C_{EFK,k} = O(n^2) \qquad (7)$$

Figure 3 shows the results of carrying out EKF SLAM

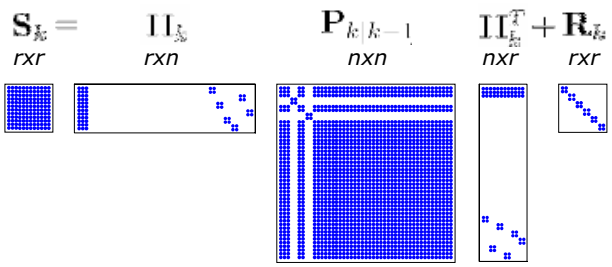

Fig. 1. Computation of the innovation covariance $\mathbf{S}_k$ matrix: the computatio requires $O(n)$ operations ($rn^2 + r^2n$ multiplications and $rn^2 + r^2n + r$ sums).

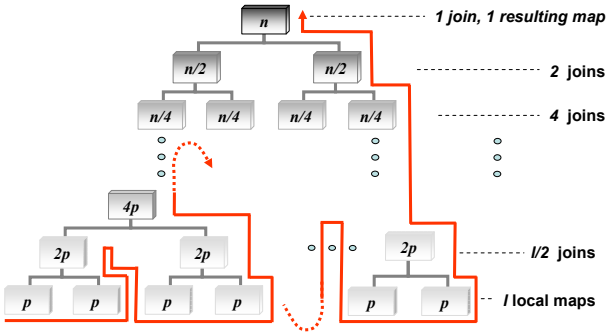

Fig. 2. Binary tree representing the hierarchy of maps that are created and joined in D&C SLAM. The red line shows the sequence in which maps are created and joined.

in four simulated scenarios. In an environment with uniform distribution of point features, the vehicle performs a $1m$ motion at every step. The odometry of the vehicle has standard deviation error of $10cm$ in the x direction (the direction o motion), $5cm$ in y direction, and $(0.5deg)$ for orientation. W simulate an onboard range and bearing sensor with a range of $3m$, so that 16 features are normally seen at every step The standard deviation error is 5% of the distance in range and $1deg$ in bearing. Four different trajectories are carried out: straight forward exploration (first column); loop closing (second column), lawn mowing (third column), and snail path (fourth column). The execution time of EKF SLAM per step for each of these trajectories is shown in fig. 3, second row.

B. Total computational complexity of EKF SLAM

Assume that the process of building a map of size n features is carried out with an exploratory trajectory, in which the sensor obtains m measurements per step as said before, s of which are new (all four examples in fig. 3, straight forward, loop closing, lawn mowing and spiral path, are exploratory trajectories). Given that s new features are added to the map per step, n/s steps are required to obtain the final map of size n, and thus the total computational complexity will be:

$$\begin{aligned} C_{EKF} &= O\left(\sum_{k=1}^{n/s}(ks)^2\right) \\ &= O\left(s^2\sum_{k=1}^{n/s}k^2\right) \\ &= O\left(s^2\frac{(n/s)(n/s+1)(2n/s+1)}{6}\right) \\ &= O\left(\frac{1}{6}2n^3/s + 3n^2 + ns\right) \\ &= O(n^3) \end{aligned} \tag{8}$$

The total cost of computing a map is cubic with the final size of the map. The total execution time of EKF SLAM for each of these trajectories is shown in fig. 3, third row.

III. The Divide and Conquer Algorithm

The Divide and Conquer algorithm for SLAM (D&C SLAM) is an EKF-based algorithm in which a sequence of local maps of minimum size p is produced using the standard EKF SLAM algorithm [26]. These maps are then joined using the map joining procedure of [12], [27] (or the improved version 2.0 detailed in [26]) to produce a single final stochastic map.

Instead of joining each new local map to a global map sequentially, as Local Map Sequencing does [12], D&C SLAM carries out map joining in a binary hierarchical fashion, as depicted in fig. 2. Although algorithms like Treemap [18] use a similar structure, the tree is not used to sort features, here it represents the hierarchy of local maps that are computed. The leaves of the tree are the sequence of local maps of minimal size p that the algorithm produces with standard EKF-SLAM. The intermediate nodes represent the maps resulting form the intermediate map joining steps that are carried out, and the root of the tree represents the final map that is computed. D&C follows algorithm 2, which performs a *postorder* traversal of the tree using a stack to save intermediate maps. This allows a sequential execution of D&C SLAM.

A. Total computational complexity of D&C SLAM

In D&C SLAM, the process of building a map of size n produces $l = n/p$ maps of size p, at cost $O(p^3)$ each (see eq. (8)), which are joined into $l/2$ maps of size $2p$, at cost $O((2p)^2)$ each. These in turn are joined into $l/4$ maps of size $4p$, at cost $O((4p)^2)$ each. This process continues until two local maps of size $n/2$ are joined into 1 local map of size n, at a cost of $O(n^2)$. Thus, the total computational complexity of D&C SLAM is (note that the sum represents all costs associated to map joining, which is $O(n^2)$ [12]):

$$\begin{aligned} C_{DC} &= O\left(p^3 l + \sum_{i=1}^{\log_2 l}\frac{l}{2^i}(2^i p)^2\right) \\ &= O\left(p^3 n/p + \sum_{i=1}^{\log_2 n/p}\frac{n/p}{2^i}(2^i p)^2\right) \\ &= O\left(p^2 n + \sum_{i=1}^{\log_2 n/p} p\frac{n}{2^i}(2^i)^2\right) \end{aligned}$$

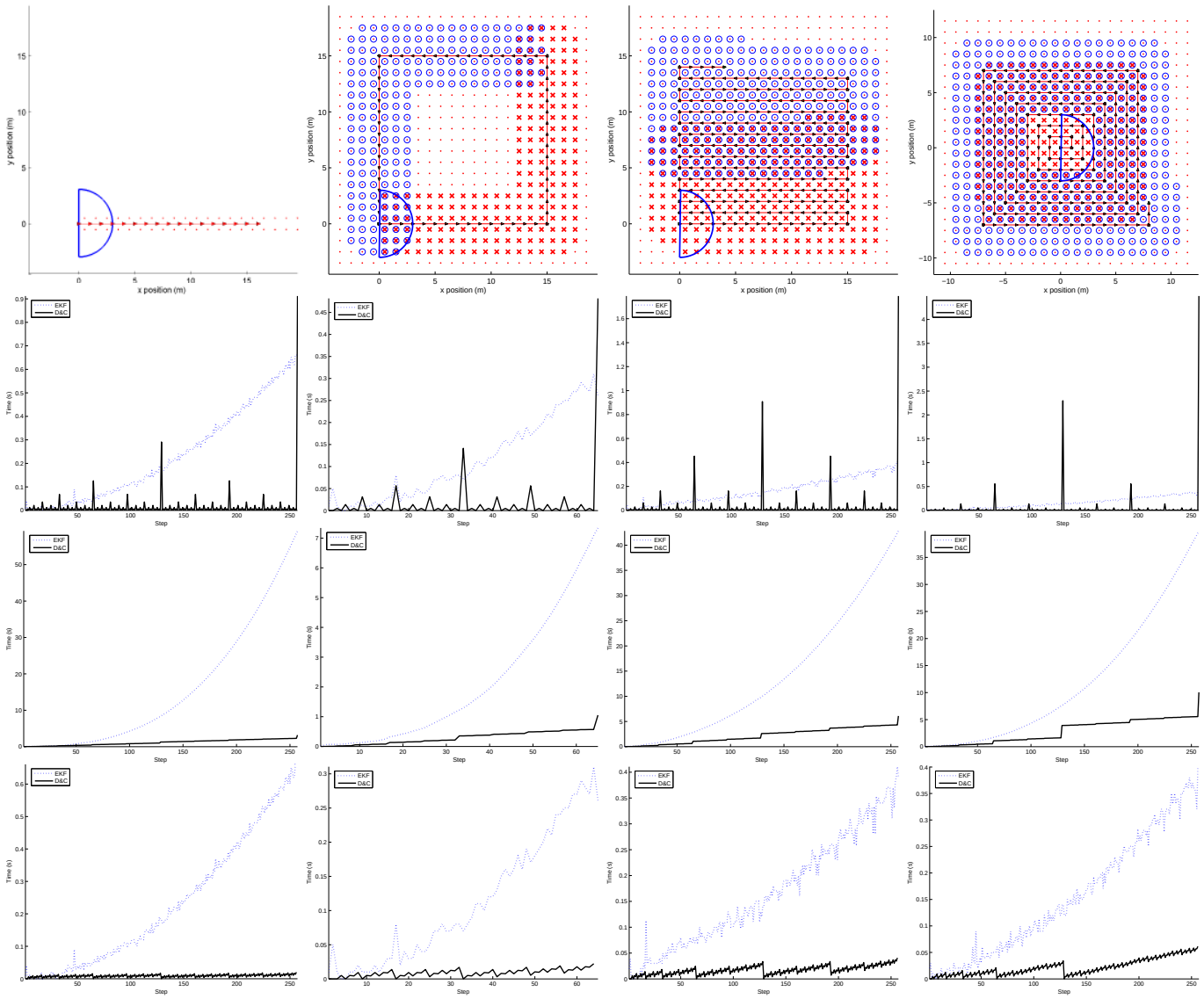

Fig. 3. Four simulated experiments for comparing the EKF and D&C SLAM algorithms: detail of a straight forward trajectory (first colum); loop closing (second column); lawn mowing (third column); snail path (fourth column). Ground truth environment, trajectory and first and second halves $n/2$ of maps features for data association analysis (top row); execution time per step of EKF .vs. D&C SLAM (second row); total execution time of EKF .vs. D&C SLAM (third row); execution time per step of EKF .vs. amortized execution time per step of D&C SLAM (bottom row).

$$= O\left(p^2 n + p\, n \sum_{i=1}^{\log_2 n/p} 2^i\right)$$

The sum in the equation above is a geometric progression of the type:

$$\sum_{i=1}^{k} r^i = \frac{r - r^{k+1}}{1 - r}$$

Thus, in this case:

$$\begin{aligned} C_{DC} &= O\left(p^2 n + p\, n \frac{2^{\log_2 n/p+1} - 2}{2 - 1}\right) \\ &= O\left(p^2 n + p\, n(2\, n/p - 2)\right) \\ &= O\left(p^2 n + 2n^2 - 2pn\right) \\ &= O(n^2) \end{aligned} \tag{9}$$

This means that D&C SLAM performs SLAM with a total cost quadratic with the size of the environment, as compared with the cubic cost of standard EKF-SLAM. The difference between this approach and other approaches that also use map joining, such as Local Map Sequencing, is that in D&C SLAM the number of map joining operations carried out is proportional to $\log(n)$, instead of n. This allows the total cost to remain quadratic with n.

Figure 3, second and third rows, show the execution time per step and total execution time, respectively, for D&C SLAM .vs. EKF SLAM for the four simulations of straight forward, loop closing, lawn mowing and spiral path. It can be seen that

Algorithm 2 : dc_slam
sequential implementation using a stack.

```
stack = new()
m_0 = ekf_slam()
stack = push(m_0, stack)
{
Main loop: postorder traversing of the map tree.
}
repeat
  m_k = ekf_slam()
  while ¬ empty(stack) and then
  size(m_k) ≥ size(top(stack)) do
    m = top(stack)
    stack = pop(stack)
    m_k = join(m, m_k)
  end while
  stack = push(m_k, stack)
until end_of_map
{
Wrap up: join all maps in stack for full map recovery.
}
while ¬ empty(stack) do
  m = top(stack)
  stack = pop(stack)
  m_k = join(m, m_k)
end while
return (m_k)
```

the total cost of D&C SLAM very quickly separates from the total cost of EKF SLAM. The reason is that the computational cost per step of D&C SLAM is lower than that of EKF SLAM most of the time. EKF SLAM works with a map of non-decreasing size, while D&C SLAM works on local maps of small size most of the time. In some steps though (in the simulation those which are a multiple of 2), the computational cost of D&C is higher than EKF. In those steps, one or more map joining operations take place (in those that are a power of 2, 2^l, l map joining operations take place).

B. Computational complexity of D&C SLAM per step

In D&C SLAM, the map to be generated at step k will not be required for joining until step $2\,k$. We can therefore amortize the cost $O(k^2)$ at this step by dividing it up between steps k to $2\,k-1$ in equal $O(k)$ computations for each step. We must however take into account all joins to be computed at each step. If k is a power of 2 ($k = 2^l$), $i = 1 \cdots l$ joins will take place at step k, with a cost $O(2^2) \ldots O((2^l)^2)$. To carry out join i we need join $i-1$ to be complete. Thus if we wish to amortize all joins, we must wait until step $k+k/2$ for join $i-1$ to be complete, and then start join i. For this reason, the amortized version of this algorithm divides up the largest join at step k into steps $k+k/2$ to $2\,k-1$ in equal $O(2\,k)$ computations for each step. Amortization is very simple, the computation of the elements of $P_{k|k}$ is divided in $k/2$ steps. If $P_{k|k}$ is of size $n \times n$, $2n^2/k$ elements have to be computed per step.

Fig. 3 (bottom row) shows the resulting amortized cost per step for the four simulated experiments. Note that at steps $i = 2^l$, the cost falls steeply. As said before, in these steps l joins should be computed, but since join i required the map resulting from join $i-1$, all l joins are postponed. We can see that the amortized cost of D&C SLAM always lower than that of EKF SLAM. D&C SLAM is an anytime algorithm, if at any moment during the map building process the full map is required for another task, it can be computed in a single $O(n^2)$ step.

C. Consistency in Divide and Conquer SLAM

Apart from computational complexity, another important aspect of the solution computed by the EKF has gained attention recently: map consistency. When the ground truth solution $\mathbf{x}$ for the state variables is available, a statistical test for filter consistency can be carried out on the estimation $(\hat{\mathbf{x}},\, \mathbf{P})$, using the Normalized Estimation Error Squared (NEES), defined as:

$$D^2 = (\mathbf{x} - \hat{\mathbf{x}})^T \; \mathbf{P}^{-1} \; (\mathbf{x} - \hat{\mathbf{x}}) \tag{10}$$

Consistency is checked using a chi-squared test:

$$D^2 \leq \chi^2_{r,1-\alpha} \tag{11}$$

where $r = dim(\mathbf{x})$ and α is the desired significance level (usually 0.05). If we define the consistency index of a given estimation $(\hat{\mathbf{x}},\, \mathbf{P})$ with respect to its true value $\mathbf{x}$ as:

$$CI = \frac{D^2}{\chi^2_{r,1-\alpha}} \tag{12}$$

when $CI < 1$, the estimation is consistent with ground truth, and when $CI > 1$, the estimation is inconsistent (optimistic) with respect to ground truth.

We tested consistency of both standard EKF and D&C SLAM algorithms by carrying 20 Monte Carlo runs on the simulated experiments. We have used simulated experiments to test consistency because this allows to have ground truth easily available. Additionally, Monte Carlo runs allow to gather statistically significant evidence about the consistency properties of the algorithms being compared, while a single experiment allows to carry out only one run of the algorithms.

Figure 4 (top) shows the evolution of the mean consistency index of the vehicle orientation during all steps of the straight forward trajectory simulation. We can see that the D&C estimate on vehicle location is always more consistent than the standard EKF estimate, EKF falls out of consistency while D&C remains consistent. In order to obtain a value for consistency in all steps, we emptied the stack and carried out all joins at every step to obtain the full map, but this is not done normally.

Figure 4 (bottom) shows the evolution of the mean absolute angular error of the vehicle. The 2σ bounds for the theoretical (without noise) and computed (with noise) uncertainty of both standard EKF and *Divide and Conquer* SLAM algorithms are also drawn. We can see how the error increases more slowly

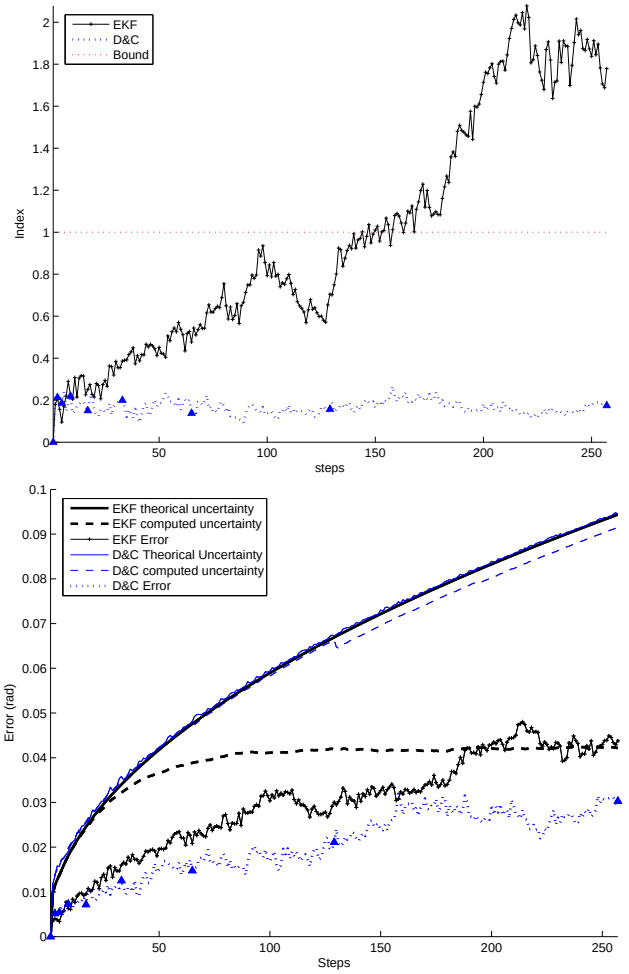

Fig. 4. Mean consistency index for the robot orientation(top); Mean absolute angular robot error (bottom).

in the case of D&C SLAM, but we can also see that the main cause of inconsistency in the standard EKF SLAM is the fast rate at which the computed uncertainty falls below its theoretical value.

IV. Data Association for Divide and Conquer SLAM

The data association problem in continuous SLAM consists in establishing a correspondence between each of the m sensor measurements and one (on none) of the n map features. The availability of a stochastic model for both the map and the measurements allows to check each measurement-feature correspondence for *individual compatibility* using a hypothesis test on the innovation of the pairing and its covariance [9]. In standard EKF SLAM, and for a sensor of limited range and bearing, m is constant and thus individual compatibility is $O(nm) = O(n)$, linear on the size of the map. This cost can be easily reduced to $O(m)$, constant, by a simple tessellation or grid of the map computed during map building, which allows to determine candidates for a measurement in constant time simply by checking the grid element in which it falls.

In cases where clutter or vehicle error are high, there may be many more than one possible correspondence for each measurement. More elaborate algorithms are required to disambiguate in these cases. Nevertheless, the overlap between the measurements and the map is the size of the sensor range plus the vehicle uncertainty, and thus more or less constant. After individual compatibility is sorted out, any disambiguation algorithm, such as **JCBB** [9], will then disambiguate between the m measurements and a region of the map of constant size, regardless of map size, and thus will execute in constant time.

We use **JCBB** in the case of building the local maps of size p, given that it is a standard EKF-SLAM process. However, data association for D&C SLAM is a critical issue because map joining involves finding correspondences between two local maps of similar size in accordance with their level in the tree. For instance, before of obtaining the final map, the data association problem has to be solved between two maps of size $n/2$ maps and so computing individual compatibility becomes $O(n^2)$. Fortunately, this can be easily reduced to linear again using a simple tessellation or grid for the maps.

The size of the region of overlap between two maps in D&C SLAM depends on the environment and type of trajectory. Consider the simulated examples of fig. 3 where two $n/2$ maps are shown (features in the first map are red crosses, features in the second are blue circles). In the second case, the square loop, the region of overlap between two maps will be of constant size, basically dependent on the sensor range. In the case of the lawn mowers trajectory, the overlap will be proportional to the length of the trajectory before the vehicle turns back, still independent of map size, and thus constant. In the fourth case, the snail path, the region of overlap between the inner map and the encircling map is proportional to the final map size. In these cases, data association algorithms like **JCBB** will not execute in constant time.

In order to limit the computational cost of data association between local maps in D&C SLAM, we use a *randomized joint compatibility* algorithm. Our **RJC** approach (see algorithm 3) is a variant of the linear **RS** algorithm [10]) used for global localization.

Consider two consecutive maps $\mathbf{m}_1$ and $\mathbf{m}_2$, of size n_1 and n_2 respectively, to be joined. First, the overlap between the two maps is identified using individual compatibility. Second, instead of performing branch and bound interpretation tree search in the whole overlap as in **JCBB**, we randomly select b features in the overlapped area of the second map and use **JCBB***: a version of **JCBB** *without exploring the star node*, i.e., considering all b measurements good. This produces a hypothesis $\mathcal{H}$ of b jointly compatible features in the first map. Associations for the remaining features in the overlap are obtained using the simple nearest neighbor rule given hypothesis $\mathcal{H}$, which amounts to finding pairings that are compatible with the first b features. In the spirit of adaptative **RANSAC** [28], we repeat this process t times, so that the probability of missing a correct association is limited to P_{fail}.

Since **JCBB** is executed using a fixed number of features,

Algorithm 3 : RJC

$\mathbf{P}_{fail} = 0.01$, $\mathbf{P}_{good} = 0.8$, $b = 4$
$i = 1$, $Best$ = []
while $(i \leq t)$ **do**
  $\mathbf{m}_2^*$ = random_select($\mathbf{m}_2$, b)
  $\mathcal{H}$ = JCBB*($\mathbf{m}_1$, $\mathbf{m}_2^*$)
  $\mathcal{H}$ = NN($\mathcal{H}$, $\mathbf{m}_1$, $\mathbf{m}_2^*$)
  if pairings($\mathcal{H}$) > pairings($Best$) **then**
    $Best = \mathcal{H}$
  end if
  $\mathbf{P}_{good}$ = $max(\mathbf{P}_{good}, pairings(Best) \backslash m)$
  $t = \log \mathbf{P}_{fail} / \log(1 - \mathbf{P}_{good}^b)$
  $i = i + 1$
end while

its cost remains constant. Finding the nearest neighbor for each remaining feature among the ones that are individually compatible with it, a constant number, will be constant. The cost of each try is thus $O(n)$. The number of tries depends on the number of features randomly selected (b), on the probability that a selected feature in the overlap can be actually found in the first map (P_{good}), and on the acceptable probability of failure in this probabilistic algorithm (P_{fail}). It does not depend on the size of either map. In this way, we can maintain data association in D&C SLAM linear with the size of the joined map.

V. Experiments

We have used the well known Victoria Park data set to validate the algorithms D&C SLAM and **RJC**. This experiment is particulary adequate for testing SLAM due its large scale, and the significant level of spurious measurements. The experiment also provides critical loops in absence of reliable features.

For **RJC**, we chose $b = 4$ as the number of map features to be randomly selected as seed for hypothesis generation. Two features are sufficient in theory to fix the relative location between the maps, but we have found 4 to adequately disambiguate. The probability that a selected feature in the overlap is not spurious, $\mathbf{P}_{good}$ is set to 0.8, and the probability of not finding a good solution when one exists, $\mathbf{P}_{fail}$ is set to 0.01. These parameters make the data association algorithm carry out 9 random tries.

Figure 5 shows the resulting maps from standard EKF SLAM .vs. D&C SLAM, which are essentially equivalent; there are some minor differences due to missed associations in the case of EKF. Figure 6, top, shows the amortized cost of D&C SLAM. We can see that in this experiment an EKF step can take 0.5 seconds, while the amortized D&C SLAM step will take at most 0.05 seconds. In this experiment, the total cost of D& C SLAM is one tenth of the total cost of standard EKF (fig. 6, bottom).

VI. Conclusions

In this paper we have shown that EKF SLAM can be carried out in time *linear* with map size. We describe and EKF SLAM

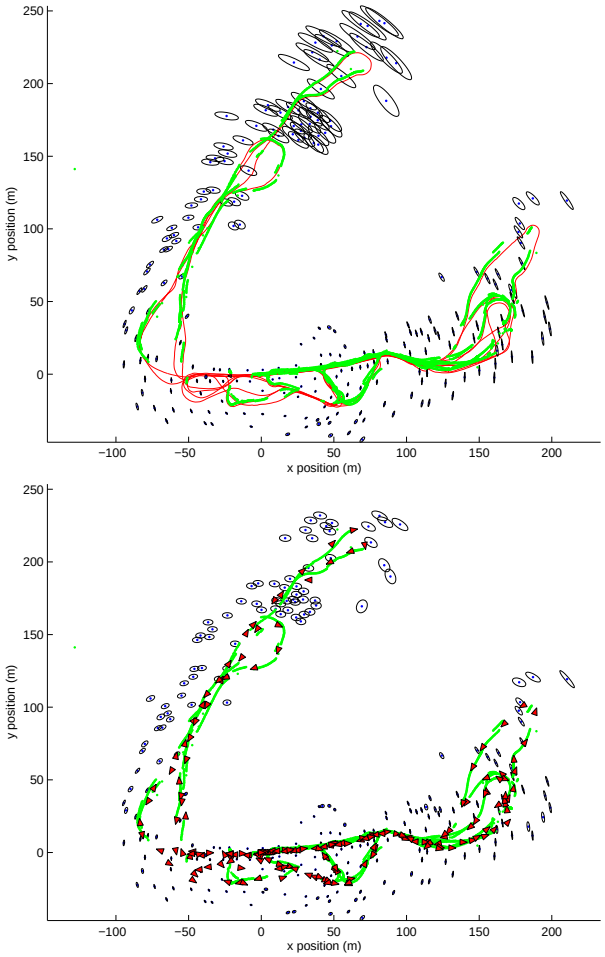

Fig. 5. Map for Victoria Park dataset: according to the standard EKF SLAM algorithm (top); according to the D &C SLAM algorithm. The results are essentially equivalent; some missed associations may result in minor differences. The estimated position along the whole trajectory is shown as a red line for EKF SLAM, and the vehicle locations are drawn as red triangles when available in D&C SLAM. Green points are GPS readings in both cases.

variant: *Divide and Conquer* SLAM, a simple algorithm to implement. In contrast with many current efficient SLAM algorithms, all information required for data association is available when needed with no further processing. D&C SLAM computes the exact EKF SLAM solution, the state *and* its covariance, with no approximations, and with the additional advantage of providing always a more precise and consistent vehicle and map estimate. All information is available for data association, which can also be carried out in linear time per step. We hope to have shown that D&C SLAM is the algorithm to use in all applications in which the Extended Kalman Filter solution is to be used.

Despite of the differences with other methods presented in section I, a very important fact to be emphasized is that the D&C map splitting strategy can also be incorporated in those recent algorithms. This idea is part of our future work.

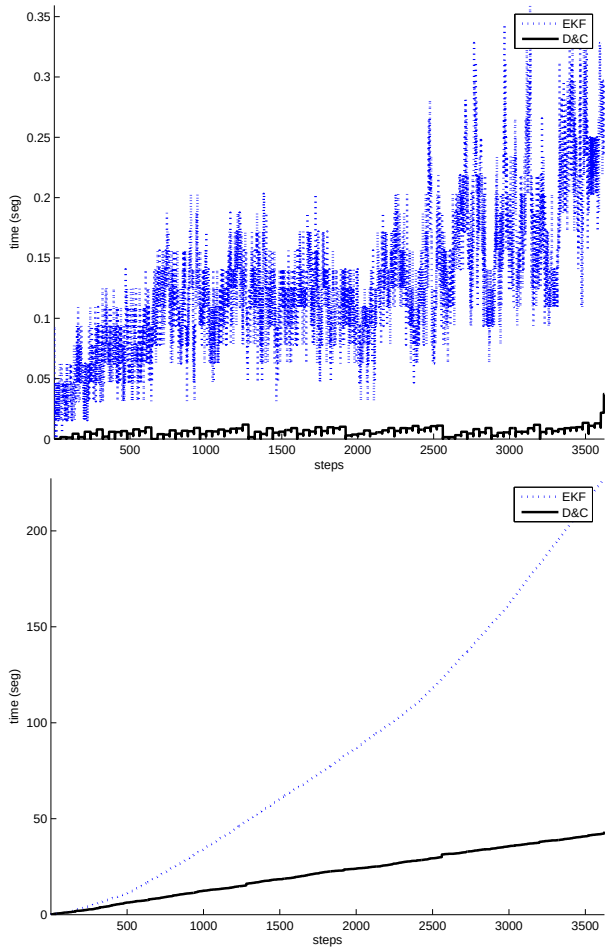

Fig. 6. Time per step of EKF and D&C SLAM for Victoria experiment (top); time per step of EKF SLAM .vs. amortized time per step of D& SLAM (middle); accumulated time of EKF SLAM .vs. D& SLAM.

ACKNOWLEDGMENT

This research has been funded in part by the Dirección General de Investigación of Spain under projects DPI2003-07986 and DPI2006-13578.

REFERENCES

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[2] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 3, pp. 108–117, 2006.

[3] R. Chatila and J. Laumond, "Position referencing and consistent world modeling for mobile robots," *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, vol. 2, 1985.

[4] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *Int. J. of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[5] R. Smith, M. Self, and P. Cheeseman, "A stochastic map for uncertain spatial relationships," in *Robotics Research, The Fourth Int. Symposium*, O. Faugeras and G. Giralt, Eds. The MIT Press, 1988, pp. 467–474.

[6] J. Leonard and H. Durrant-Whyte, "Simultaneous Map Building and Localization for an Autonomous Mobile Robot," in *1991 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Osaka, Japan, 1991, pp. 1442–1447.

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

[8] J. E. Guivant and E. M. Nebot, "Optimization of the Simultaneous Localization and Map-Building Algorithm for Real-Time Implementation," *IEEE Trans. on Robotics and Automation*, vol. 17, no. 3, pp. 242–257, 2001.

[9] J. Neira and J. D. Tardós, "Data Association in Stochastic Mapping Using the Joint Compatibility Test," *IEEE Trans. Robot. Automat.*, vol. 17, no. 6, pp. 890–897, 2001.

[10] 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.

[11] J. Knight, A. Davison, and I. Reid, "Towards constant time SLAM using postponement," in *IEEE/RSJ Int'l Conf on Intelligent Robots and Systems*, Maui, Hawaii, 2001, pp. 406–412.

[12] J. Tardós, J. Neira, P. Newman, and J. Leonard, "Robust mapping and localization in indoor environments using sonar data," *Int. J. Robotics Research*, vol. 21, no. 4, pp. 311–330, 2002.

[13] L. Paz and J. Neira, "Optimal local map size for ekf-based slam," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China., October 2006.

[14] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous Localization and Mapping with Sparse Extended Information Filters," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 693–716, 2004.

[15] R. Eustice, M. Walter, and J. Leonard, "Sparse extended information filters: Insights into sparsification," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Alberta, Canada, August 2005.

[16] M. Walter, R. Eustice, and J. Leonard, "A provably consistent method for imposing sparsity in feature-based slam information filters," in *Proc. of the Int. Symposium of Robotics Research (ISRR)*, 2004.

[17] M. A. Paskin, "Thin Junction Tree Filters for Simultaneous Localization and Mapping," in *Proc. of the 18th Joint Conference on Artificial Intelligence (IJCAI-03)*, San Francisco, CA., 2003, pp. 1157–1164.

[18] U. Frese, *Treemap: An o(logn) algorithm for simultaneous localization and mapping*. Springer Verlag, 2005, ch. Spatial Cognition IV, p. 455476.

[19] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing," *Intl. Journal of Robotics Research*, vol. 25, no. 12, December 2006.

[20] K. Ni, D. Steedly, and F. Dellaert, "Tectonic SAM: Exact, Out-of-Core, Submap-Based SLAM," in *2007 IEEE Int. Conf. on Robotics and Automation*, Rome, Italy, April 2007.

[21] S. J. Julier and J. K. Uhlmann, "A Counter Example to the Theory of Simultaneous Localization and Map Building," in *2001 IEEE Int. Conf. on Robotics and Automation*, Seoul, Korea, 2001, pp. 4238–4243.

[22] J. Castellanos, J. Neira, and J. Tardós, "Limits to the consistency of EKF-based SLAM," in *5th IFAC Symposium on Intelligent Autonomous Vehicles*, Lisbon, Portugal, 2004.

[23] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the ekf-slam algorithm," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[24] S. Julier and J. Uhlmann, "A new extension of the Kalman Filter to nonlinear systems," in *International Symposium on Aerospace/Defense Sensing, Simulate and Controls*, Orlando, FL, 1997.

[25] R. Martinez-Cantin and J. A. Castellanos, "Unscented SLAM for large-scale outdoor environments," in *2005 IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, Edmonton, Alberta, Canada, 2005, pp. pp. 328–333.

[26] L. Paz, P. Jensfelt, J. D. Tards, and J. Neira, "EKF SLAM Updates in O(n) with Divide and Conquer," in *2007 IEEE Int. Conf. on Robotics and Automation*, Rome, Italy., April 2007.

[27] S. B. Williams, "Efficient solutions to autonomous mapping and navigation problems," Ph.D. dissertation, Australian Centre for Field Robotics, University of Sydney, September 2001, available at http://www.acfr.usyd.edu.au/.

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