2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

EKF SLAM updates in $O(n)$ with *Divide and Conquer* SLAM

L.M. Paz, P. Jensfelt, J.D. Tardós and J. Neira

***Abstract*— In this paper we describe *Divide and Conquer* SLAM (D&C SLAM), an algorithm for performing Simultaneous Localization and Mapping using the Extended Kalman Filter. D&C SLAM overcomes the two fundamental limitations of standard EKF SLAM: 1- the computational cost per step is reduced from $O(n^2)$ to $O(n)$ (the cost full 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 bi-product of reduced computations, and without losing precision because of approximations, D&C SLAM has better consistency properties than standard EKF SLAM. Both characteristics allow to extend the range of environments that can be mapped in real time using EKF. We describe the algorithm and study its computational cost and consistency properties.**

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) consists in building a map of an unknown environment by traversing it using a vehicle with an onboard sensor, while simultaneously determining the vehicle location within the map. In the Extended Kalman Filter solution to SLAM (EKF SLAM), this problem is stated as a stochastic estimation process, in which a move-sense-update cycle is carried out. At every step, the EKF is used to obtain the state vector estimate $\hat{\mathbf{x}}$ containing the vehicle pose and n feature locations, along with the estimated error covariance matrix $\mathbf{P}$.

The EKF solution to SLAM has been used successfully in small scale environments. However, the $O(n^2)$ cost of updating the covariance matrix at each step limits the use of EKF SLAM in large environments [3], [9]. This has been subject of much interest in research. Early improvements include Postponement [12], the Compressed EKF filter [9], and Local Map Sequencing [16]. These algorithms 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. More recently, researchers have pointed out the approximate sparseness of the information matrix, the inverse of the full covariance matrix $\mathbf{P}$, that suggests using the Extended Information Filter, the dual of the Extended Kalman Filter, for SLAM updates. The Sparse Extended Information Filter (SEIF) algorithm [17] approximates the information matrix by a sparse form that allows $O(1)$ updates and $O(n)$ computations of the state vector $\mathbf{x}$. Nontheless, data association becomes more difficult when covariance matrix is not available, and the approximation can yield overconfident estimations of the state [6]. This overconfidence is overcome by the Exactly Sparse Extended Information Filter (ESEIF) [18] with a strategy that produces an exactly sparse Information matrix with no introduction of inaccuracies through sparsification. The Thin Junction Tree Filter algorithm [14] 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 [8] is a closely related technique.

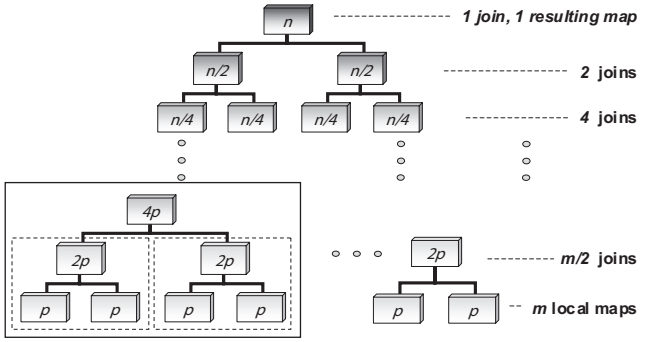

Fig. 1. Binary tree representing the hierarchy of maps that are created and joined in D&C SLAM. The leaves of the tree are the sequence of local maps of minimal size (p) that the algorithm computes with normal EKF SLAM. The intermediate nodes represent the maps resulting from intermediate joining steps that are carried out, and their final size.

Most of these algorithms focus on computational issues, ignoring another important limitation of standard EKF SLAM that has gained attention recently: the effect of linearizations in the consistency of the final vehicle and feature estimates. Given that SLAM is a nonlinear problem, the Kalman Filter, designed for linear systems, is extended by linearizing around the current estimate. This introduces errors in the estimation process that can render the result inconsistent, in the sense that the computed state covariance $\mathbf{P}$ does not represent the real error in the estimation [11], [4], [1]. 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 (UKF) [10] avoids linearization via a parametrization of means and covariances through

L.M. Paz, J.D. Tardós and J. Neira are with the Departamento de Informática e Ingenieria de Sistemas, Centro Politécnico Superior, Universidad de Zaragoza, Zaragoza, Spain {linapaz, jneira, tardos}@unizar.es

P. Jensfelt is with the Royal Institute of Technology (KTH), Stockholm, Sweden patric@nada.kth.se

selected points to which the nonlinear transformation is applied. Unscented SLAM has been shown to have improved consistency properties [13]. Graphical SLAM [7] works on the Gaussian graphical model, and handles non-linearities and reversible data associations. These solutions however ignore the computational complexity problem.

In this paper we propose *Divide and Conquer* SLAM (D&C SLAM), an EKF SLAM algorithm that addresses both the computational problem and the consistency problem. It is based on the idea of Local Map Sequencing proposed in [16]. In Local Map Sequencing, instead of working on a single absolute map, a sequence of local independent maps *of equal constant size* are generated as the vehicle traverses the environment, and then joined at fixed intervals during the process to produce the final absolute map. For local maps of fixed size, it was shown in [16] that the computational cost could be reduced by a large constant factor, but was still $O(n^2)$ in every map joining step. The algorithm proposed here works with a binary tree of local maps of different sizes (see fig. 1), so that the number of map joining steps is minimized and map joining can be performed in an amortized way. We show that the computational complexity at each step is reduced from $O(n^2)$ to $O(n)$, with a total cost for full SLAM of $O(n^2)$ compared to $O(n^3)$ for standard EKF SLAM. Furthermore, map joining is known to exhibit better consistency properties than full EKF SLAM [15], [5]. Here we show that the parametrization which the proposed algorithm carries out on the environment map produces always vehicle and map estimates with better consistency properties than those provided by standard EKF SLAM. Since the algorithm works with the Kalman Filter form and no approximations or simplifications are required, central processes such as data association can be carried out as in standard EKF SLAM with no further processing.

This paper is organized as follows: section II contains a description of the proposed algorithm, and a study of its computational cost. In section III we study the consistency properties of D&C SLAM in the simulated experiments. We have used a simulated experiment because Monte Carlo runs allow to gather statistically significant evidence about the consistency properties of the algorithms being compared. Finally in section IV we draw the main conclusions of this work and discuss future directions of research. Appendix I contains the mathematical details of the improved map joining process that we use in this work.

II. THE DIVIDE AND CONQUER SLAM ALGORITHM

The central idea of D&C SLAM is very simple: instead of doing *Local Map Sequencing*, building a sequence of local maps of some fixed size (see [15] for a discussion on how to decide this size for a given sensor and environment), and then joinining them sequentially to form the complete map, D&C SLAM joins local maps in a *binary tree* fashion (see fig. 1). Standard EKF SLAM is carried out up to a fixed maximal (small) size p. For a given environment requiring m such local maps, they will be joined into $m/2$ local maps of double size, which in turn will be joined into $m/4$ local maps of quadruple size, until the final map of size n will be the result of joining 2 maps of size $n/2$.

Algorithm 1 `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)
```

Carrying out this process sequentially amounts to traversing the binary tree in *postorder*, and can be easily implemented using a stack of maps (see Algorithm I).

A. Computational Complexity of full D&C SLAM

Without loss of generality, consider performing SLAM in an environment where the density of features is uniform. At every step k, the onboard sensor of limited range provides a set of measurements. During exploration, a fraction will correspond to features already in the map, and the rest will correspond to new features that should be included in the map. While we carry out a straightforward trajectory (before loop closing, see fig. 2), the map will grow in size in proportion to k. Thus, the cost of an update will be $O(k^2)$, and the final cost of full EKF SLAM will be cubic on the total number of steps.

Assume mapping such an environment using D&C SLAM, starting with local maps of some fixed maximal size of p features. If the total size of the environment requires m such maps to be covered fully, the total number of map features will be at most $n = p\,m$. The cost of building each map of p features will be $O(p^3)$. Considering only the higher order term, each local map will cost $K_1\,p^3$. Thus, the computational cost of these m local maps will be $K_1\,p^3\,m$.

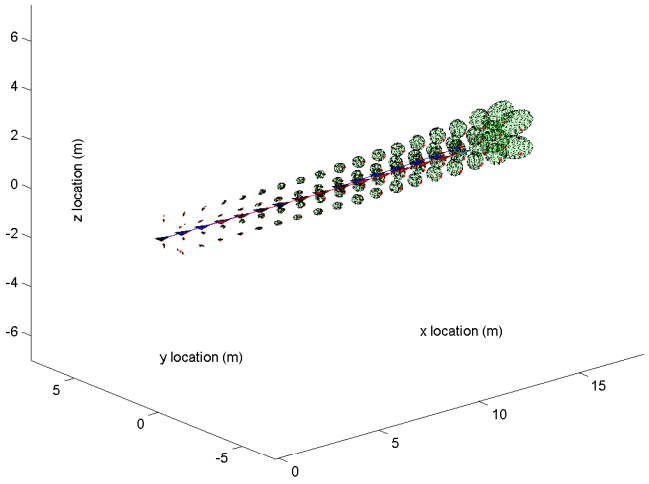

Fig. 2. Initial trajectory and map for the simulated 3D experiment.

In the proposed algorithm, these m maps will be joined into $m/2$ maps of approximately double size (slightly less when there are repeated features). Map joining is $O(n^2)$ on the final map size n. Again considering only the higher order computational cost, each of these map joinings will thus have a $K_2\,(p+p)^2$ computational cost, for a total cost of $K_2\,(2p)^2\,(m/2)$. These in turn will be joined into $m/4$ maps at a cost of $K_2\,(4p)^2$ each, for a total cost of $K_2\,(4p)^2\,(m/4)$. Continuing in this fashion, the total computational cost of this process will be:

$$\begin{aligned} C &= K_1\,p^3\,m + \sum_{i=1}^{\log_2 m} K_2\,\frac{m}{2^i}(2^i\,p)^2 \\ &= K_1\,p^3\,m + \sum_{i=1}^{\log_2 m} K_2\,p^2\,\frac{m}{2^i}(2^i)^2 \\ &= K_1\,p^3\,m + K_2\,p^2\,m\sum_{i=1}^{\log_2 m} 2^i \end{aligned}$$

The sum is a geometric progression of the type:

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

Thus, in this case:

$$\begin{aligned} C &= K_1\,p^3\,m + K_2\,p^2\,n\left(\frac{2^{\log_2 m+1}-2}{2-1}\right) \\ &= K_1\,p^3\,m + K_2\,p^2\,m\,(2\,m-2) \\ &= 2\,K_2\,p^2\,m^2 + \left(K_1\,p^3 - 2\,K_2\,p^2\right)m \end{aligned}$$

Given that $n = p\,m$:

$$C = 2\,K_2\,n^2 + \left(K_1\,p^2 - 2\,K_2\,p\right)n$$

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.

B. Computational Complexity of D&C SLAM per step

Fig. 3 (top) shows the computational cost per step for 256 steps of D&C SLAM versus EKF SLAM in a simulated experiment in which the vehicle performs a $1m$ motion at every step in a 3D environment of 1036 features. The odometry of the vehicle has errors in each motion step with a standard deviation of $10cm$ in the x direction (the direction of motion), $5cm$ in y and z directions, and $(1deg, 0.5deg, 0.5deg)$ for Roll, Pitch and Yaw angles. We simulate an onboard range and bearing sensor with a range of $3m$, so that 12 features are normally seen at every step. The measurement error is 2% of the distance in range, and $0.5deg$ in bearing.

We can see that the computational cost of D&C SLAM is very low for most steps compared with standard EKF SLAM, except in those steps that are a power of 2. In those cases, several map joinings may take place at the same step to complete the map. This results in a slightly higher computational cost for D&C compared with EKF. All map joining operations are quadratic on the number of features on the resulting map. However, 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 $O(k^2)$ of this join 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 joinings to be computed at each step. If k is a power of 2 ($k = 2^j$), j joinings will take place at step k, with a cost $O(2^2)+\ldots+O((2^j)^2)$. To carry out the last join in the step, the previous join $j-1$ in the same step should be complete. Thus if we wish to amortize all joins, we must wait until step $k + k/2$ for join $j-1$ to be complete, and then start join j. For this reason, the amortized version of this algorithm is carried out by dividing up the largest join at step k into steps $k + k/2$ to $k + k - 1$ in equal $O(2\,k)$ computations for each step. The next-to-largest join in the step will be divided into steps $k + k/4$ to $k + k/2 - 1$ in equal $O(k)$ computations each, and so on. In this way, the cost of D&C SLAM per step becomes *linear* with n in the amortized version.

Fig. 3 (middle) shows the resulting amortized cost for 256 steps in this simple simulation. Note that at steps $i = 2^j$, the cost falls steeply. As said before, in these steps j joins should be computed, but since join i required the map resulting from join $i-1$, all j joins are postponed. 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. D&C SLAM can then continue normally with this single map in the stack. Fig. 3 (bottom) shows the total execution time of both algorithms. Given that the computational cost per step of D&C SLAM is lower than that of EKF SLAM most of the time, the total cost of D&C SLAM increases very slowly compared to the total cost of EKF SLAM.

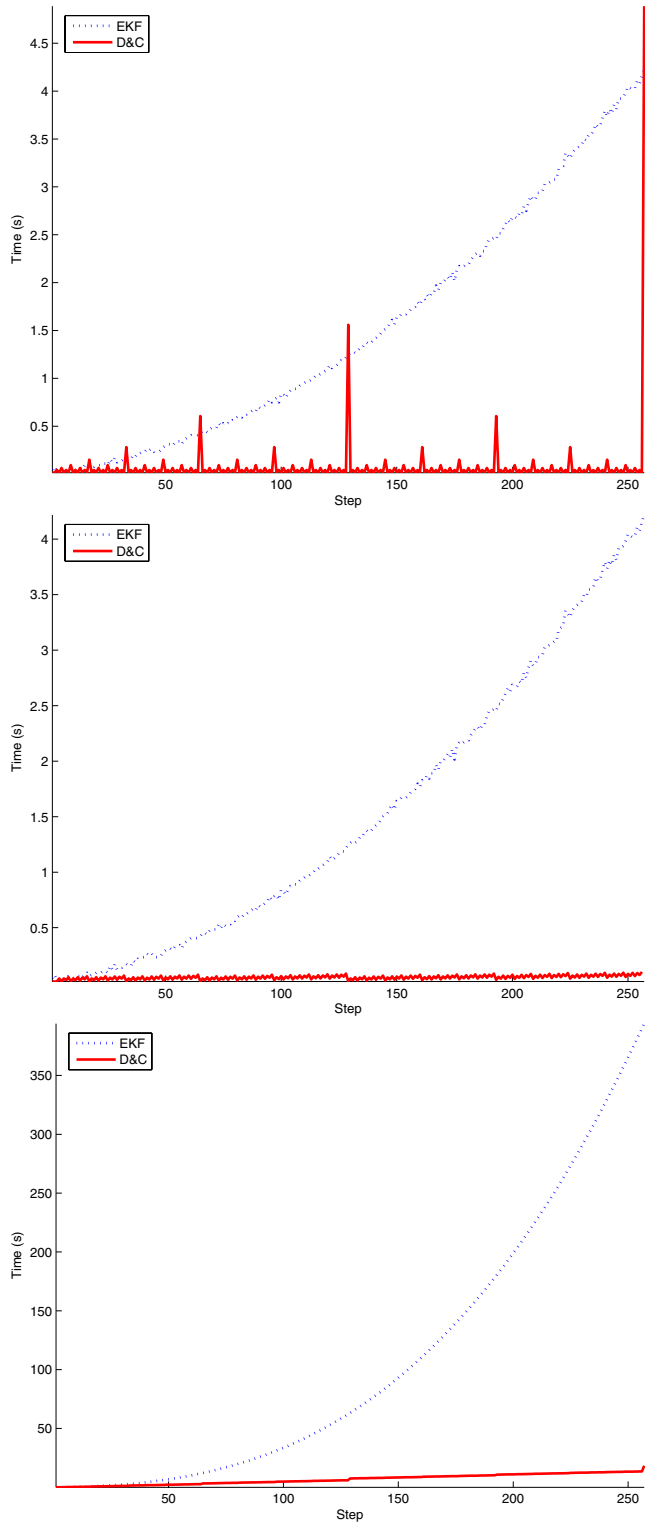

Fig. 3. Cost per step of D&C v.s. EKF SLAM (top); Amortized cost per step of D&C and EKF SLAM (middle). Total execution time of EKF .vs. D&C SLAM (bottom).

III. CONSISTENCY IN D&C SLAM

Consistency analysis determines how well the covariance matrix $\mathbf{P}$ represents the error in estimate $\hat{\mathbf{x}}$.

If the ground truth solution $\mathbf{x}$ for the state variables is available, a statistical test for filter consistency can be carried out on the state estimate $(\hat{\mathbf{x}}, \mathbf{P})$. The Normalized Estimation Error Squared (NEES) is defined as:

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

Consistency is then checked using a chi-squared test:

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

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:

$$CI = \frac{D^2}{\chi^2_{r,1-\alpha}}, \tag{3}$$

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

We tested consistency of both standard EKF and D&C SLAM algorithms by carrying out 20 Monte Carlo runs on the simulated experiment. Figure 4, top left, shows the evolution of the mean consistency index of the vehicle orientation (roll angle) during all steps of the simulation. We can see that the D&C estimate always has a lower consistency index than the standard EKF estimate, and falls out of consistency at a much lower rate. Fig. 4 (bottom, left) shows the mean consistency index for all features in the map. Again, the D&C feature estimates have always a lower consistency index than those of standard EKF SLAM and fall out of consistency at a much lower rate. Note that for D&C SLAM, consistency is pointed out by triangles in those steps which are a power of 2, when a full map is available (although we compute consistency at every step).

Figures 4 (right) show the evolution of the mean absolute roll error of the vehicle (top) and mean absolute lateral error for all features (bottom). 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 that the error increases at a slower rate in the case of D&C SLAM; 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.

We carried out another simulated experiment to test the D&C SLAM consistency when the vehicle moves along a loop trajectory of 64 steps. The robot estimate was computed by joining all maps available in every step. Both D&C SLAM and EKF SLAM were executed with exactly the same data (including random errors). Fig. 5 shows a typical situation: EKF SLAM (top) is overconfident: errors are larger than the computed covariances suggest, and thus loop closing is not possible. D&C SLAM (bottom) computes estimates and covariances that allow it to easily close the loop.

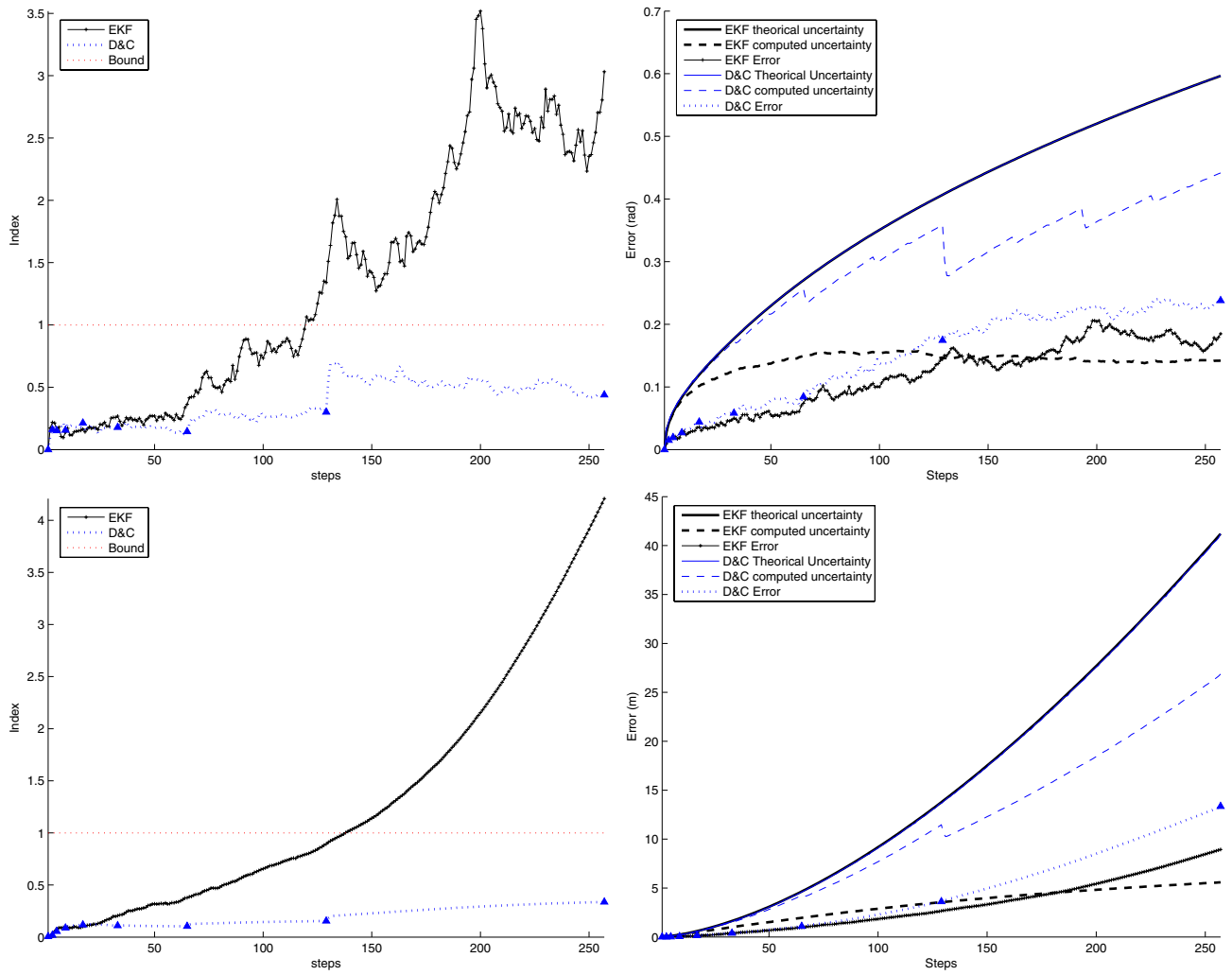

Fig. 4. Mean consistency index (left) and Mean Absolute error (right) for standard EKF SLAM (black) and D&C SLAM (blue). Roll Robot error(top); lateral error for all map features (bottom); in all cases, at all steps of the vehicle trajectory.

IV. CONCLUSIONS

In this paper we propose *Divide and Conquer* SLAM, a computationally more feasible alternative to standard EKF SLAM. Its computational cost per step is $O(n)$, as compared to $O(n^2)$ for standard EKF SLAM. D&C SLAM is a simple algorithm to implement, and 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, both the estimate $\hat{\mathbf{x}}$ and covariance $\mathbf{P}$, with no approximations, and with the additional advantage of providing always a more precise and consistent vehicle and map estimate.

One of the main subjects of our future work will be to study D&C with other simulations experiments (explore and return, loop closing, lawn mowing) to show that the amortized cost is kept linear. Also, future work will be to carry out a large scale experiment where the advantages and limits of D&C SLAM can be experimentally evaluated. Data association is required to carry out the joining process between two local maps. Depending on the type of trajectory, the size and overlap between the maps will change. This will require developing specialized data association algorithms to keep the amortized cost always linear.

We hope to demonstrate that D&C SLAM is the algorithm to use in all applications in which the Extended Kalman Filter solution is to be used.

V. ACKNOWLEDGMENTS

This research has been funded in part by the Dirección General de Investigación of Spain under projects DPI2003-07986 and DPI2006-13578, the Swedish Foundation for International Cooperation in Research and Higher Education (STINT), under project IG2003-2 060, and the Swedish Foundation for Strategic Research through the Centre for Autonomous Systems.

APPENDIX I: MAP JOINING 2.0

This appendix describes the map joining process used in D&C SLAM, an improved version with respect to the original map joining 1.0 in [16]. The general idea is the

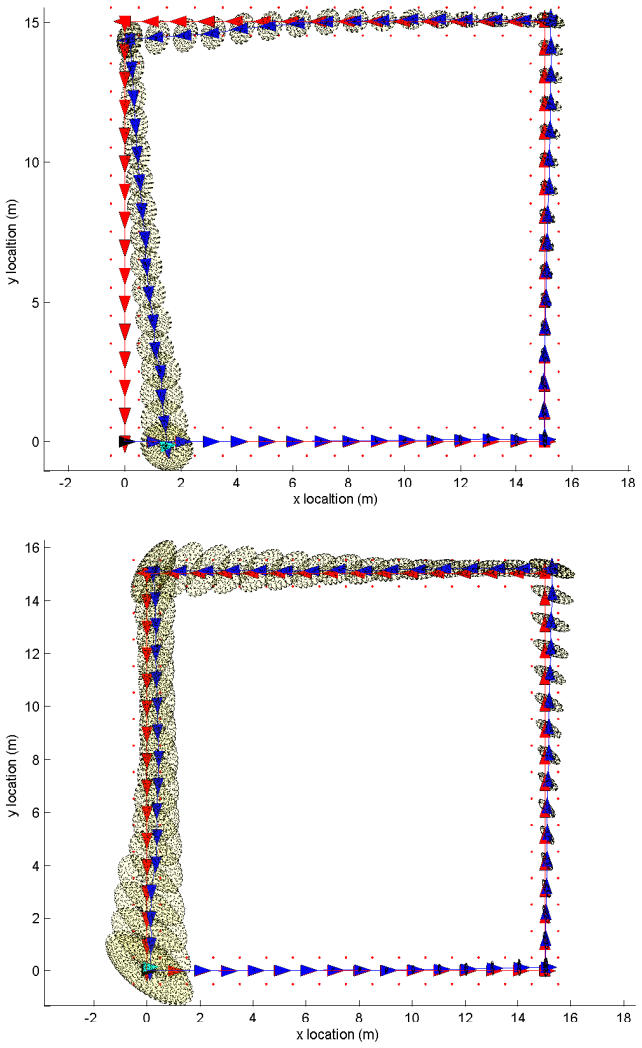

Fig. 5. Conditions on arriving at loop closing: EKF SLAM (top), D&C SLAM (bottom). Ground true trajectory in red, estimated trajectory in blue.

same: in a sequential move-sense-update cycle, a local map is initialized at some moment i using the current vehicle location R_i as base reference, and thus the initial vehicle location in the map is $\mathbf{x}_{R_iR_i} = 0$ an also the initial vehicle uncertainty $\mathbf{P}_{R_i} = 0$. Standard EKF SLAM is carried out in a this move-sense-update fashion, until the map reaches a certain size of n features $F_1 \dots F_n$ at step j. In this moment the state vector $\hat{\mathbf{x}}_{i\dots j}$ will be:

$$\hat{\mathbf{x}}_{i\dots j} = \begin{bmatrix} \hat{\mathbf{x}}_{R_iR_j} \\ \hat{\mathbf{x}}_{R_iF_1} \\ \vdots \\ \hat{\mathbf{x}}_{R_iF_n} \end{bmatrix}$$

with corresponding covariance matrix $\mathbf{P}_{i\dots j}$. This map is then closed, and a new local map $\mathbf{m}_{j\dots k} = (\hat{\mathbf{x}}_{j\dots k}, \mathbf{P}_{j\dots k})$ is initialized in the same way (for simplicity, assume the sensor measurements at step j are used to update the first map, and the vehicle motion from R_j to R_{j+1} is carried out in the second map). This results in having the last vehicle location in the first map, R_j, be the base reference of the second map, which allows maps to be joined into a full map in a three step process of (1) joining; (2) update; and (3) transformation, as it is explained next.

A. The Map Joining step

Consider two sequential local maps $\mathbf{m}_{i\dots j} = (\hat{\mathbf{x}}_{i\dots j}, \mathbf{P}_{i\dots j})$, $\mathbf{m}_{j\dots k} = (\hat{\mathbf{x}}_{j\dots k}, \mathbf{P}_{j\dots k})$, with n features $F_1 \dots F_n$ and m features $G_1 \dots G_m$ each:

$$\hat{\mathbf{x}}_{i\dots j} = \begin{bmatrix} \mathbf{x}_{R_iR_j} \\ \mathbf{x}_{R_iF_1} \\ \vdots \\ \mathbf{x}_{R_iF_n} \end{bmatrix}; \hat{\mathbf{x}}_{j\dots k} = \begin{bmatrix} \mathbf{x}_{R_jR_k} \\ \mathbf{x}_{R_jG_1} \\ \vdots \\ \mathbf{x}_{R_jG_m} \end{bmatrix} \tag{4}$$

In this approach, the *joining step* allows to obtain a stochastic map $\mathbf{m}^-_{i\dots k} = \left(\hat{\mathbf{x}}^-_{i\dots k}, \mathbf{P}^-_{i\dots k}\right)$ in the following simple way:

$$\hat{\mathbf{x}}^-_{i\dots k} = \begin{bmatrix} \hat{\mathbf{x}}_{i\dots j} \\ \hat{\mathbf{x}}_{j\dots k} \end{bmatrix} \tag{5}$$

$$\hat{\mathbf{P}}^-_{i\dots k} = \begin{bmatrix} \mathbf{P}_{i..j} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{j..k} \end{bmatrix} \tag{6}$$

Note that the elements in the second map are kept in their own reference R_j instead of being referenced to reference frame R_i as in map joining 1.0. This has the effect of *delaying* the linearization process of converting all features to base reference R_i until the update step has taken place, and thus an improved estimation is used for this linearization. This is the fundamental difference between map joining 1.0 and map joining 2.0

B. The update step

Data association is carried out to determine correspondences between features coming from the first and second map. This allows to refine the vehicle and environment feature locations by the EKF update step on the state vector. Let $\mathcal{H}$ be a hypothesis that pairs r features $F_{f_1} \dots F_{f_r}$ coming from local map $\mathbf{m}_{i\dots j}$ with features $G_{g_1} \dots G_{g_r}$ coming from map $\mathbf{m}_{j\dots k}$. A modified ideal measurement equation for r re-observed features expresses this coincidence:

$$\mathbf{h}_{\mathcal{H}}(\hat{\mathbf{x}}^-_{i\dots k}) = \begin{bmatrix} \mathbf{h}_{f_1,g_1} \\ \vdots \\ \mathbf{h}_{f_r,g_r} \end{bmatrix} = \mathbf{0} \tag{7}$$

where for each pairing:

$$\mathbf{h}_{f_r,g_r} = \mathbf{x}_{R_iF_{f_r}} - \mathbf{x}_{R_iR_j} \oplus \mathbf{x}_{R_jG_{g_r}}.$$

Linearization yields:

$$\mathbf{h}_{\mathcal{H}}(\hat{\mathbf{x}}^-_{i\dots k}) \simeq \mathbf{h}_{\mathcal{H}}(\hat{\mathbf{x}}^-_{i\dots k}) + \mathbf{H}_{\mathcal{H}}(\mathbf{x}^-_{i\dots k} - \hat{\mathbf{x}}^-_{i\dots k}) \tag{8}$$

where:

$$\mathbf{H}_{\mathcal{H}} = \frac{\partial \mathbf{h}_{\mathcal{H}}}{\partial \mathbf{x}_{i\ldots k}^{-}}|_{(\hat{\mathbf{x}}_{i\ldots k}^{-})} = \begin{bmatrix} \frac{\partial \mathbf{h}_{f_1 g_1}}{\partial \mathbf{x}_{R_i R_j}} & \mathbf{0} & \cdots & \mathbf{I} & \mathbf{0} & \frac{\partial \mathbf{h}_{f_1 g_1}}{\partial \mathbf{x}_{R_j G_{g_1}}} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \mathbf{h}_{f_r g_r}}{\partial \mathbf{x}_{R_i R_j}} & \mathbf{0} & \mathbf{I} & \cdots & \mathbf{0} & \cdots & \frac{\partial \mathbf{h}_{f_r g_r}}{\partial \mathbf{x}_{R_j G_{g_r}}} \end{bmatrix} \quad (9)$$

The update step allows to obtain a new estimate $\mathbf{m}_{i\ldots k}^{+} = \left(\hat{\mathbf{x}}_{i\ldots k}^{+}, \mathbf{P}_{i\ldots k}^{+}\right)$ by applying modified EKF update equations:

$$\begin{aligned} \hat{\mathbf{x}}_{i\ldots k}^{+} &= \hat{\mathbf{x}}_{i\ldots k}^{-} - \mathbf{K}\mathbf{h}_{\mathcal{H}}(\hat{\mathbf{x}}_{i\ldots k}^{-}) \\ \mathbf{P}_{i\ldots k}^{+} &= (\mathbf{I} - \mathbf{K}\mathbf{H}_{\mathcal{H}})\mathbf{P}_{i\ldots k}^{-} \end{aligned}$$

where:

$$\mathbf{K} = \mathbf{P}_{i\ldots k}^{-}\mathbf{H}_{\mathcal{H}}^{T}\left(\mathbf{H}_{\mathcal{H}}\mathbf{P}_{i\ldots k}^{-}\mathbf{H}_{\mathcal{H}}^{T}\right)^{-1}$$

C. The transformation step

A final step is carried out to transform all the elements of $\hat{\mathbf{x}}_{i\ldots k}^{+}$ to the same base reference R_i and obtain the final joined map $\mathbf{m}_{i\ldots k} = (\hat{\mathbf{x}}_{i\ldots k}, \mathbf{P}_{i\ldots k})$:

$$\begin{aligned} \hat{\mathbf{x}}_{i\ldots k} &= \begin{bmatrix} \hat{\mathbf{x}}_{R_i R_k} \\ \hat{\mathbf{x}}_{R_i F_1} \\ \vdots \\ \hat{\mathbf{x}}_{R_i F_n} \\ \hat{\mathbf{x}}_{R_i G_1} \\ \vdots \\ \hat{\mathbf{x}}_{R_i G_m} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_{R_i R_j}^{+} \oplus \hat{\mathbf{x}}_{R_j R_k}^{+} \\ \hat{\mathbf{x}}_{R_i F_1}^{+} \\ \vdots \\ \hat{\mathbf{x}}_{R_i F_n}^{+} \\ \hat{\mathbf{x}}_{R_i R_j}^{+} \oplus \hat{\mathbf{x}}_{R_j G_1}^{+} \\ \vdots \\ \hat{\mathbf{x}}_{R_i R_j}^{+} \oplus \hat{\mathbf{x}}_{R_j G_m}^{+} \end{bmatrix} \\ \mathbf{P}_{i\ldots k} &= \frac{\partial \hat{\mathbf{x}}_{i\ldots k}}{\partial \hat{\mathbf{x}}_{i\ldots k}^{+}} \mathbf{P}_{i\ldots k}^{+} \left(\frac{\partial \hat{\mathbf{x}}_{i\ldots k}}{\partial \hat{\mathbf{x}}_{i\ldots k}^{+}}\right)^{T} \\ \frac{\partial \hat{\mathbf{x}}_{i\ldots k}}{\partial \hat{\mathbf{x}}_{i\ldots k}^{+}} &= \begin{bmatrix} \frac{\partial \mathbf{x}_{R_i R_k}}{\partial \mathbf{x}_{R_i R_j}} & \mathbf{0} & \frac{\partial \mathbf{x}_{R_i R_k}}{\partial \mathbf{x}_{R_j R_k}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \frac{\partial \mathbf{x}_{R_i E}}{\partial \mathbf{x}_{R_i R_j}} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{x}_{R_i E}}{\partial \mathbf{x}_{R_j E}} \end{bmatrix} \end{aligned} \quad (10)$$

Note again that this linearization is carried out once the map has been refined in the previous update step, thus using a better estimate.

REFERENCES

[1] T. Bailey, J. Nieto, J. Guivant, M. Stevens and E. Nebot, *"Consistency of the EKF SLAM Algorithm"*, IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006.

[2] Y. Bar-Shalom, X. Rong Li and T. Kirubarajan, *"Estimation with Applications to Tracking and Navigation"*, Wiley InterScience, 2001.

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

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

[5] J.A. Castellanos, R. Martinez-Cantin, J.D. Tardós and J. Neira, *"Robocentric Map Joining: Improving the Consistency of EKF SLAM"*, to appear in Robotics and Autonomous Systems.

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

[7] J. Folkesson and H. Christensen, *Graphical SLAM - A Self-Correcting Map*, Proc. of the IEEE International Conference on Robotics and Automation (ICRA'04), New Orleans, LA, USA, 2004.

[8] U. Frese, *"Treemap: An $O(logn)$ Algorithm for Indoor Simultaneous Localization and Mapping"*, Autonomus Robots, 21(2) pp. 103-122, 2006.

[9] J. Guivant and E. Nebot, *"Optimization of the Simultaneous Localization and Map-Building Algorithm for Real-Time Implementation"*, IEEE Transactions on Robotics and Automation, 17(3) pp. 242-257, 2001.

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

[11] S.J. Julier, J.K. Uhlmann, J.K. A Counter Example to the Theory of Simultaneous Localization and Map Building 2001 IEEE Int. Conf. on Robotics and Automation, 2001, 4238-4243

[12] J. Knight, A. Davison and I. Reid, *"Towards Constant Time SLAM using Postponement"* IEEE/RSJ Int'l Conf on Intelligent Robots and Systems, pp 406-412, 2001.

[13] R. Martinez-Cantin and J. A. Castellanos, *"Unscented SLAM for large-scale outdoor environments"*, 2005 IEEE/RSJ Int. Conference on Intelligent Robots and Systems, IROS'05, Edmonton, Alberta, Canada, pp. 328-333.

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

[15] L. Paz and J. Neira, *"Optimal local map size for EKF-based SLAM"*, IEEE/RSJ Int. Conference on Intelligent Robots and Systems, IROS'06, Beijing, China.

[16] J. Tardós, J. Neira, P. Newman and J. Leonard, *"Robust Mapping and Localization in Indoor Environments using Sonar Data"* Int. J. Robotics Research, 21, 311-330, 2002.

[17] 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, 23, 693-716, 2004.

[18] M. Walter, R. Eustice and J. Leonard, *"A Provably Consistent Method for Imposing Sparsity in Feature-based SLAM Information Filters"*, Proc. of the Int. Symposium of Robotics Research (ISRR), 2004.