

EFFICIENT GLOBAL LOCALIZATION BY SEARCHING A BIT-ENCODED GRAPH

P. San Segundo, D. Rodriguez-Losada, R. Galan, F. Matia, A. Jimenez.

*Technical University of Madrid, Spain
C/ Jose Gutierrez Abascal 2, 28006 Madrid.
pablo.sansegundo@upm.es*

Abstract: Global localization is the problem of finding a totally unknown robot pose given a known map and a set of observations relative to the robot. This is the combinatorial correspondence problem between map landmarks and observations, and its complete solution is an NP-hard problem that makes it computationally intractable when the size of the map increases.

We propose the use of bit encoded graphs that take advantage of bit parallelism when solving instances of the deeply studied in graph theory Maximum Clique Problem (MCP). We have implemented and compared our solution with standard branch and bound heuristic search over the space of possible associations. Although the computational complexity of our algorithm is still exponential because it is complete, our results prove an impressive reduction of two orders of magnitude in required processing time thanks to the efficient underlying model, making it possible to deal with previously computationally intractable scenarios. Moreover, it is the first time (up to our knowledge) that this kind of technique is used in the robotics domain and it could possibly be applied to other problems such as path planning.

Keywords: Bitboards, global localization, maximum clique, search, algorithm

1. INTRODUCTION

Any mobile robot must know its position in the environment to successfully accomplish its task. If the map is previously unknown, the robot must incrementally build a representation (map) of the environment while it moves around it, which is known as the Simultaneous Localization and Mapping problem (SLAM). If the map is already known by the robot, then the problem is known as localization. Continuous localization or pose tracking is applied if a good initial estimation of the robot position, typically computed incrementally using the robot odometry, is known. In this case the observations are just used for correcting that position.

A completely different approach is necessary if no initial estimation of the robot is available which is known as the global localization problem. We assume that the map is made up of landmarks, individual features that can be detected and observed by the robot relative to its position. The global localization problem is essentially a correspondence problem between the landmarks and the observations read by the robot sensors at a specific position. The goal is to find the correct set or all sets (in case of symmetric environments) of associations between the observations and the landmarks. Once this set has been computed, obtaining the robot position is a simple task of triangulation or error minimization that can be done in constant time.

The cost of finding a correct set of jointly compatible pairings between observations and landmarks is

exponential in time with the number of landmarks, and many factors can potentially complicate finding that set, such as spurious observations (observations that do not correspond to any landmark of the map) and noise in the observations and map.

Several approaches deal with this computational complexity by assuming extra hypothesis or simplifications, or by searching for a suboptimal solution. (Betke and Gurvits 1997) find a solution in linear time in the number of landmarks by using just three observations to triangulate the robot position. Also (Neira, *et al.* 2003; Paz *et al.* 2005) proposes a linear time localization algorithm, but this time using an approximate representation of the environment and a voting algorithm. Their experiments evaluated just one hundred landmarks.

Despite the robustness of these solutions it is clear that they are not complete, nor guaranteed to compute all possible robot poses that jointly satisfy the maximum number of pairings. This problem, no matter the approach, if solved completely has an exponential cost.

Typical heuristics that search the associations tree to get the solution as quickly as possible are always implemented in programming languages that represent the binary information of a possible association (0 no match, 1 match) in an integer variable, which is used many times during search. When this information is handled inside the processor, a full word (typically 32-64 bits) is actually used as storage, as single bits cannot be managed independently. Repeat this kind of operation millions of time and the inefficiency becomes clear.

Following our own recent research on bit parallelism in search (San Segundo, et al., 2006), we have used bit strings (sometimes called bitboards when used in board game domains such as chess (Heinz 1997)), to solve the generic correspondence problem implicit in global localization. See (Neira et al., 2003; Bailey et al., 2000).

To exploit bit parallelism to the maximum, we have designed a new *bit-graph* model to reach an efficient implementation of MCP, nevertheless complete, and applied it to global localization. Our empirical results outperform other existing solutions such as a generic branch and bound heuristic over the associations space (Neira et al., 2003), reducing the required processing time up to two orders of magnitude.

Up to our knowledge, this implementation constitutes the first successful use of bitboards in the robotics domain, but more importantly, it opens the door to a promising technique that could be also used in other problems as data association in SLAM (Neira et al., 2001), computer vision correspondence matching or path planning.

2. THE GLOBAL LOCALIZATION PROBLEM

Global localization in two dimensions can be viewed as point set matching between a set of landmarks L and a set of observations O for instance read by a robot. Since observations are generated at a given moment in time through sensor reading (i.e. a laser beam), matching needs only to consider rigid motion (translation and rotation) and noise, but not scaling. Furthermore, $|L| \gg |O|$ in the general case.

In this paper we propose a new algorithm that uses bit parallelism for global localization. It is explained full detail in the following section. To test it, we have also implemented an alternate classical solution. In both cases we use distance between landmarks as a location independent feature to formulate binary geometric constraints that are used to explore the tree of relations.

The classical algorithm proceeds in a depth-first manner, expanding the best known current hypothesis to try to increase the number of matched pairings. The list of hypothesis is stored in order to improve performance. Hypothesis that could not satisfy the current number of matched pairings are pruned. This algorithm is conceptually the same as the Geometric Constraints Branch and Bound (GCBB) (Neira, et al., 2003) but with some differences: we do not search just for the best hypothesis but for all hypothesis of the same number of pairings that would simultaneously satisfy all the constraints. Furthermore we do not use unary constraints, as our landmarks are just two dimensional points.

3. MAXIMUM CLIQUE SEARCH USING BITBOARDS

For a given map of landmarks and a set of observations, a graph model is built which stores all matching distances between pairs of observations and landmarks. We then proceed to extend arc-

consistency to n couples by searching through the graph space using a branch and bound algorithm which is known to be complete (Carraghan et al., 1990). In this way the initial correspondence problem is transformed into maximizing ‘ n ’ through search, a deeply studied problem in the graph domain known as the Maximum Clique Problem (MCP).

MCP is known to be NP-Hard (Garey, et al., 1979) and no ‘good’ complete algorithm is expected to be found. However our results show amazing speed for a reasonable large number of landmarks and observations. The ultra-fast search is explained by a specific encoding of the graph based on bit strings, a novel technique which has been one of our lines of research in recent years (San Segundo, et al., 2005; San Segundo et al., 2006). In this paper we have used this idea and adapted it to global localization problem. We call our new bit encoded algorithm BE-MCP.

3.1. The Maximum Clique Problem

Given an undirected graph G made up of vertices V and edges E , $G = \{V, E\}$, two vertices are said to be adjacent if they are connected by an edge. A graph is called *complete* if any two of its vertices are adjacent. Informally, a complete graph is also called a *clique*. A k -complete graph, or k -clique, is a complete graph where $|V| = k$.

Typical problems from the clique domain are the *k-clique* problem which tries to find a clique of k vertices and the *maximum clique problem* which looks for the maximum possible size of a clique in the given graph. Other important related graph problems are the *independent set problem* (where the aim is to find induced graphs with every pair of vertices not adjacent) or the *vertex cover problem*. All of them are well known to be NP-hard (Garey, et al., 1979).

Clique study has several important practical application fields such as computer vision or signal processing. It can also be used for backward reasoning in rule based models. MCP in particular, is a very important problem in combinatorial optimization, with many applications which include: market analysis (Roy 1970), project selection (Christofides 1975), and signal transmission (Berge 1991). The interest for this problem led to the algorithm thread challenge on experimental analysis and algorithm performance promoted by Second DIMACS Implementation Challenge (Johnson and Trick 1996).

In this paper we have focused on global localization in two dimensions, and reduced the problem to a Maximum Clique problem and used a complete branch and bound search algorithm to scan the search space. Following our previous work on bitboard search models we have adapted our general purpose bit encoded algorithm in (San Segundo, et al., 2006) to the specific requirements of the graph generated for the global localization model.

3.2. Bitboard search models

A bit string is an array data structure which stores individual bits in a compact form and is effective at

exploiting bit-level parallelism in hardware to perform operations quickly. An example of their application can be found in priority queues of operative systems (i.e the Linux kernel), where the bit at index k is set iff the k -th process is in the queue. Many known examples can also be found in board games, such as chess (Heinz, 2000), which is where the term *bitboard* was first employed. In these domains, bits have an interpretation in most cases related to a particular condition concerning each square of the board. Consider the bitboard whose interpretation is the ‘placement of all white pawns on a chessboard’; a board position can thus be encoded using only 12 of these bitboards (6 for each piece, each piece having two possible colours).

Our own research on this field (San Segundo, et al., 2006) has allowed us to extend this idea successfully for search problems, so that complexity can be improved by exploiting this form of parallelism. This paper constitutes a further example against the extended *a priori* opinion throughout the scientific community, that the benefits of parallelism at bit level have at least an equal counterpart in the overhead needed to extract information relative to a single bit from the compact bit array

The chess example from above shows an obvious advantage when using bitboard modelling: space complexity is reduced by a factor equivalent to the size in bits of the processor word (W_{size}). This fact alone cannot, however, justify the use of modelling at bit level in search domains. The key factor is the reduction in time complexity resulting from the use of bitwise operations either as frame knowledge or as additional procedural knowledge during search.

In this paper we show the feasibility of this approach by adapting our general purpose MCP solver to global localization, obtaining an impressive performance compared to state of the art solutions.

4. THE GRAPH MODEL

To solve the global localization problem we have reduced the problem to an MCP over a bit graph model G . This is a key point in the whole process as information relative to rigid motion and noise must be included in G , and a bit interpretation must be correctly chosen to gain overall efficiency from bit-parallelism. We have then adapted a classical branch and bound algorithm as in (Carraghan et al., 1990) to find the closest match. For simplicity we consider a two dimension domain, but the technique used could be extended to three or more dimensions trivially. In this section we analyse in detail the complete procedure.

4.1. Building the graph

Let L and O be two sets of points. Correspondence has been studied extensively in graph theory setting, where the problem is to find a matching in graph (V,E) with vertices $V = L \cup O$ using edges with weights interpreted as distances between points. See (Vaidya 1989). This approach is quite good for rigid

motion correspondence invariant, as is the case of the global localization problem

However we propose a new approach to build the graph, an approach which is *bit-encoding* oriented. To avoid confusion with the classical weighted approach, we name this graph a *bit-graph* model, or simply a *bit model*. Our bit model is an undirected non weighted graph G , with vertices the Cartesian product of both sets $(L \times O)$. Thus, each vertex can be interpreted as a landmark-observation pair.

More formally, let $G = \{V, E\}$ be our bit-graph model; let $|O|=o$, $|L|=l$ be the cardinalities of our observation set captured by the robot and the previously known landmark set respectively. Then for $L = \{l_1, l_2, \dots, l_l\}$ and $O = \{o_1, o_2, \dots, o_o\}$, $V = \{v_1, v_2, \dots, v_{l \times o}\}$ where $v_i = \{l_k, o_m\}$, $k = \text{quotient}(i/o)$, $m = 1 \text{ mod}(o)$ and $1 \leq k \leq l$, $1 \leq m \leq o$.

As a discrete metric between vertices we use an extension of Euclidean distance to pairs of points (D) such that D equals 1 when distances between pairs of observations are the same as distances between pairs of landmarks. It returns 0 otherwise.

More formally, let $d(x_1, x_2)$ be the Euclidean distance between two points x_1, x_2 . We define distance between sets of pairs of points $V = \{l_1, o_2\}$ as a function $D : \{V, V\} \rightarrow \{0,1\}$ such that:

$$D(V_1, V_2) = \begin{cases} 1 & \text{if } d(l_1, l_2) = d(o_1, o_2) \\ 0 & \text{otherwise} \end{cases}$$

To build the adjacency matrix M of our graph we map D to every element in M :

$$m_{ij} = D(V_i, V_j)$$

The i -th file of M encodes the potential matching of the pair $\{L_{\text{quotient}(i/o)}, O_{i \text{ mod}(o)}\}$ with the rest of possible $\{L, O\}$ pairs. The same reasoning can be made for the k -th column. D is a symmetrical relation; therefore M is symmetrical and represents an undirected graph. For implementation reasons, we have chosen $m_{ii} = 0$.

```

Initialization: U=Vertex set of G, size=0
function clique(U,size)
Step 1:Leaf node (|U|=0)
    1. if (max_size>size) record max_size
    2. return
Step 2:Fail upper bound (size +|U|<max_size)
    1. return
Step 3:Expand node from U
    1. vi:=N_sel(U)
    2. U:= U \ {vi}
    3. clique(U ∩ Nadj(vi),size+1)

```

Fig. 1: basic form of maximum clique problem (MPC) algorithm

The bit encoding. The bit encoding is a 1 to 1 mapping of bits to elements of the M matrix. This same bit encoding technique was used in (San Segundo, et al., 2006), and is optimal for the overall efficiency of our solving algorithm. It has an upper threshold of reduction in TIME the size of the CPU

word (W_{size}), compared to the same non bitboard model. See (San Segundo, et al., 2006).

The algorithm used for our experiments does not try to further exploit bit-parallelism by matching bit encoded knowledge obtained dynamically during search. This is an interesting line of research we intend to follow in the near future.

Advantages of the bit-graph model: In the first place, our model design is bit oriented, and uses bit parallelism to speed up search. As explained before, this gives us an upper threshold of reduction in TIME the size of the CPU word.

Bit parallelism is best exploited in non weighted graphs, which explains our choice of vertices, $|V|=l \cdot o$. Unfortunately, this increases the need for initial space storage for the graph, which is somewhat palliated by the w_{size} compression factor of any bit modelling technique.

Another important point is that our approach doesn't consider dissimilarity metrics between the whole set of map landmarks and the set of observations (i.e bottleneck distance, Hausdorff distance etc., see (Veltkamp et al., 1999) for further examples), but rather classical distances related to correspondence pairs. Once the graph has been built by precomputing all possible distances between all $L \times O$ pairs, the extension of the distance relation to n -tuples of pairs is solved using an MCP algorithm and distances are never evaluated again. To precompute all distances during graph generation we have used a simple algorithm which is in $O(l^2o^2)$. Even so, results have been quite impressive.

For the same reason, the bit graph model turns out to be very robust to bad observations, as they are filtered during generation time. For example, a bad observation o_{15} for a total number of landmarks 100 and a total number of observations 20 will result in all members of files and columns 15, 35, 55, 75, ..., 1975, 1995 of adjacency matrix M (with size 2000 by 2000) equalling zero, as no possible matchings between the pairs $\{l_j, o_{15}\}$ ($1 < j < 100$), and the rest of possible $L \times O$ pairs can be found.

Furthermore, the approach can easily be extended to higher dimensions, and different kinds of noise can be added *without changing the search algorithm*. It is only necessary to modify the metrics employed during graph generation.

Disadvantages of the bit-graph model: A disadvantage of our approach is related to SPACE complexity as the adjacency matrix M grows in size $O(l \cdot o)^2$ in the number of bits. This means that initial space storage for an instance with $l=1000$ and $o=20$ is $(20 \cdot 10^3)^2 / 8 \cdot 1024 \cong 49 MB$ which can be handled easily by everyday commercial PC's. However, the number grows quite quickly. For $l=10000$, $o=50$ RAM space storage is now 3GB approx.

This setback, nevertheless, is not the end of the story. In the first place, the latter instance already is of practical use in many global localization problems.

Secondly, because of the intrinsic nature of the problem and the way we build the adjacency matrix M , it must necessarily be very sparse. Since in our solver space is only needed initially, no further storage is needed during search, it seems reasonable that sparse techniques can be found to reduce overall space impact.

4.2. The search algorithm

We have adapted a classical branch and bound algorithm from the literature of maximum clique problem solvers for bit parallelism, following (Carraghan et al., 1990). The main paradigms for optimal clique solving algorithms are *backtracking* and *branch and bound*. Recent algorithms for MCP, which use some form of branch and bound are (Pardalos et al., 1994; Pardalos, P.M., Rodgers, G.P 1992; Carraghan et al., 1990) amongst others.

The basic form of MCP *branch and bound* solvers is that of (Carraghan et al., 1990). Figure 1 illustrates this algorithm in its most general form: $N_{adj}(v_j)$ returns the set of adjacent vertices to v_j ; max_size is a global variable which stores the best clique found so far and $N_{sel}(U)$ is any node selection strategy to chose a vertex from the set of vertices U .

Complexity. MCP is known to be NP-Hard (Garey, et al., 1979) Indeed a brute force approach for the k -

Clique problem is $O\left(\binom{n}{k} k^2\right)$ for a graph, $|V| = n$.

This can be improved by *branch and bound* approaches but non polynomial algorithms on the size of the graph are not expected to be found. However, we have been able to solve the correspondence problem for some instances with 8 noisy observations against 5000 random landmarks in around a second using a P4 with 2GB of RAM. Considering that, in this case, our bit-graph model is $(5000 \times 8)^2$ in bit-size, how can this be explained?

The answer lies in the bit encoding and in the specific nature of the global localization problem, which, as has been mentioned earlier, we model with a new technique to build very sparse bit-graph models (low density and low average degree). Branch and bound techniques work fine here because there is a lot of inexpensive cutting going on during search.

Thus, although obviously complexity increases non polynomially on the size of l and o , it rises slowly and makes our optimal solver an important new tool for real-life robot global localization applications.

Implementation details. The basic algorithm in figure 1 has been specially adapted for bit-parallelism as in (San Segundo P et al., 2006). At the present moment, no specific strategy for node expansion is being used, so there is ample ground for further improvements here. It is important to take into account that current implementation of BE-MCP finds all possible optimal solutions, so it can be applied as is to symmetrical environments. In spite of this, results still remain quite impressive.

5. EXPERIMENTS AND RESULTS

To test our approach, we have performed several experiments in a simulated environment with the two algorithms, GC-BB and BE-MCP. The map of the environment was defined as a set of ‘ n ’ indistinguishable two dimensional point landmarks randomly generated in a 100m size square. In each run, the robot is positioned at a random location and orientation, obtaining a maximum of ‘ o ’ observations, not necessarily observing all features within sensor range (figure 2).

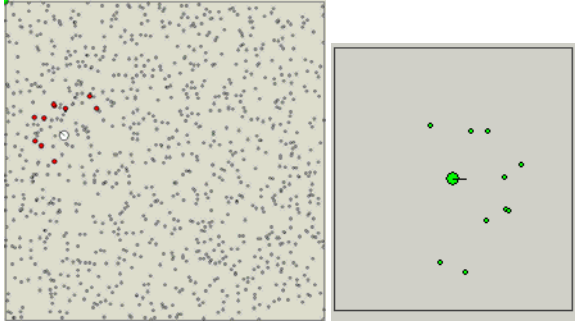


Fig. 2: Experimental simulated environment with 1000 landmarks randomly located (left) and 10 observations performed at a random robot pose

Both algorithms have been fully implemented in C++ as efficiently as possible, and tested in a P4 2,6GHz, 512Mb laptop. The distances between pairs of landmarks are precomputed for efficiency in both cases.

In the first experiment we compare the processing time of both algorithms for different number of map landmarks and observations. Each setting is repeated 10 times (ten different robot random poses), and the average, maximum and minimum processing times are computed, and represented in the tables. For this experiment we have fixed a tolerance for distance of 1% and 1 spurious observation has been forced in each set of observations (tables 1 and 2).

Table 1 Processing times (mean, max, min) of the GCBB algorithm. Landmark-Observations

Time(s)	Observations							
	5		8		11		14	
Landmarks								
250	0,25	0,38 0,13	0,25	0,56 0,14	0,31	1,09 0,14	0,25	0,49 0,16
500	2,63	5,39 1,11	4,61	11,89 1,28	2,42	4,94 1,22	2,00	3,33 1,25
750	10,36	21,22 5,44	19,70	41,75 7,36	8,15	17,63 3,70	12,69	26,28 5,03
1000	32,11	52,05 17,38	60,81	190,52 17,89	104,37	636,42 19,16	269,32	1320,80 30,75

Table 2 Processing times (mean, max, min) of the BE-MCP algorithm. Landmark-Observations

Time(s)	Observations							
	5		8		11		14	
Landmarks								
250	0,00	0,02 0,00	0,01	0,02 0,00	0,03	0,03 0,02	0,03	0,05 0,00
500	0,02	0,03 0,02	0,05	0,06 0,05	0,10	0,11 0,09	0,17	0,19 0,16
750	0,04	0,06 0,03	0,13	0,16 0,11	0,28	0,31 0,22	0,48	0,52 0,44
1000	0,09	0,09 0,06	0,42	0,52 0,27	0,99	1,06 0,89	1,79	1,94 1,58

It can be seen how the BE-MCP clearly outperforms the GCBB algorithm. This fact is even more relevant

as the number of landmarks increases, being the processing time at least two orders of magnitude lower for one thousand landmarks environments. It is important to note that the GCBB approach suffers a much higher variability. E.g. for 1000 landmarks and 14 observations, the processing time can be up to 1320s despite the average is 269s. The variance for the BE-MCP is much lower, practically guaranteeing the required time for every run.

Figure 3 shows the mean, maximum and minimum processing time for different settings. It can be seen that the variance of the BE-MCP practically remains constant while the variance of the GCBB increases with the number of landmarks.

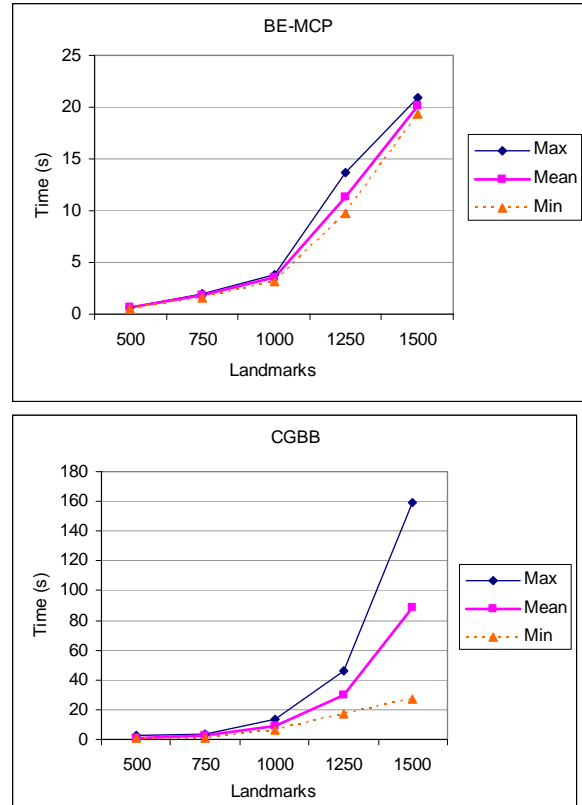


Fig. 3: Processing times for different numbers of landmarks and 25 observations, without spurious observations. Variance for BE-MCP remains close to constant

The second experiment fixes the number of landmarks to 500, the number of observations to 15, and evaluates the processing time while varying the number of spurious observations and the distance tolerance. To force a spurious observation, its actual values are randomly changed within the sensor range.

Table 3 Processing times (mean, max, min) of the GCBB algorithm. Tolerance-Spurious

Time(s)	Spurious observations											
	0		1		2		3		4		5	
Tol												
1%	0,54	1,20 0,30	2,33	3,17 1,45	10,73	45,89 2,65	21,54	48,91 6,92	42,43	76,13 16,59	76,09	134,30 31,48
2%	1,41	2,22 0,78	8,04	42,88 2,63	15,84	64,33 6,81	44,51	189,09 15,25	69,52	117,84 29,42	63,04	132,47 36,28
3%	1,17	3,16 0,58	13,13	89,59 3,36	8,64	12,31 4,58	22,70	57,11 11,80	39,04	62,03 28,05	65,54	150,20 48,42
4%	1,70	3,16 0,89	13,56	91,13 2,91	26,02	77,89 5,61	35,50	103,44 15,94	42,29	82,45 29,28	70,26	114,05 39,50
5%	1,50	2,88 0,61	9,88	55,09 2,67	15,38	45,02 8,95	40,60	153,70 14,36	56,67	160,27 23,97	71,42	144,89 43,19

Table 4 Processing times (mean, max, min) of the BE-MCP algorithm. Tolerance-Spurious

Time(s)	Spurious observations											
	0		1		2		3		4		5	
Tol												
1%	0,20	$\frac{0,22}{0,17}$	0,20	$\frac{0,22}{0,16}$	0,19	$\frac{0,22}{0,16}$	0,21	$\frac{0,22}{0,19}$	0,32	$\frac{0,42}{0,17}$	0,41	$\frac{0,42}{0,39}$
2%	0,40	$\frac{0,44}{0,36}$	0,39	$\frac{0,41}{0,34}$	0,40	$\frac{0,42}{0,38}$	0,40	$\frac{0,44}{0,36}$	0,39	$\frac{0,45}{0,33}$	0,39	$\frac{0,41}{0,34}$
3%	0,41	$\frac{0,45}{0,38}$	0,39	$\frac{0,42}{0,36}$	0,38	$\frac{0,41}{0,34}$	0,38	$\frac{0,41}{0,36}$	0,39	$\frac{0,44}{0,34}$	0,40	$\frac{0,42}{0,34}$
4%	0,40	$\frac{0,42}{0,36}$	0,39	$\frac{0,44}{0,36}$	0,39	$\frac{0,42}{0,36}$	0,40	$\frac{0,42}{0,38}$	0,39	$\frac{0,41}{0,38}$	0,38	$\frac{0,44}{0,34}$
5%	0,39	$\frac{0,44}{0,33}$	0,38	$\frac{0,42}{0,34}$	0,38	$\frac{0,42}{0,34}$	0,39	$\frac{0,44}{0,38}$	0,38	$\frac{0,42}{0,34}$	0,38	$\frac{0,42}{0,34}$

In this experiment the computational savings are again clearly visible, with processing times two orders of magnitude lower. But more importantly, the BE-MCP is not affected by the number of spurious observations, while the GCB is highly influenced as expected (tables 3 and 4).

In our third experiment, we used 2000 landmarks and 25 observations, with a fixed number of spurious observations of 10 (almost 50%) of the observations, with 1% distance tolerance. The average time for this setting of the BE-MCP algorithm is 25.1 s, with maximum time of 27.5s and a minimum of 21.985. With the GCB approach, this setting is basically intractable, and could require many hours of processing. Because of this no tabulated results are shown.

Both implementations have been tested in highly symmetric environments, with up to 20 correct solutions for a given set of observations, to check for completeness and in all cases both found all correct solutions. However, our novel BE-MCP algorithm proved to be much quicker, showing an impressive reduction in computation time.

6. CONCLUSIONS

We described a complete and efficient algorithm for the global localization problem given a noisy set of landmark readings from a map of the environment. The algorithm, being complete, runs in time non polynomial in the number of landmarks but employs, for the first time to our knowledge in this domain, a graph modelling technique to exploit bit parallelism to the maximum.

Our results show an impressive reduction of two orders of magnitude in required processing time compared to a classical search algorithm. This leads us to believe that the key ideas used could be extended to other robotic domains such as, for instance, path finding.

The current limit of our implementation is space requirement, as the adjacency graph matrix is stored in complete form. Further work will focus on implementing the sparsification of this matrix as well as testing our algorithm against real life applications. Another interesting line of research would be the correspondence problem of different features rather than distances to determine relative bearings of observations. Finally, extension to three dimensions should be a critical test.

7. REFERENCES

- Bailey T., Nebot E. M., Rosenblatt J. K., and Durrant-Whyte H. F. (2000). Data association for mobile robot navigation: A graph theoretic approach. *IEEE Int. Conf. Robotics and Automation*, pp 2512–2517, San Francisco, CA.
- Betke M. and Gurvits L. (1997). Mobile Robot Localization Using Landmarks. *IEEE transactions on robotics and automation*, vol. 13, no. 2.
- Carraghan R., Pardalos P.M. (1990). *An exact algorithm for the maximum clique problem*, Oper. Res. Lett. 9: pages 375-382.
- Garey M.R., Johnson D.S. (1979) *Computers and Intractability: A guide to the theory of NP-completeness*. Freeman, New York.
- Heinz E.A., (2000) *Scalable Search in Computer Chess*. Vieweg.
- Heinz E.A., (1997) How DarkThought plays chess, *ICCA Journal*, 20(3): pages 166-176.
- Neira J., Tardós J.D (2001), Data association in stochastic mapping using the joint compatibility test. *IEEE Trans. on Robotics and Automation*, vol. 17, no. 6, pp. 890-897.
- Neira J., Tardós J.D, Castellanos J.A. (2003) Linear time vehicle relocation in SLAM. 2003 IEEE Int. Conf. Robotics and Automation, Taipei, Taiwan, May, 2003.
- Pardalos, P.M., Rodgers, G.P. (1992) A branch and bound algorithm for the maximum clique problem, *Comput. Oper. Res.* 19(5): pages 363-375.
- Pardalos, P.M., Xue, J. (1994) The maximum clique problem. *Global Optimization*. 4: pages 301-328.
- Paz L.M., Pinies P., Neira J., Tardós J.D. (2005) Global Localization in SLAM in Bilinear Time. 2005 IEE/RSJ Int.Conf.on Intelligent Robots and Systems, August 2-6, Edmonton, Canada
- Veltkamp R.C., Hagedoorn M. (1999) State-of-the-art in shape matching. Technical Report UU-CS-1999-27, Utrecht University, the Netherlands.
- San Segundo P., Galán, R, Rodríguez-Losada, D et al. (2006). Efficient search using bitboard models. *Proceedings XVIII Int. Conf. Conference on Tools for AI (ICTAI 06)* : 132-138.
- San Segundo P. and Galán R. (2005) *Bitboards: A new approach*. Artificial Intelligence and applications (AIA) Innsbruck, Austria.
- Vaidya, P.M. (1989). Geometry helps in matching. *SIAM Journal of Computing*, 18(6): 1201-1224.