2009 IEEE International Conference on Robotics and Automation
Kobe International Conference Center
Kobe, Japan, May 12-17, 2009

Finding good cycle constraints for large scale multi-robot SLAM

Carlos Estrada, José Neira and Juan D. Tardós

Abstract— **In this paper we describe an algorithm to compute cycle constraints that can be used in many graph-based SLAM algorithms; we exemplify it in Hierarchical SLAM. Our algorithm incrementally computes the minimum cycle basis of constraints from which any other cycle can be derived. Cycles in this basis are local and of minimum length, so that the associated cycle constraints have less linearization problems. This also permits to construct regional maps, that is, it makes possible efficient and accurate intermediate mapping levels between local maps and the whole global map. We have extended our algorithm to the multi-robot case. We have tested our methodology using the Victoria Park data set with satisfactory results.**

I. INTRODUCTION

Graph-based SLAM is one of the three fundamental paradigms for solving SLAM [28]. Frese's Treemap algorithm [10] uses a binary-tree to divide the map into regions and subregions to compute nonlinear map estimates. It assumes, as most of the following graph-based methods, known data association and requires a suitable building topology. Graphical SLAM [6] compresses the graph into star nodes. These star nodes are used to perform a coarse adjustment of the graph in a similar manner as Hierarchical SLAM [4] imposes cycle constraints [8]. They perform afterwards a fine tuning to eliminate inconsistency between features common to different star nodes. In their experiments the graph relaxation is quite hard, resulting in computation times significantly longer that EKF [7]. As the topology of the graph becomes more complex (many cycles) it takes longer to relax the graph. GraphSLAM [27] transforms the SLAM posterior into a graphical network, representing the log-likelihood of the data. It then reduces this graph using variable elimination techniques, arriving at a lower dimensional problem that is then solved using conventional optimization techniques. When closing a cycle, GraphSLAM reduces the number of variables to the path variables (just like Graphical SLAM). Konolige's method for Large-Scale Map-making [19] reduces a pose chain graph to poses that have a cycle constraint attached by marginalizing all poses not directly involved in the cycle constraints. Tectonic SAM [22] uses a submap-based SLAM. It employs what they call a factor graph that then is compressed in a way very similar to Graphical SLAM. It makes use of a smoothing approach to optimize every single submap and a Levenberg-Marquardt optimization to align the submaps. Incremental SAM (iSAM) [16] is an incremental smoothing and mapping method that exploits the natural sparsity of the smoothing information matrix. Reordering of the variables and factorization of the new measurement Jacobian are not done at any step but periodically to achieve real time. The method uses conservative estimates for online data association. Sophisticated optimization techniques such as Multilevel relaxation of Frese [9] or Olson's algorithm [23] use the incremental sequence of robot poses along the travelled path to perform the optimization, making the algorithm depend on the length of the path and difficult to apply to the scenario of multi-robot SLAM. Grisetti et al. [12] presented an extension of Olson's Stochastic Gradient Descent algorithm that, using a tree parametrization that takes into account the topology of the environment, defines and efficiently updates local regions in each iteration. An online version of this method is achieved by integrating adaptive learning rates in the parametrization of the network [13].

The problem of nonlinearities makes graph-based methods using relative coordinates more precise to perform SLAM than graph-based methods using absolute coordinates. However, most of the above graph-based methods working with relative coordinates assume that the constraints are given. If, on the contrary, the constraints are obtained by the method, nothing is said about classifying them to avoid redundancy of constraints. The method that we propose in the present paper is based in the Hierarchical SLAM [4] representation and achieves, not only a classification of the constraints, but an incremental classification that allows to decompose the global map into regions that accelerate the obtention of a regional map when we are not interested in the whole global map (countries, cities, etc) but only in a reduced part of it.

The Hierarchical SLAM method obtains accurate metric maps of large environments in real time by using two levels of representation: a lower (or local) level of statistically independent submaps and an upper (or global) level that, using an adjacency graph, maintains a relative stochastic map between the local map references (see Figure 1). Due to its graph-based upper level structure and data association capabilities, the Hierarchical SLAM method is an excellent candidate for implementing the new methodology that we describe in this paper.

To generalize the algorithm proposed we have extended it to the multi-robot case. Several approaches to the multi-robot SLAM problem can be found in the literature. Some of them make the assumption that the relative initial pose of the robots is known from the beginning as in [26]. Others assume that, although the relative initial poses are not known, each

This work was supported by the European Union under Project RAWSEEDS FP6-IST-045144 and by the Dirección General de Investigación of Spain under Project DPI2006-13578.

The authors are with the Dept. Informática e Ingeniería de Sistemas and the Inst. de Investigación en Ingeniería de Aragón, Universidad de Zaragoza, c/María de Luna 1, 50018 Zaragoza, Spain. Email: {cestrada, jneira, tardos}@unizar.es.

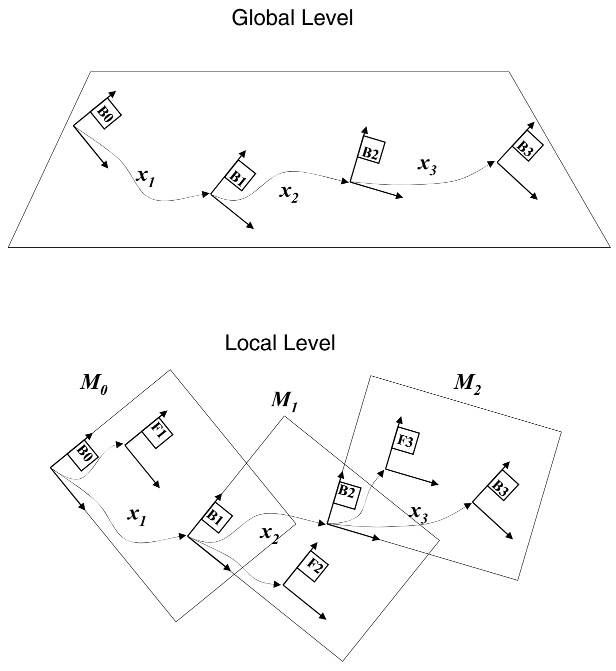

Fig. 1. Two level hierarchical SLAM model.

robot starts within the map of another robot [25], [5]. Finally, when there is no knowledge of the pose of a robot within other robot's map, we can recur to rendezvous strategies as in [18] or search for map correspondences as in [29]. We have decided to adopt this last strategy to make the method as general as possible. Some examples of multi-robot SLAM using a constraint graph have appeared recently, as C-SAM [1], a multi-robot extension of Tectonic SAM using relative pose observations to connect maps from different robots, or as the Topological/Metric approach of [3], where the map fusion is represented in the graph by adding a link between the corresponding nodes using the observations of relative robot poses. As in the single-robot approach, none of the multi-robot methods that use relative coordinates make any mention about constraints classification.

This paper is organized as follows: after the introduction in section I, we analyze, in section II, the treatment of constraints in the upper level of the Hierarchical SLAM, proposing an incremental minimum cycle basis algorithm to organize them. We describe our algorithm in section III. In Section IV we extend our algorithm to the multi-robot case. Section V describes the experiment employed to validate our approach. Finally, in section VI we draw the main conclusions of this work and outline the future research directions.

II. Treatment of constraints in the upper level of the Hierarchical SLAM

A. Selection of constraints to avoid redundancy

When a cycle formed by a sequence of n local maps is detected by the Hierarchical SLAM method the following constraint is imposed in the global level:

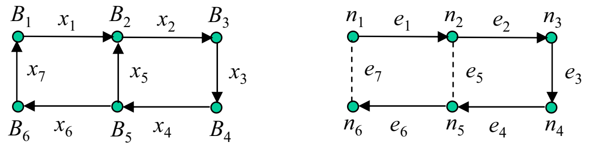

Fig. 2. Example of the global level after closing two cycles.

$$\mathbf{h}(\mathbf{x}) \equiv \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \cdots \oplus \mathbf{x}_{n-1} \oplus \mathbf{x}_n = \mathbf{0} \tag{1}$$

As more cycles are closed more constraints can be imposed in the global level. Figure 2a shows an example of a possible situation after a closing of two cycles. We can observe in this figure that closing the two cycles generates three possible constraints:

$$\begin{aligned} &\mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \mathbf{x}_3 \oplus \mathbf{x}_4 \oplus \mathbf{x}_6 \oplus \mathbf{x}_7 = \mathbf{0} \\ &\mathbf{x}_1 \ominus \mathbf{x}_5 \oplus \mathbf{x}_6 \oplus \mathbf{x}_7 = \mathbf{0} \\ &\mathbf{x}_2 \oplus \mathbf{x}_3 \oplus \mathbf{x}_4 \oplus \mathbf{x}_5 = \mathbf{0} \end{aligned} \tag{2}$$

where $\oplus$ represents composition of transformations and $\ominus$ composition with the inverse.

However, one of the three constraints is redundant, as it can be obtained using the other two. For instance, the first constraint can be obtained by composing the second and third constraints:

$$\mathbf{x}_1 \oplus (\mathbf{x}_2 \oplus \mathbf{x}_3 \oplus \mathbf{x}_4 \oplus \mathbf{x}_5) \ominus \mathbf{x}_5 \oplus \mathbf{x}_6 \oplus \mathbf{x}_7 = \mathbf{0} \tag{3}$$

The problem of redundancy of constraints can be easily treated using graph theory. In Figure 2b we have drawn a graph whose edges $e_1, e_2, \cdots, e_7$ correspond to the transformations $\mathbf{x}_1, \mathbf{x}_1 \cdots, \mathbf{x}_7$ and whose nodes $n_1, n_2 \cdots, n_6$ correspond to the local map references $B_1, B_2, \cdots, B_6$. A tree is a connected subgraph containing no cycles. If it contains all the nodes of the graph it is called a spanning tree. Chords are all the edges of the connected graph that don't belong to the spanning tree. In our example, the edges of the graph can be classified into edges (or branches) of a spanning tree $(e_1, e_2, e_3, e_4, e_6)$ and chords (e_5, e_7). The spanning tree of a directed graph can be used to obtain the absolute references of the local maps with respect to the origin, defined by the reference of the map associated to the tree root (in our case B_1). We can represent the cycles of a graph by means of a vector whose rows are associated to the edges of the graph: 1 if the cycle contains the edge and 0 otherwise. One constraint is redundant if its associated cycle can be expressed as a composition, using the xor operation $(+)$, of other cycles. In our example, if the $i-th$ row represents e_i, the cycle associated to the first constraint can be obtained by the composition of the cycles associated to the second and third constraints:

$$\begin{bmatrix}1\\0\\0\\0\\1\\1\\1\end{bmatrix}+\begin{bmatrix}0\\1\\1\\1\\1\\0\\0\end{bmatrix}=\begin{bmatrix}1\\1\\1\\1\\0\\1\\1\end{bmatrix} \tag{4}$$

The maximal set of linearly independent cycles is called a cycle basis. Every cycle of a graph can be expressed as a linear combination of the cycles that form a cycle basis. The number of cycles that belong to a cycle basis is $N = m - n + 1$, where m is the number of edges, and n the number of nodes of the graph.

Adding a chord to a spanning tree of a connected graph produces a unique cycle in the graph, called a fundamental cycle. The set of fundamental cycles with respect to a spanning tree are easy to construct and form a cycle basis called fundamental cycle basis. In Figure 2b the fundamental cycle basis will be $\{e_7e_1e_2e_3e_4e_6, e_5e_2e_3e_4\}$. Another possible cycle basis is the minimum cycle basis, that is, the cycle basis of minimum weight in the graph. We have weighted the edges of a graph with the distances between local map references. Assuming that the length of the edges of the graph of Figure 2b is proportional to their weights, the minimum cycle basis will be $\{e_7e_1e_5e_6, e_5e_2e_3e_4\}$.

In practical situations we are only interested in mapping a specific part of a global map, what we call a regional map, instead of the whole of it. For this purpose we propose to use the minimum cycle basis to obtain the set of independent constraints in the upper level. To build an accurate regional map we only need to apply the Hierarchical SLAM method with the constraints associated to that region. Another reason for this decision is that nonlinearities play an increasing role as cycles are longer in length: orientation uncertainty can be very large in a long cycle, producing large linearization errors [15].

B. Algorithms to obtain the minimum cycle basis of a connected graph

The problem of computing a cycle basis of minimum weight in a graph was solved in polynomial time for the first time by Horton [14] with running time $O(m^3n)$. Using a different approach de Pina [24], and recently Berger et al. [2], solved the problem in $O(m^3 + mn^2logn)$. For very large and sparse graphs the running time of the algorithm is $O(n^3logn)$. Experimental findings have shown [20] that the true bottleneck of a sophisticated implementation of de Pina's algorithm, which can be decomposed into two parts with time $O(m^3)$ and $O(m^2n + mn^2logn)$ respectively, is the $O(m^2n+mn^2logn)$ part. Golynski and Horton [11] used fast matrix multiplication to improve Horton's algorithm to $O(m^wn)$, where $w < 2.376$. Kavitha et al. [17], using also fast matrix multiplication, improved de Pina's algorithm into $O(m^2n + mn^2logn)$. More recently Mehlhorn and Michail [21] achieved an $O(m^2n/logn + n^2m)$ algorithm.

However, all the above algorithms have in common that they solve the minimum cycle basis problem in batch mode using all the nodes and edges of the graph and don't take advantage of the incremental construction of the graph inherent to our SLAM methodology. We propose an incremental algorithm to obtain the minimum cycle basis of the connected graph associated to the upper level of the Hierarchical SLAM method. Our algorithm only needs to check for an update of the cycle basis when there is a fusion between local maps and, what is more important, the update process is maintained locally as changes in the minimum cycle basis affect only cycles around the fused maps.

III. INCREMENTAL MINIMUM CYCLE BASIS ALGORITHM

When a robot traverses the environment using the Hierarchical SLAM method, a graph associated to the relative transformations between the local map references is maintained. If the graph has any cycles, the minimal cycle basis of the graph will be used to impose the constraints in the upper level as explained in the previous section. A complete representation of the minimum cycle basis consists on the enumeration, for each cycle of all its chords and branches, in the consecutive order in which they appear starting from an arbitrary edge. Apart from this we will also have as in [17] a simplified representation of a cycle basis that consist of two matrices: cycles matrix *C* and witnesses matrix *S*, both of dimension $N \times N$, where N is the number of cycles in the basis. Each matrix is formed by a set of N column vectors C_j and S_j in the space $\{0,1\}^N$ spanned by the set of all the chords of the graph $Ch = \{ch_1, ..., ch_N\}$. A cycle vector C_j has 1 in row i if chord ch_i forms part of the associated cycle and 0 otherwise. A witness S_j is a non-zero vector in the subspace orthogonal to $\{C_1, ..., C_{j-1}\}$ and not orthogonal to C_j. A witness S_j is orthogonal to a cycle C_k if the standard inner product $\langle S_j, C_k\rangle = 0$. This inner product is defined using an AND as the product operator and a XOR as the sum operator.

When a local map is incorporated to the global map it produces the addition of a new node and a new branch to the graph. This operation, although it modifies the graph, does not modify the minimum cycle basis of the graph. The only operation that may modify the cycle basis of the graph is the fusion between maps. The fusion between maps occurs, in our implementation, when two local maps with a high degree of overlapping are matched and its fusion is geometrically consistent with the global map. This situation arrives when revisiting previous mapping areas or when, after an optimization process of the Hierarchical SLAM method, new matchings between maps are detected. As each node of the graph represents a unique map, the fusion of two local maps will produce in the graph the fusion of the two nodes representing the maps. We have defined a sequence of steps in our algorithm to update the graph and the minimum cycle basis after the fusion of two maps. We will describe these steps in the following subsections using the example of Figure 3. During the intermediate steps of the process

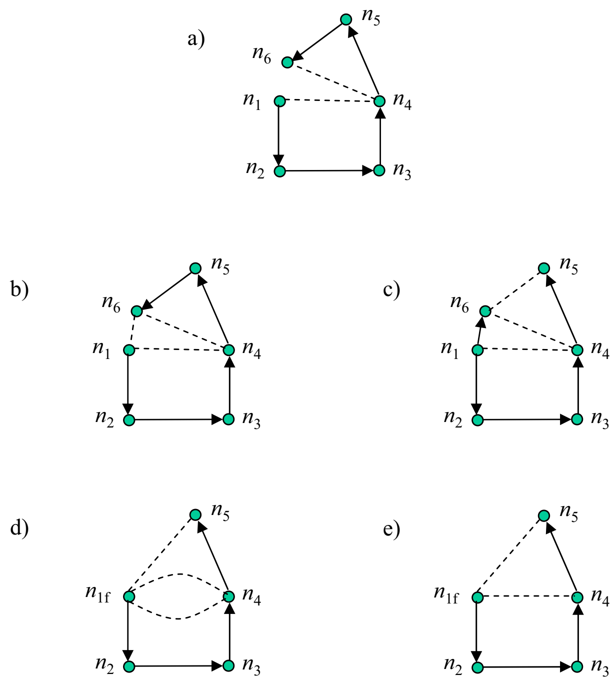

Fig. 3. Process of node fusion

we may have a cycle basis that is not a minimum cycle basis. However, at the end of the last step, we will obtain the minimum cycle basis. New matrices C and S resulting from the process will verify the orthogonality conditions mentioned above. Although we will focus mainly in the cycle basis update it is understood that simultaneously to the cycle basis we also update the graph representation.

$\langle C_i', S_i\rangle = 1$ is equivalent to say that cycle C_i' has odd intersection with S_i. Every time a pair C_i, S_i is updated in matrices C and S we will obtain, using Dijkstra's algorithm, all the nodes n_j from the graph whose minimum distance to at least one of the nodes from the chords of S_i, $\mathrm{d_{min}}(n_j, S_i)$, is less than length($C_i$). This set of nodes, $\mathrm{NodeSet}(i)$, contains all the nodes of possible cycles C_i' that verify simultaneously length(C_i')<length(C_i) and $\langle C_i', S_i\rangle = 1$. However, we are interested in the inverse function, $\mathrm{CycleSet}(n_j)$: for each node n_j of the graph we want to obtain all the cycles i from C that verify $n_j \in \mathrm{NodeSet}(i)$. We will use $\mathrm{NodeSet}(i)$ to update CycleSet. We will also store the minimum distances in $\mathrm{DSet}(i, n_j)$. The purpose of CycleSet and DSet is to maintain locality at the final step of the algorithm. As we can stop Dijkstra's algorithm when distances are greater than length(C_i) the algorithm is locally bounded.

A. Adding a chord between two nodes.

When there is a fusion between two maps represented by two nodes from a graph we have two situations: either the two nodes are previously connected to each other or not. If they are not connected, our algorithm proceeds to connect them by adding a chord between the two nodes. This situation is represented in Figure 3a and Figure 3b for nodes n_1 and n_6.

Adding a chord ch_i between two nodes increments the number of cycles of our base by one. This update of the cycle basis is done by adding a cycle orthogonal to the previous ones. The fundamental cycle associated to chord ch_i satisfies this condition and will be added to the cycle basis. After this operation the cycle basis may stop being a minimum cycle basis. If the previous minimum cycle basis is represented by chords Ch^1 and matrices C^1 and S^1, the new cycle basis is obtained as follows:

$$Ch = \{Ch^1, ch_i\} \tag{5}$$

$$C = \begin{bmatrix} C^1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6}$$

$$S = \begin{bmatrix} S^1 & 0 \\ 0 & 1 \end{bmatrix} \tag{7}$$

If the two nodes are connected (or we have connected them using the previous procedure) we have again two situations: the two nodes are connected either by a chord or by a branch.

B. Edge-swapping between a chord and a branch.

If they are connected by a chord, say ch_i, we first must locate which of the two nodes is further from the origin of the graph, and then perform an edge-swapping between chord ch_i and branch b_m defined by the furthest node and its father. Following with our example chord ch_i corresponds to edge n_1n_6 and branch b_m to edge n_5n_6 in Figure 3b, as the furthest node from the origin is n_6. Edge swapping between these two edges gives the situation represented in Figure 3c.

Although edge-swapping doesn't change the cycle basis itself, its representation by means of Ch, C and S changes. Supposing that the cycle basis previous to edge-swapping is represented by chords Ch^1 and matrices C^1 and S^1, the new cycle basis is obtained as follows:

We will update the set of chords Ch^1 by substituting chord ch_i with branch b_m, now transformed into a chord. The new matrix C will be the same as C^1 except for the i-th row: $C(i,j) = 1$ if branch b_m belonged to the cycle represented by C_j^1 and 0 otherwise. A change in row i of cycle C_j^1 is equivalent to the following update:

$$C_j = C_j^1 + FC_i \tag{8}$$

where FC_i is the simplified representation of the fundamental cycle associated to chord i:

$$FC_i(i) = 1 \ ; \ FC_i(j) = 0 \ \ \forall j \neq i \tag{9}$$

We will also construct column vector S_k^1 using witnesses from all the cycles $C_{k1}^1, ..., C_{kn}^1$ that have been updated using FC_i:

$$S_k^1 = S_{k1}^1 + \cdots + S_{kn}^1 \tag{10}$$

Matrix S will be obtained from S^1 by the following operations:

When $\langle FC_i, S_j^1\rangle = 1$:

$$S_j = S_j^1 + S_k^1 \tag{11}$$

When $\left\langle FC_i, S_j^1 \right\rangle = 0$, $S_j = S_j^1$.

C. Branch contraction.

If the two nodes to be fused are connected by a branch (or we have connected them this way using the previous procedure) we proceed to the contraction of the branch that joins them. Contraction of edge $n_1 n_6$ of Figure 3c into node n_{1f} produces the graph represented in 3d. The contracted branch will be eliminated from every cycle of the basis that has it, reducing by one the number of edges of the cycle. Suppose that before the contraction the cycle basis is represented by chords Ch^1 and matrices C^1 and S^1. Any chord of Ch^1 that has at one of its extremes one of the two nodes involved in the fusion will update the label associated to this node to the new node label. Matrices C^1 and S^1 will remain the same unless cycles of two edges appear in the graph (two nodes linked between them with more than one edge, what we call a degenerate cycle). This is the situation of Figure 3d where a degenerate cycle formed by two chords has appeared between nodes $n_{1f} n_4$ as a consequence of the branch contraction. If this situation arrives we shall proceed to eliminate every degenerate cycle, leaving only one edge connecting every two nodes. The two edges of a degenerate cycle are either two chords or a branch and a chord. Depending on the case we will use one of the following procedures.

D. Elimination of a chord-chord cycle.

If we have two chords between two nodes we must eliminate one of them and a cycle from the basis. This operation is done with the graph represented in Figure 3d producing the final graph of Figure 3e. Suppose the cycle basis is represented by Ch^1, C^1 and S^1. We can locate the rows associated to the chords using Ch^1, as two chords of this set will have the same extreme nodes. Suppose that these chords are located at positions i and j in Ch^1 and that we decide, arbitrarily, to eliminate the chord associated to row i. First we must locate in C^1 the column k associated to the double chord cycle with chords i and j. This column will satisfy the following criteria:

$$\left\langle DC_{ij}, S_k^1 \right\rangle = 1 \ ; \ \left\langle DC_{ij}, S_m^1 \right\rangle = 0 \ \ \forall m > k \tag{12}$$

where $DC_{ij} = FC_i + FC_j$ is the double chord cycle with chords i and j.

We will update columns of C^1, changing chord i to chord j in all cycles m where $C^1(i,m) = 1$, $m \neq k$:

$$C_m^2 = C_m^1 + DC_{ij} \tag{13}$$

The rest of rows and columns of C^2 remain the same as in C^1. The same update will be performed in the complete representation of the cycles.

After the update we will look in S^1 for columns m, $m < k$, where

$$\left\langle DC_{ij}, S_m^1 \right\rangle = 1 \tag{14}$$

and update them as follows:

$$S_m^2 = S_m^1 + S_k^1 \tag{15}$$

The rest of the columns of S^1 will remain the same in S^2.

Next we eliminate chord i from Ch^1 and row i and column k from C^2 and S^2 to produce the final Ch, C and S. We also eliminate cycle k from the complete representation of the cycles.

E. Elimination of a branch-chord cycle.

If we have a branch and a chord between two nodes (we detect it by checking the graph) we must eliminate the chord and a cycle from the basis. Suppose the cycle basis is represented by Ch^1, C^1 and S^1. First we must locate the position of the chord in Ch^1: position i. Next we must locate the column k in C^1 associated to the fundamental cycle of chord i. Column k must satisfy:

$$\left\langle FC_i, S_k^1 \right\rangle = 1 \ ; \ \left\langle FC_i, S_j^1 \right\rangle = 0 \ \ \forall j > k \tag{16}$$

We substitute chord i for a branch (if the branch doesn't exist yet) in all the cycles of the complete representation except in cycle k. If $C^1(i,j) = 1$, $j \neq k$ we eliminate chord i from cycle j using (8) and assign to the other elements of C^2 the same value they had in C^1.

Matrix S^2 will be obtained from S^1 doing the following operations:

When $\left\langle FC_i, S_j^1 \right\rangle = 1$, $j < k$, $S_j^2 = S_j + S_k$.

When $\left\langle FC_i, S_j^1 \right\rangle = 0$, $S_j^2 = S_j^1$.

We don't modify column k of S^1: $S_k^2 = S_k^1$, as we are going to delete it.

Next we eliminate chord i from Ch^1 and row i and column k from C^2 and S^2 to produce the final Ch, C and S. We also eliminate cycle k from the complete representation of the cycles.

F. Calculation of the minimum cycle basis after the fusion.

Fusion of nodes n_1 and n_2 to produce node n_f may add, in the case nodes n_1 and n_2 weren't previously connected, new cycles that contain n_f to the cycle space of the graph and, in any case, may modify the length of cycles that contain n_f, as distance between n_f and their adjacent nodes may have changed with the contraction of $n_1 n_2$. This means that cycles that may modify the cycle basis obtained in the previous subsections are only cycles that contain node n_f.

Before proceeding with the last part of the algorithm we need to construct $\mathrm{CycleSet}(n_f)$ for n_f. For that purpose we will find the nodes connected to n_f, say n_j, and obtain, using these nodes, distances from S_i to n_f, $\mathrm{d}(i, n_f)$:

$$\mathrm{d}(i, n_f) = \min_j(\mathrm{DSet}(i, n_j) + d_{jf}) \tag{17}$$

where d_{jf} is the graph distance from node n_j to node n_f and $i = \mathrm{CycleSet}(n_j)(k)$ for a certain k.

If $\mathrm{d}(i, n_f) < \mathrm{length}(C_i)$ we will add $\mathrm{CycleSet}(n_j)(k)$ to $\mathrm{CycleSet}(n_f)$ and set $\mathrm{DSet}(i, n_f) = \mathrm{d}(i, n_f)$. If there is any chord adjacent to n_f we will search for all S_i that contain that chord, add i to $\mathrm{CycleSet}(n_f)$ and set $\mathrm{DSet}(i, n_f) = 0$. As n_f may have established a new path between their adjacent nodes we must propagate $\mathrm{CycleSet}(n_f)$ in a radial way using a similar approach until no more actualization of $\mathrm{CycleSet}$ is detected.

To update the minimum cycle basis after the fusion we have developed an incremental algorithm, derived from de Pina's batch algorithm, that uses as inputs matrices $C = [C_1 \cdots C_N]$ and $S = [S_1 \cdots S_N]$ obtained using the rules of the previous subsections, being N the final number of chords of the graph after the fusion. It also uses $\mathrm{CycleSet}(n_f)$ as input to maintain locality. $\mathrm{CycleSet}$ is updated in the form explained at the beginning of the section each time a pair C_i, S_i is changed.

The algorithm proposed is:

```
$\mathrm{Candidates} = \{\}$.
For $i \in \mathrm{CycleSet}(n_f)$ or with $\mathrm{Candidates}\{i\} \neq \{\}$
    If $i \in \mathrm{CycleSet}(n_f)$ then
        Find the shortest cycle $C_a$ that contains $n_f$
        and satisfies $\langle C_a, S_i \rangle = 1$.
    end
    Select cycle $C_b$ with minimum length
    from $\mathrm{Candidates}\{i\}$ that
    satisfies $\langle C_b, S_i \rangle = 1$.
    $C_i'$ is the cycle with minimum length
    chosen between $C_a$ and $C_b$.
    If $\mathrm{length}(C_i') < \mathrm{length}(C_i)$ then
        Add $C_i$ to $\mathrm{Candidates}\{i\}$.
        $C_i = C_i'$.
        Update $\mathrm{CycleSet}$.
        For $j = 1$ to $i - 1$
            If $\langle C_i, S_j \rangle = 1$ then
                $S_j = S_j + S_i$.
                Update $\mathrm{CycleSet}$.
            end
        end
        For $j = i + 1$ to $N$
            If $\langle C_i, S_j \rangle = 1$ then
                $S_j = S_j + S_i$.
                Add $\mathrm{Candidates}\{i\}$ to $\mathrm{Candidates}\{j\}$.
                Update $\mathrm{CycleSet}$.
            end
        end
    end
    Delete $\mathrm{Candidates}\{i\}$.
end
```

For large and sparse graphs, node fusion will only modify locally a few cycles of the graph, and most of the cycles of the previous minimum cycle basis will remain the same. The shortest path method used to compute the shortest cycle C_i' is only executed for node n_f, reducing the shortest cycle computation at step i with respect to de Pina's algorithm from $O(mn + n^2 logn)$ to $O(m + nlogn)$. The cost of updating S_j at step i is N^2, that is $O(m^2)$. The total cost of our algorithm is $O(m^3 + mnlogn)$. In practice, S_j is only updated when either it exists C_i' and its length is smaller than length of C_i or there are candidates for that column of S. The first condition, for large and sparse graphs, only happens for a small percentage of cycles C_i of the basis. If we also take into consideration that C_k and possibly also S_k are sparse vectors and that we can manage them locally for all operations described we may state, similar to [20] with respect to de Pina's algorithm, that the $O(m^3)$ part is not the bottleneck of our algorithm and that the practical cost is $O(nlogn)$. Additionally, as the shortest path computation using Dijkstra's algorithm is stopped when distance from node n_f is greater than length(C_i), or even before if a shortest cycle C_i' is found, only local nodes to n_f are checked and not all the nodes of the graph, reducing even more the computational time of our algorithm and making it local to node n_f.

IV. MULTI-ROBOT HIERARCHICAL SLAM USING A MINIMUM CYCLE BASIS OF CONSTRAINTS

We extend the results of the previous section to the multi-robot case. We have considered a centralized approach, although leaving the vehicles full autonomy when performing SLAM to obtain their own local maps. A central manager will be in charge of collecting all the local maps from the different vehicles when they make them available, and constructing the global map from them, fusing local maps, if the fusion is geometrically consistent with the global map, when it detects a high degree of overlapping between maps. This situation arrives, as explained before, when revisiting previous mapping areas or when, after an optimization process of the Hierarchical SLAM method, new matchings between maps are detected. In the multi-robot scenario, the fusion between local maps belonging to different graphs will also produce the fusion of the graphs, as it is explained in the following subsection.

Although each vehicle can use the global and local information provided by the manager to make decisions when performing its own SLAM, it doesn't incorporate uncertainty information from other maps into its new local maps, to assure at every time the maintenance of map independence.

A. *Graphs management*

To facilitate global map management a set of graphs will be used. At the beginning of the process there will be as many graphs as vehicles performing SLAM. Each graph will only have one root node and a spanning tree constructed from this node, to provide a unique calculation of the absolute position of the features from the origin of the initial map, using the relative transformations between maps. Fusion between maps of the same graph will be treated as in the single-robot case.

Local map fusions can also be done between local maps collected from different vehicles, leading to the fusion of their respective graphs. Before considering a fusion between graphs we need to define a minimum number of nodes to be previously paired. This is done to check for consistency of the graph fusion. Priority between graphs must be established at the beginning, to select, during the graph fusion process, the root of the new graph from the two roots available. The direction of the tree path from the other root to the fused node will be inverted to maintain the tree structure after the fusion. If during the fusion process an inconsistency in the upper level is detected, the fusion is not performed. In the Victoria Park experiment detailed below we have set to two the minimum number of paired nodes necessary to proceed to check for a graph fusion. The process of fusing the first pair of nodes produces the fusion of the two graphs. Once this operation is done the next pairs of nodes to be fused belong to the same graph.

B. Calculation of the minimum cycle basis after a fusion of two graphs.

In the process of fusion of two graphs, these two graphs become subgraphs that are connected, but not 2-connected, by means of the fused node. For this reason, the new minimum cycle basis consists of the concatenation of the two previous minimum cycle bases [2].

Before fusion, the minimum cycle basis of graph 1 is represented by chords Ch^1 and matrices C^1 and S^1 of dimension $N_1 \times N_1$, and the minimum cycle basis of graph 2 by chords Ch^2 and matrices C^2 and S^2 of dimension $N_2 \times N_2$. After fusion the minimum cycle basis of the new graph is represented by chords Ch and matrices C and S constructed as follows:

$$Ch = \{Ch^1, Ch^2\} \tag{18}$$

$$C = \begin{bmatrix} C^1 & 0_{N_1 \times N_2} \\ 0_{N_2 \times N_1} & C^2 \end{bmatrix} \tag{19}$$

$$S = \begin{bmatrix} S^1 & 0_{N_1 \times N_2} \\ 0_{N_2 \times N_1} & S^2 \end{bmatrix} \tag{20}$$

V. EXPERIMENTAL RESULTS

We have tested the multivehicle hierarchical SLAM methodology described in this paper using a real world data set: the Victoria Park data set, widely used as a benchmark for SLAM algorithms. Although the data set was collected by only one vehicle, it will be adapted to several vehicles by constructing consecutive local maps and using every local map around the origin to finish with the local maps assigned to one vehicle and to start with the maps assigned to the following one. We have used, as criterium to finish a local map, that the distance travelled by the vehicle is 40 m. We have obtained 99 local maps that we have assigned to 6 different vehicles: 21 consecutive maps for the first vehicle, 14 for the second, 13 for the third and forth, 17 for the fifth and 21 for the sixth vehicle. We assume that the vehicles don't know their initial position: it is initially assigned as $[0, 0, 0]'$ for all of the 6 vehicles. We have established a priority between vehicles to be used when fusing two graphs: vehicle 1 has the highest priority and vehicle 6 the lowest.

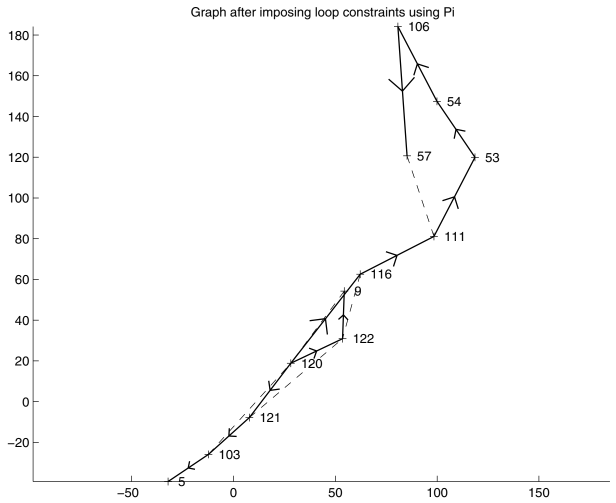

Fig. 4. Graph after the fusion of graphs 1 and 4.

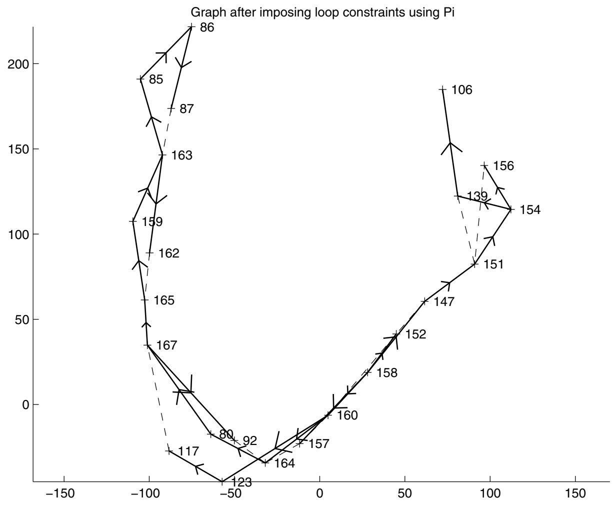

Fig. 5. Graph after the fusion of graphs 1+2+3+4+5 and 6.

We have employed a process of 21 steps. During each step the 6 vehicles, in a consecutive way, pass one of their maps to the central manager, until they finish with their last map. The central manager updates the different graphs and cycle bases using the methods detailed above.

Figures 4 and 5 show the new graphs produced by the Multivehicle Hierarchical SLAM at two different moments of the process: when the first fusion of graphs occurs, and when the last fusion of graphs occurs. Figure 6 show the global map generated after this last fusion. At the end of the process all graphs from the 6 different vehicles have been fused into a unique graph.

During the experiment we have compared the minimum cycle bases obtained with our incremental algorithm with the minimum cycle bases obtained using an implementation of the batch algorithm from de Pina [24]. All the minimum cycle bases obtained are the same for the two algorithms.

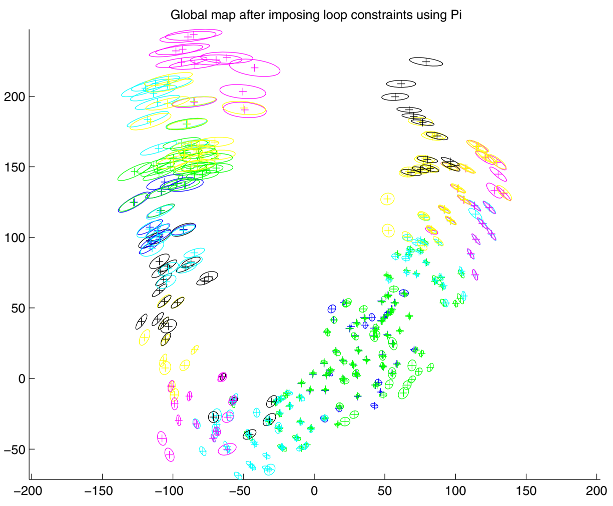

Fig. 6. Global map after the fusion of graphs 1+2+3+4+5 and 6.

VI. CONCLUSION

We have proposed to divide the upper (or global) level of the Hierarchical SLAM method using a minimum cycle basis of constraints. Doing so we can obtain accurate regional maps that are very useful when we are only interested in mapping a part and not the whole of the global map. For this purpose we have developed an incremental algorithm that, after two local maps are fused, updates the minimum cycle basis. This update is local to the fusion area. We have extended our algorithm to the multi-robot case. Finally we have tested our method using the Victoria Park data set with satisfactory results. In future work we will research for algorithms that accelerate the detection of graph fusion and try to obtain, using different simulations, the practical cost of the incremental minimum cycle basis algorithm for different map topologies.

REFERENCES

[1] Lars A. A. Andersson and Jonas Nygårds, "C-SAM: Multi-Robot SLAM using Square Root Information Smoothing", Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2798-2805, 2008.

[2] F. Berger, P. Gritzmann, S. de Vries, "Minimum cycle basis for network graphs", Algorithmica, 40,51-62, 2004.

[3] H. Jacky Chang, C. S. George Lee, Y. Charlie Hu and Yung-Hsiang Lu, "Multi-Robot SLAM with Topological/Metric Maps", Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1467-1472, 2007.

[4] C. Estrada, J. Neira, and J. D. Tardós, "Hierarchical SLAM: Real-Time Accurate Mapping of Large Environments", IEEE Transactions on Robotics, vol. 21, No. 4, pp. 588-596, 2005.

[5] J. W. Fenwick, P.M. Newman, and J. J. Leonard, "Cooperative concurrent mapping and localization", Proceedings of the IEEE International Conference of Robotics and Automation, 2002.

[6] J. Folkesson and H. I. Christensen, "Graphical SLAM - a self-correcting map", Proceedings of the IEEE International Conference on Robotics and Automation, pp. 791-798, 2004.

[7] J. Folkesson, and H. I. Christensen, "Graphical SLAM for Outdoor Applications", Journal of Field Robotics, vol. 24, no. 1/2, pp. 51-70, 2007.

[8] J. Folkesson, and H. I. Christensen, "Closing the Loop With Graphical SLAM", IEEE Transactions on Robotics, vol. 23, no. 4, pp. 731-741, 2007.

[9] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localisation and mapping", IEEE Transactions on Robotics, vol. 21, No. 2, pp. 1-12, 2005.

[10] U. Frese, "Treemap: An O(logn) algorithm for indoor simultaneous localization and mapping", Autonomous Robots, vol. 21, No. 2, pp. 103-122, 2006.

[11] A. Golynski, J. D. Horton, "A polynomial time algortihm to find the minimum cycle basis of a regular matroid", 8th. Scandinavian Workshop on Algorithm Theory, 2002.

[12] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent", Proceedings of Robotics: Science and Systems, Atlanta, GA, USA, 2007.

[13] Giorgio Grisetti, Dario Lodi Rizzini, Cyrill Stachniss, Edwin Olson, Wolfram Burgard, "Online Constraint Network Optimization for Efficient Maximum Likelihood Map Learning", Proceedings of the IEEE International Conference on Robotics and Automation, 2008, pp. 1880-1885.

[14] J.D. Horton, "A polynomial-time algorithm to find a shortest cycle basis of a graph", SIAM Journal of Computing 16, 359-366, 1987.

[15] S. Huang and G. Dissanayake, "Convergence and consistency analysis for Extended Kalman Filter based SLAM", IEEE Transactions on robotics, vol. 23, no. 5, pp.1036-1049, 2007.

[16] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Fast incremental smoothing and mapping with efficient data association", Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1670-1677, 2007.

[17] T. Kavitha, K. Mehlhorn, D. Michail, and K. E. Paluch, "A faster algorithm for minimum cycle basis of graphs", 31st International Colloquium on Automata, Languages and Programming, Finland, 846-857, 2004.

[18] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai, "A practical decision-theoretic approach to multi-robot mapping and exploration", Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003, pp. 3232-3238.

[19] K. Konolige, "Large-scale map-making", Proceedings of the AAAI, pp. 457-463, 2004.

[20] K. Mehlhorn and D. Michail, "Implementing Minimum Cycle Basis algorithms", ACM Journal of Experimental Algorithmics, 11(2):1-14, 2006.

[21] K. Mehlhorn and D. Michail, "Minimum Cycle Bases: Faster and Simpler". Accepted for publication in ACM Transactions on Algorithms.

[22] Kai Ni, Drew Steedly, and Frank Dellaert, "Tectonic SAM: Exact, Out-of-Core, Submap-Based SLAM", Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1678-1685, 2007.

[23] E. Olson, JJ. Leonard, And S. Teller, "Fast iterative alignment of pose graphs with poor inicial estimates", Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2262-2269, 2006.

[24] J.C. de Pina, "Applications of shortest path methods", PhD thesis, University of Amsterdam, Netherlands, 1995.

[25] S. Thrun, Wolfram Burgard, Dieter Fox, "A Real-Time Algorithm for Mobile Robot Mapping With Applictions to Multi-Robot and 3D Mapping", Proceedings of the IEEE International Conference on Robotics and Automation, pp. 321-328, 2000.

[26] S. Thrun, "A probabilistic online mapping algorithm for teams of mobile robots", International Journal of Robotics Research, vol. 20, no. 5, 2001.

[27] S. Thrun, M. Montemerlo, "The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures", The International Journal of Robotics Research, vol. 25, pp. 403-429, 2006.

[28] S. Thrun AND J.J. Leonard, "Simultaneous Localization and Mapping", Ch. 33, pp. 871-889, Springer Handbook of Robotics, ed. B. Siciliano AND O. Khatib, Springer, 2008.

[29] S. B. Williams, G. Dissanayake, and H. Durrant-Whyte, "Towards multi-vehicle simultaneous localisation and mapping", Proceedings of the IEEE International Conference of Robotics and Automation, 2002.