Dense Mapping for Telemetric Sensors:

efficient algorithms and sparse representation

Blind Submission. Paper-ID [140]

***Abstract*— This paper focuses on efficient occupancy grid building based on a sparse new grid representation: wavelet occupancy grids and a new update algorithm for telemetric sensors. The update algorithm takes advantage of the natural multiscale properties of the wavelet expansion to update only parts of the environement that are modified by the sensor measurements and at the proper scale. The sparse wavelet representation coupled with an efficient algorithm presented in this paper provides efficient and fast updating of occupancy grids. It leads to real-time results especially in 2D grids and for the first time in 3D grids. Experiments and results are discussed for both real and simulated data.**

I. Introduction and Previous Work

The Simultaneous Localization And Mapping (SLAM) issue has found very convincing solutions in the past few years, especially in 2 dimensions. Thanks to a number of contributions [1] [2] [3] [4] [5] [6], it is now feasible to navigate and build a map while maintaining an estimate of the robot's position in an unknown 2D indoor environment on a planar floor. In these 2D conditions, the problem is theoretically and practically solved even in populated environment [7]. Some of the most impressive approaches are based on grid-based fast-slam algorithms [8] [3] [4], which offers a unified framework for landmark registration and pose calculation thanks to occupancy grids (OG) [9]. There are several advantages in doing so. They provide robots with the ability to build an accurate dense map of the static environment, which keeps track of all possible landmarks and represents open spaces at the same time. Only a simple update mechanism, which filters moving obstacles naturally and performs sensor fusion, is required. In contrast to other methods, there is no need to perform landmark extraction as the raw data from range measurements are sufficient. One of the benefits is accurate self positioning, which is particularly visible in the accuracy of the angle estimate. However, the major drawback is the amount of data required to store and process the grid, as a grid that represents the environment has an exponential memory cost in the number of dimensions. In 2D SLAM, this drawback is overcome by the sheer power of the computer and its huge memory. But this issue cannot be avoided for 3D SLAM even with today's desktop computing capabilities. Recently methods to deal with the 3D instance of the SLAM problem, in undulating terrains [6] have used landmark extraction, clustering and a special algorithm for spurious data detection. However, this map framework does not handle out-of-date data and hence the extra cost of removing or updating data coming from past poses of moving objects is not considered.

In this paper, we choose to use OGs and we present a new algorithm called wavelet hierarchical rasterization that hierarchically updates the wavelet occupancy grid in the relevant area of the environment. It does not require, as with other approaches [10], any intermediate representation for adding observations in the wavelet grid. This leads to real-time dense mapping in 2D and we propose a special instance of this algorithm that performs well enough for real-time 3D grid modelling. This method is intrinsically multi-scale and thus one of its major advantages is that the mapping could be performed at any resolution or with any precision in an anytime way.

There exists a large panel of dense mapping techniques: amongst the other popular representations of dense 3D data are raw data points [5], triangle mesh [11] [12] or elevation maps [13], [14]. However there are major drawbacks in using such representations. With clouds of points it is not easy to generalize: roughly speaking, there is no simple mechanism to fill in the holes. Moreover, the clouds of points are generated by the successive records of telemetric measurements, thus the amount of data is prohibitive after a few hours of recording. The triangle mesh representation is a kind of $2\frac{1}{2}$-D map and the space representation is also incomplete. In simple elevation maps [11], for the same reasons, holes in the environment such as tunnels are not part of the set of representable objects. This problem is overcome in [14] since there is a little number of vertical steps for each part of the map. The worst point is that most of these methods lack a straightforward data fusion mechanism. In particular, it is rarely simple to include information on the absence of features. Triangle mesh [12] and elevation maps [14] suffer most from this problem. Therefore most of the time these representations are obtained as a batch processing or for a little environment.

For telemetric sensors OGs represent the probability for the presence of a reflective surface at any world location. Therefore the ability to update the map for both the presence and the absence of data is a major advantage, which we call the evolution property. With OGs this property comes not from a batch process but is part of the probabilistic map model definition. The cost is that a huge amount of memory is needed to cope with the map discretization. In [10], a wavelet grid based approach was introduced which enables the representation of grids in a compact but flexible format. Wavelet occupancy grids, unlike pyramid map representations where redundant information is stored at each scale, store at finer scale only the differences with the previous coarser scale. This representation allows the elimination of redundant information where there is no additional detail such as for empty spaces. Furthermore it provides a natural multi-scale representation with different levels of detail at different scales of resolution. In order to build the map, a standard approach will use an intermediate standard grid representation on which a wavelet transform will be performed. Even if a 2D wavelet transform can be performed in real-time, the extension to the case of a 3D transform in real-time is not apparent. So for a reasonable field of view, it makes the previous method unfeasible for 3D data. Our algorithm overcomes this difficulty with a hierarchical strategy that updates only the relevant area of the environment and at the proper scale. In a first section, we will present the wavelet framework and the data structure. In a second section the sensor model within the occupancy grid framework for the wavelet space is described. Next, we present the wavelet hierarchical rasterization algorithm. Lastly, we present our results in 2D on real data and in simulated 3D data where correct localisation is provided. Although in all the paper the algorithm is described for any kind of telemetric sensor, the implementation and the experimental section are with laser data only.

II. Wavelets

In this paper, the occupancy state is represented as a spatial function. Our main contribution is an occupancy updating technique that can be performed in a compact manner. At the heart of the method is wavelet representation which is a popular tool in image compression. Indeed, there exists a similarity between OGs and images [9]. The wavelet transform known as the Mallat algorithm successively averages each scale, starting from the finest scale (1 right to left). This mean produces an oracle to predict the information at finer cells, then only differences from the oracle are encoded. This averaging produces the next coarser scale and differences with neighboring samples at the fine scale gives the associated so called detail coefficients. There is no loss of information in that process since the information contained in the finer scale can be recovered from its average and detail coefficients. Since two neighboring samples are often similar, a large number of the detail coefficients turn out to be very small in magnitude, truncating or removing these small coefficients from the representation introduces only small errors in the reconstructed signal, giving a form of "lossy"signal compression. Lossless compression is obtained by removing only zero coefficients.
In this paper wavelets are just used as a special kind of vector space basis that allows good compression. Details about wavelet theory is beyond the scope of this paper and references can be found in [15] [16] [17].

A. Notations

Wavelets are built from two set of functions: scaling and detail functions (also known as wavelet functions). Scaling functions, $\Phi(x)$, capture the average or lower frequency information and a scaling coefficient is noted s_t^l. Detail functions, $\Psi(x)$, capture the higher frequency information and a detail coefficient for a detail function f is noted $d_{t,f}^l$. The set of wavelet basis functions can be constructed by the translation and dilation of the scaling and detail functions. Thus each of the basis function or coefficient is indexed by a scale l and a translation index t. Moreover a detail function is indexed by its type f. In this paper, the non-standard Haar wavelet basis is used. For non-standard Haar wavelet basis, there is only one mother scaling function and $2^d - 1$ mother wavelet functions, where d is the dimension of the signal. Expanding a function O in the Haar wavelet basis is described as:

$$O(x) = s_0^{-N}\Phi_0^{-N} + \sum_{l=-N}^{l=0}\sum_t\sum_f d_{t,f}^l \Psi_{t,f}^l, \tag{1}$$

where f is an index from 1 to $2^d - 1$, and N the level such that the whole grid appears as one cell. As can be seen in eq. 1, only one scaling coefficient and one scaling function are required in the expansion of any function $O(x)$. As shown in fig. 1, the scaling coefficients at other levels are computed as part of the decompression (left to right) or compression (right to left) process.
The scaling coefficient for a certain level l and translation t holds the average of values contained in the support of the scaling function. The support of any Haar basis function in dimension d is a d-cube *e.g.* a square in 2D and a cube in 3D. If the finest level is 0 and coarser levels are indexed by decreasing negative integers, the side of such a d-cube is 2^{-l} where the unit is in number of samples at level 0.

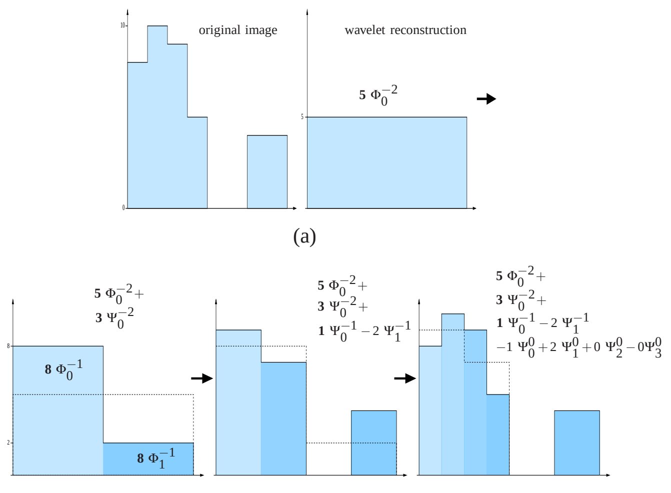


Fig. 1. The 1D image (upper, left) is: $[8,10,9,5,0,0,4,4]$, and its unnormalized (used here because it is simpler to display) Haar representation is: $[5,3,1,-2,0,0]$. The image is then reconstructed one level at a time as follows: $[5] \rightarrow [5+3, 5-3] = [8,2] \rightarrow [8+1, 8-1, 2-2, 2+2] = [9,7,0,4]$ and so on. Here 0 is the finest scale index or the scale where data is gathered and -2 is the coarsest scale.

B. Tree structure

The key step in a wavelet decomposition is the passage from one scale to another. The support of a Haar wavelet function at level l is exactly partitioned by the support of the 2^d wavelet functions at level $l+1$, (see Fig. 1 for dimension 1). This leads to a quadtree for the case of 2D space and octree for 3D space that hierarchically maps the whole space. A node of the tree stores 2^d-1 detail coefficients and potentially 2^d children that encode finer details if they are necessary to reconstruct the expanded function. The key step of a node creation is described figure 2.

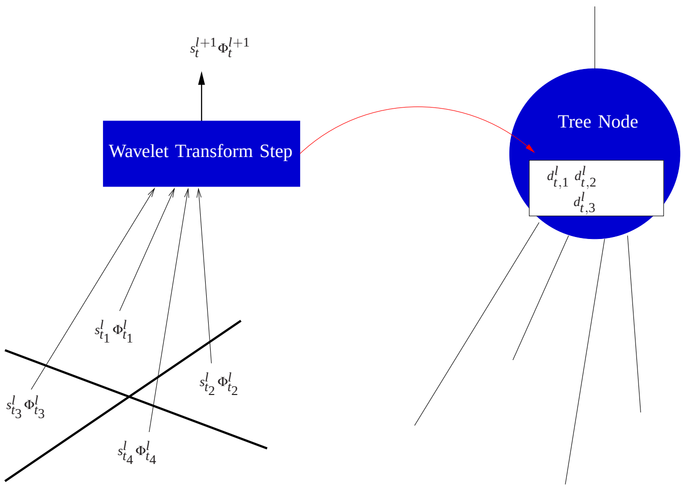


Fig. 2. A key step of a Haar wavelet transform in 2D. 4 scaling samples at scale l generates 1 coarser scaling coefficient at scale $l+1$ and 3 details coefficients at scale l that are stored in a wavelet tree node. In general the tree node has 4 children that described finer resolutions for each space subdivision. But if each child is a leaf and has only zero details coefficients then all the child branches can be pruned without information loss. And the tree node becomes a leaf.

This data structure is exactly a 2^d-tree, but it not only stores spatially organized data, but also summarizes the data at different resolutions. The root of the tree stores the scaling coefficient at the coarsest level and the support of the corresponding scaling function includes all the spatial locations of the signal data.

III. Occupancy Grids and Telemetric Sensor Models

OG is a very general framework for environment modelling associated with telemetric sensors such as laser range-finders, sonar, radar or stereoscopic video camera. Each measurement of the range sensor consist of the range to the nearest obstacle for a certain heading direction. Thus a range measurement divides the space into three area: an *empty* space before the obstacle, an *occupied* space at the obstacle location and the *unknown* space everywhere else. In this context, an OG is a stochastic tessellated representation of spatial information that maintains probabilistic estimates of the occupancy state of each cell in a lattice [9]. In this framework, every cell are independently updated for each sensor measurement, and the only difference between cells is their positions in the grid. The distance which we are interested in, so as to define cell occupancy, is the relative position of the cell with respect to the sensor location. In the next subsection, the Bayesian equations for cell occupancy update are specified with cell positions relative to the sensor.

A. Bayesian cell occupancy update.

a) Probabilistic variable definitions:

- Z a random variable[1] for the sensor range measurements in the set $\mathscr{Z}$.
- $O_{x,y} \in \mathscr{O} \equiv \{\text{occ}, \text{emp}\}$. $O_{x,y}$ is the state of the cell (x,y), where $(x,y) \in \mathbb{Z}^2$. $\mathbb{Z}^2$ is the set of indexes of all the cells in the monitored area.

b) Joint probabilistic distribution: the lattice of cells is a type of Markov field and in this article sensor model assumes cell independence. This leads to the following expression of a joint distribution for each cell.

$$P(O_{x,y}, Z) = P(O_{x,y})P(Z|O_{x,y}) \tag{2}$$

Given a sensor measurement z we apply the Bayes rule to derive the probability for cell (x,y) to be occupied 3:

$$p(o_{x,y}|z) = \frac{p(o_{x,y})p(z|o_{x,y})}{p(\text{occ})p(z|\text{occ}) + p(\text{emp})p(z|\text{emp})} \tag{3}$$

The two conditional distributions $P(Z|\text{occ})$ and $P(Z|\text{emp})$ must be specified in order to process cell occupancy update. Defining these functions is an important part of many works ([9], [18]) and, in the following, the results in [19] which proves that for certain choice of parameters[2] these functions are piecewise constants:

$$p(z|[O_{x,y} = \text{occ}]) = \begin{cases} c_1 & \text{if } z < \rho \\ c_2 & \text{if } z = \rho \\ c_3 & \text{otherwise.} \end{cases} \tag{4}$$

$$p(z|[O_{x,y} = \text{emp}]) = \begin{cases} c_1 & \text{if } z < \rho \\ c_4 & \text{if } z = \rho \\ c_5 & \text{otherwise.} \end{cases} \tag{5}$$

when ρ is the range of the cell (x,y).

As explained in [10], the cell update requires operations that are not part of the set of wavelet vector operations[3] (product and quotient). Thus a better form is necessary to operate update on wavelet form of occupancy functions.

B. Log-ratio form of occupancy update

As the occupancy is a binary variable, a quotient between the likelihoods of the two states of the variable is sufficient to describe the binary distribution. The new representation used is:

$$\text{odd}(O_{x,y}) = \log \frac{p([O_{x,y} = \text{occ}])}{p([O_{x,y} = \text{emp}])} \tag{6}$$

[1]For a certain variable V we will note in upper case the variable, in lower case v its realization, and we will note $p(v)$ for $P([V = v])$ the probability of a realization of the variable.

[2]The sensor model failure rate, the sensor range discretization and the prior occupancy probability are the parameters. Prior occupancy is chosen very low, the world being assumed very empty. Only the last parameter is relevant for establishing the piece-wise constantness of the functions [19].

[3]product and quotient are not base inner operators of a vector space

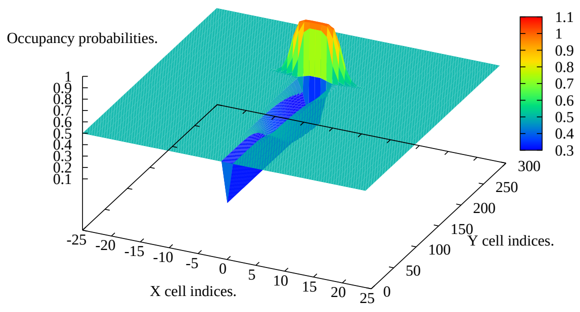


Fig. 3. Update of a 2D OG after a sensor reading, initially each cell occupancy was unknown, *i.e.* 0.5 probability. The sensor beam has an aperture of 7 degrees. The sensor is positioned in (0,0).

In the Bayesian update of the occupancy, the quotient makes the marginalization term disappear and thanks to a logarithm transformation, sums are sufficient for the inference:

$$\begin{aligned}\log\frac{p(\mathrm{occ}|z)}{p(\mathrm{emp}|z)} &= \log\frac{p(\mathrm{occ})}{p(\mathrm{emp})}+\log\frac{p(z|\mathrm{occ})}{p(z|\mathrm{emp})}\\ &= odd_0+odd(z) \qquad (7)\end{aligned}$$

Therefore the vector space generated by the wavelet basis with its sum inner operator is sufficient to represent and update OGs. This inference with sums was originally proposed by Elfes and Moravec [9], but only for performance reasons. Here it is also necessary to allow inference to be performed within the compressed data.

C. Log-ratio form of sensor model functions

It is straightforward to derive from eq. 4 and 5, the sensor model equations in log-ratio form that we note as the following:

$$\mathrm{odd}(z) = \begin{cases} 0 \text{ if } z>\rho \\ \log(c_2/c_4)=\mathrm{odd}_{\mathrm{occ}} \text{ if } z=\rho \\ \log(c_3/c_5)=\mathrm{odd}_{\mathrm{emp}} \text{ otherwise.}\end{cases} \qquad (8)$$

when ρ is the range of the cell (x,y). One can notice that the update term is zero if the cell is beyond the sensor readings, thus no update is required in this case.

IV. Hierarchical rasterization of polygon or polyhedron

This section describe the main contribution of this article which consists of a fast algorithm for updating an occupancy grid expanded as a non-standard Haar wavelet series from a set of range measurements.

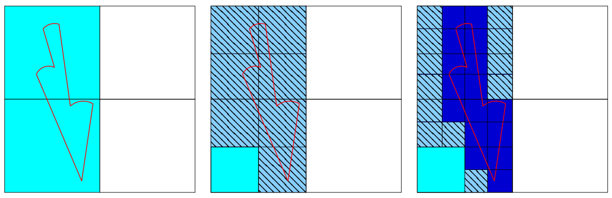


Fig. 4. The hierarchical process of updating the grid: from the coarsest scale to the finest. To save computing time, area that are outside the polygon of view or totally included inside the area classified as empty are detected and processed early in the hierarchy.

A. Problem statement

Given sensor position, beams geometry and measured ranges, it is possible to define the polygon (fig. 5) or polyhedron viewed by the sensor within the grid. Each time the sensor position changes or measured ranges changes a new relative position of the polygon or polyhedron and the grid must be computed in order to update the grid. The standard approach for updating occupancy grids, in the context of laser sensors, will be to traverse the cells along each laser sensor beam and update the cells. This method of traversal induces difficulties in calculating the area of coverage for each laser sensor beam in order to avoid inaccuracies such as aliasing. An easier alternative will be to traverse every cell of the grid and for each cell, perform a simple test to determine the state of the cell. In this case, with a grid size of 1024 cells per dimension, a 2D square grid contains more than 1 million cells and a 3D cubic grid contains more than 1 billion. Even if real-time performance can be obtained in 2D, it does not seem to be the case in 3D. Therefore the problem is to find a method that efficiently updates the grid without traversing every cell of the grid. As shown in fig. 5 and eq. 8, a range measurement defines three sets of cells. The first set, E, contains cells that are observed as empty. The second set, U, contains cells that are considered as unknown. The third set, B (for boundaries), contains cells that are partially empty, unknown or occupied. The elements of the third set are mainly found at the boundaries formed by the sensor beams at its two extreme angles and at the neighborhood of an obstacle. The remark in section III-C states that the U set can be avoided in the update process. Therefore an update step must iterate through the cells that intersect either the polygon in 2D or the polyhedron in 3D that describe the sensor beam boundaries (fig. IV). The following describes an algorithm that performs the correct iteration through the grid in an efficient manner through the utilisation of wavelets.

B. First hierarchical space exploration

The key idea in the exploration of the grid space is to define a predicate: *existIntersection* that is true if a given set of grid cells intersect the volume defined by the field of view of the sensor beams (blue plus red cells in fig. 5). The absence of intersection indicates that the given set of cells are outside the sensor field of view and don't need updating. For the case of

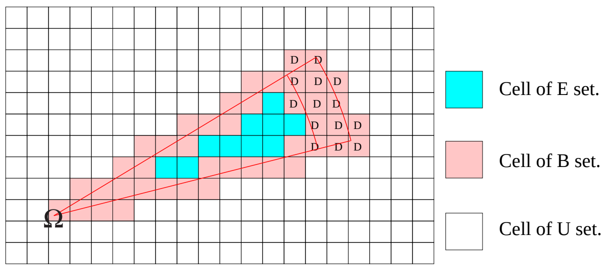


Fig. 5. A range-finder beam. The range finder is located at Ω and its field of view is surrounded by red boundaries. It defines the three kind of cell types. The band within the obstacle lies is at the top right end of the field of view. Thus the cells marked with a "D" stand for cells where a detection event occurs.

existIntersection evaluating to true, a special sub case would be when the set of cells are totally included in the sensor field of view, then all the cells of the set belong to E (blue cells in fig. 5) and their occupancy are decreased by the same amount of odd_{emp}, eq. 7.

As the algorithm is able to detect uniform regions recursively, the grid representation should allow the update of regions, and wavelets provide a natural mechanism for doing so. In this first version of the algorithm, the grid is traversed hierarchically following the Haar wavelet support partition. For each grid area, the *existIntersection* predicate guides the search. If there is intersection the traversal reaches deeper into the grid hierarchy, *i.e.* exploring finer scales. Otherwise it stops at the current node. Then the wavelet transform is performed recursively beginning from this last node as described in fig. 2 for the 2D case.

Algorithm 1 HierarchicalWavRaster(subspace S, sensor beam B)

```
1:  for each subspace $i$ of $S$: $i = 0, \dots, n$ do
2:    if sizeof($i$) = minResolution then
3:      $v_i$ = evalOccupancy($i$)
4:    else if existIntersection( $i$, $B$ ) then
5:      if $i \in E$ then
6:        $v_i = \text{odd}_{\text{emp}}$        /*eq. 8*/
7:      else
8:        $v_i$ = HierarchicalWavRaster( $i$, $B$ )
9:      end if
10:   else
11:     $v_i = 0$        /*$i \in U$*/
12:   end if
13: end for
14: $\{s_S^{l+1,obs}, d_{f_1,S}^{l,obs}, \cdots, d_{f_n,S}^{l,obs}\}$ =waveletTransform($\{v_0, \cdots, v_n\}$)
15: for each $d_{f,S}^{l}$: do
16:   $d_{f,S}^{l} \leftarrow d_{f,S}^{l} + d_{f,S}^{l,obs}$        /*update inference*/
17: end for
18: returns the scaling coefficient $s_S^{l+1,obs}$
```

Algorithm 1 gives the pseudo-code of the first hierarchical grid traversal. The algorithm is recursive and begins with the whole grid as the first subspace defined by the root of the wavelet tree. Its result is used to update the mean of the wavelet tree which is also the coefficient of the scaling function at the coarsest level. The *sizeof* function get the resolution of the subspace i and *minResolution* represents the resolution of a cell in the grid.
The *evalOccupancy* function evaluates the occupancy of a cell; it can proceed by sampling the cell occupancy.
Such an algorithm is very efficient in 2D but as it refines every area on the sensor beam boundaries it explores at least all the perimeter of the polygon of view in 2D (red cells in fig. 5). Equivalently in 3D, the explored part is all the surface of the polyhedron of view and it is to huge to be explored in real-time. That is why a better algorithm is required.

C. *Improved hierarchical space exploration*

In the space were a robot must evolve most of the space is empty. Thus it is not efficient to begin with a map initialized with a probability of 0.5 since this probability will evolve almost every where toward the minimum probability p_{emp}. Equivalently, since each boundary between an area observed as an empty one and an area outside the sensor field of view separates cells that are almost all empty, updating occupancy along this boundary is useless. Following this remark algorithm 1 is modified in a lazy algorithm that investigate finer iterations through the grid only if an update is required.

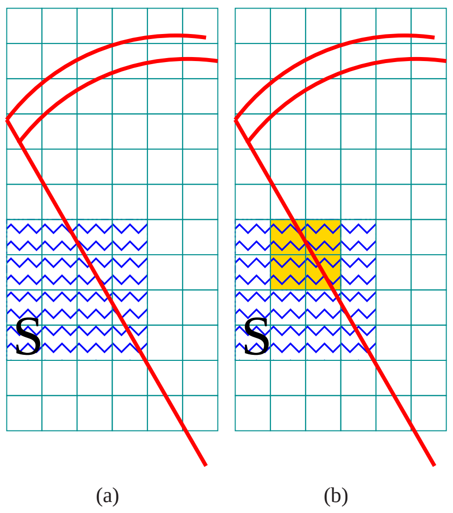


Fig. 6. Two different cases for the iteration along a boundary of the field of view that separates E set and U set. Fig. 6(a) artificial separation, S (with waves) was totally empty and the observation of a part of its interior (on the right of the red boundary) does not bring any information gain. Fig. 6(b) the separation brings information about the state of the yellow area that is inside the field of view (on the right of the red boundary).

An update is almost always required for cells that are in the obstacle neighborhood (cells marked with 'D' in fig. 5) so iteration is always performed in area that contains such a cell. But for boundaries that separate cells that belongs to U set and to E set (white and blue cells in fig. 5) iteration is required only if the E set corrects the knowledge in the grid (fig. 6(b)) otherwise the iterations can stop early in the hierarchy (fig. 6(a)).

Algorithm 2 HierarchicalWavRaster(subspace S, mean occupancy of subspace S: s_S^{l+1}, empty bound p_{emp}, sensor beam B)

```
    $\{v_0^g, \cdots, v_n^g\}$ = inverse
2:      WaveletTransform($\{s_S^{l+1}, d_{f_1,S}^l, \cdots, d_{f_n,S}^l\}$)
    **for** each subspace $i$ of $S$: $i = 0, \dots, n$ **do**
4:    **if** sizeof($i$) = minResolution **then**
        $v_i$ = evalOccupancy($i$)
6:    **else**
        spaceState = existIntersection( $i$, $B$ )
8:      **if** spaceState is UNKNOWN **then**
          $v_i = 0$
10:     **else if** spaceState is OCCUPIED **then**
          $v_i$ = HierarchicalWavRaster( $i$, $B$ )
12:     **else if** spaceState is EMPTY and $v_i^g > p_{\text{emp}}$ **then**
          $v_i$ = HierarchicalWavRaster( $i$, $B$ )
14:     **else if** spaceState is EMPTY **then**
          $v_i = \text{odd}_{\text{emp}}$        /*eq. 8*/
16:     **end if**
      **end if**
18:   $v_i^g \leftarrow v_i^g + v_i$        /*update inference*/
    **end for**
20: $\{\delta_S^{l+1}, d_{f_1,S}^l, \cdots, d_{f_n,S}^l\}$ =waveletTransform($\{v_0^g, \cdots, v_n^g\}$)
    returns the scaling coefficient $s_S^{l+1,obs} = s_S^{l+1} - \delta_S^{l+1}$
```

In algorithm 2 three main differences appears: first an inverse wavelet transform is performed to retrieve the information about the current state of the traversed subspace (line $1-2$). Second, line 7, the intersection function returns *OCCUPIED* only if the subspace intersect a neighborhood of an obstacle and it returns *EMPTY* if the subspace is included in $E \cup U$. Third the value of the minimum possible occupancy p_{emp} is a parameter of the algorithm in order to compare the state of the traversed subspace with information gain brought by the sensor observations (line 12).

The major difference between the maps produced by the first and the second algorithm is that in the second algorithm there is no *a priori* unknown area. Thus it is not possible anymore to store the position of the unexplored parts of the world. This could be a problem if one wants to drive the robot toward *terra incognita*. Nevertheless in the observation processing the information of unknown area is conserved such that occlusion are handled at the observation level.

One of the most important part of the previous algorithms are the intersection queries: the definition of *existIntersection*. This functions must be really optimized in order to retrieve fast algorithms. Each kind of telemetric sensor requires its own implementation of *existIntersection*. A simple implementation of such a function is easy to write since it involves only geometric intersection primitives, therefore we will not describe extensively one here for a lack of space. In our own implementation we have used an explicit representation of polygon or polyhedron of the sensor view with vertices and edges and implicit representation of a grid cell with its index. Then polygon-polygon or polyhedron-polyhedron intersection is computed, if this test fails an inclusion test is performed to test if one object is included in other.

algorithm	time (ms)	map size	mem.	nb. points	nb. cells
alg. 1 (2D)	29	$400m \times 200m$	$5MB$	107.6 10^6	8 10^6
alg. 2 (2D)	2.8	$400m \times 200m$	$5MB$	107.6 10^6	8 10^6
alg. 2 (3D)	30	$(50m)^2 \times 20m$	$23MB$	2.4 10^6	50 10^6

TABLE I
COMPUTING RESULTS OF THE 2 ALGORITHMS IN 2D AND 3D.

V. EXPERIMENTS

A. Computing time and required memory

We performed experiments[4] on 2D real data with the first and second algorithm with moving obstacles and on 3D simulated data with the second algorithm and with only static obstacles.

In the 2D experiment a big truck equipped with four SICK LMS-291 at each corner carries a big hot metal container behind it during a 1.5 hours experiment, the data are noisy and the evolution property of the map is required by the presence of a lot of moving obstacles (in particular the hot metal container). The algorithm processes the 4 laser range-finder at real-time ($40Hz$ each). Important number for evaluating algorithm efficiency are described in tab. V-A. Despite the inverse wavelet transform the second algorithm performs better than the first one (10 times faster in 2D). The localization was given by a dedicated algorithm. In the 3D simulation a rotating sick was simulated. The localization was given by the simulation and a noise was simulated on the range measurement. 3D occupancy grids are constructed using the approach described in this paper and fig. 7(a) and 7(b) shows two different views of the 3D occupancy iso-surface. In the 3D case the number of nodes in the wavelet tree is 178304 *i.e.* $5MB$ of data which compared to the 2414368 3D points gathered is less than 8%. The required memory of the whole tree structure is $23MB$ and the required memory is $200MB$ for the complete grid shows that the wavelet representation saves more than 91% of the required memory compared with a classic OG representation.

B. Qualitative results

For 2D and 3D grids, comparisons with a standard grid construction algorithm show that there are no significant differences. In the 3D results fig. 7, part of the grounds (on the right of the map) is not entirely mapped because the density of measurements is not uniform but depends on the vehicle velocity. As the map is considered empty *a priori* unseen parts of the ground appear as holes. Thus it would be interesting to use a ground model or ground inpainting procedure to initialize the map. Then, ground measurements would only correct the

[4]Every experiment was done with an Intel(R) Pentium(R) 4 CPU 3.00GHz.

a priori and that would save a lot of computing time, as the ground is the main obstacle.

VI. Conclusions and future works

A. Conclusions

The objective of this work is to present new algorithms that make OG building in wavelet space feasible. We show that wavelet space is naturally a good space to represent huge functions such as occupancy functions in 3D. In contrast to previous works, we don't need intermediate representation to build and fusion OGs in wavelet space with the new wavelet hierarchical rasterization algorithms. Thanks to the hierarchical organization of the computation, computing time is definitely sufficient for real-time in 2D and enough for real-time in 3D. With that achievement, the main contribution of this work is to present an OG representation which is also useful in 3D. Our long-term objective is to use the unified grid-based fast-slam framework in 3D environments. The requirements for an environment representation suitable for fast-slam are:

1) fast updating and scan matching to construct the map and calculate the current robot's pose in real time,
2) a hierarchical grid representation to handle multiple maps in multiple particles efficiently,
3) a small amount of memory per grid to ensure efficiency in the previously stated conditions.

Half of the first requirement and two other are fulfilled by this work, thus it is now possible to consider, very powerful algorithms such as a navigation grid-based fast-slam or grid-based multiple-target tracking in 3D based upon wavelet occupancy grids.

B. Future Works

In the future we will explore many important areas of improvement and it opens many research possibilities. As the intersection query is the most time-consuming part of the algorithm, we plan to work first on optimizing this part of the algorithm. Next, we will explore ground *a priori* to initialize the map efficiently. Another area of improvement is the kind of wavelets which is used to compress the map. Haar wavelets are the poorest kind of wavelets for compression properties, so it will be interesting to work with higher order wavelets that are able to compress much complex functions such as quadrics because it will approximate a map with locally Gaussian occupancy density in a far better way for example. Finally, the tree structure of the data allows parallel traversal of the environment and we plan to develop parallel instances of the wavelet rasterization algorithm. The proposed algorithm is a general one and its validity area is theoretically the set of all telemetric sensors.We plan to apply this algorithm with other kinds of telemetric sensors such as a stereo camera. However, our main objective is now to derive a localization algorithm based on this grid representation to obtain a complete grid based slam algorithm in 3D.

References

[1] J. Gutmann and K. Konolige, "Incremental mapping of large cyclic environments," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, Monterey, California, November 1999, pp. 318–325.

[2] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller, "An atlas framework for scalable mapping," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[3] C. Stachniss, G. Grisetti, and W. Burgard, "Recovering particle diversity in a Rao-Blackwellized particle filter for SLAM after actively closing loops," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[4] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 2443–2448.

[5] D. Cole and P. Newman, "Using laser range data for 3d slam in outdoor environments," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, Florida, 2006.

[6] D. C. P. Newman and K. Ho, "Outdoor slam using visual appearance and laser ranging," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, Florida, 2006.

[7] D. Hähnel, D. Schulz, and W. Burgard, "Map building with mobile robots in populated environments," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.

[8] K. P. Murphy, "Bayesian map learning in dynamic environments," in *NIPS*, 1999, pp. 1015–1021.

[9] A. Elfes, "Occupancy grids: a probabilistic framework for robot perception and navigation," Ph.D. dissertation, Carnegie Mellon University, 1989.

[10] M. Yguel, O. Aycard, and C. Laugier, "Wavelet occupancy grids: a method for compact map building," in *Proc. of the Int. Conf. on Field and Service Robotics*, 2005.

[11] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk, "The digital michelangelo project: 3D scanning of large statues," in *Siggraph 2000, Computer Graphics Proceedings*, ser. Annual Conference Series, K. Akeley, Ed. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000, pp. 131–144.

[12] S. Thrun, C. Martin, Y. Liu, D. Hähnel, R. Emery Montemerlo, C. Deepayan, and W. Burgard, "A real-time expectation maximization algorithm for acquiring multi-planar maps of indoor environments with mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 433–442, 2004.

[13] M. Hebert, C. Caillas, E. Krotkov, I. S. Kweon, and T. Kanade, "Terrain mapping for a roving planetary explorer," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, May 1989, pp. 997–1002.

[14] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *"Proc. of the International Conference on Intelligent Robots and Systems (IROS)"*, 2006.

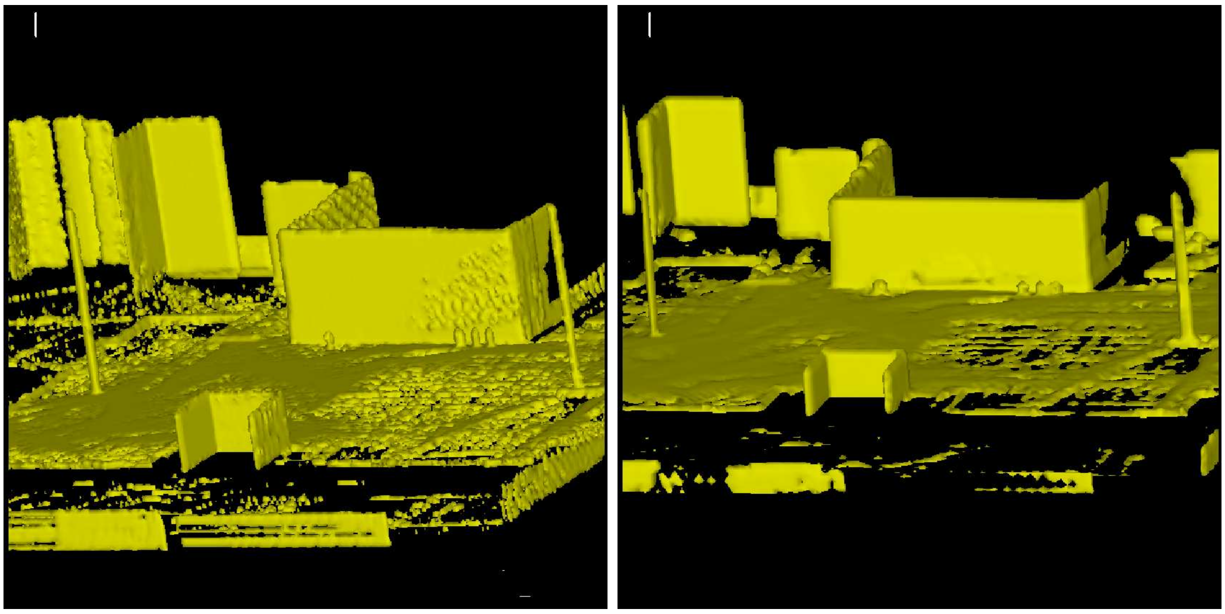
[15] I. Daubechies, *Ten Lectures on Wavelets*, ser. CBMS-NSF Series in Applied Mathematics. Philadelphia: SIAM Publications, 1992, no. 61.

[16] S. Mallat, *A Wavelet Tour of Signal Processing*. San Diego: Academic Press, 1998.

[17] E. J. Stollnitz, T. D. Derose, and D. H. Salesin, *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann, 1996.

[18] S. Thrun, "Learning occupancy grids with forward sensor models," *Autonomous Robots,*, vol. 15, pp. 111–127, 2003.

[19] M. Yguel, O. Aycard, and C. Laugier, "Efficient gpu-based construction of occupancy grids using several laser range-finders," *Int. J. on Vehicle Autonomous System*, to appear in 2007.



(a) (b)

Fig. 7. The wavelet OG obtained from a simulation of 3D data gathering with a rotating laser range-finder. Fig. 7(a) and 7(b) two views of the reconstruction of the grid from the wavelet grid at scale -1 (cell side of $0.20m$) and scale -2 (cell side of $0.40m$). It is noticeable that salient details as the lamp-post or the 4 pole landmarks before the wall are accurately mapped. The wall is interestingly smooth too, and that is a feature obtained by the oracle of the scaling view, details appear at finer views.