Dense Mapping for Range Sensors:
Efficient Algorithms and Sparse Representations

Manuel Yguel, Christopher Tay Meng Keat, Christophe Braillon,
Christian Laugier, Olivier Aycard

***Abstract*— This paper focuses on efficient occupancy grid building based on wavelet occupancy grids, a new sparse grid representation and on a new update algorithm for range 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], 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 a populated environment [5]. Some of the most impressive approaches are based on grid-based fast-slam algorithms [6] [7] [3], which offer a unified framework for landmark registration and pose calculation thanks to occupancy grids (OG) [8]. This approach yields several advantages: it provides 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 as the number of dimensions increases. 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 [4] have used landmark extraction, clustering and a special algorithm for spurious data detection. However, this map framework does not handle out-of-date data; therefore 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 in a hierarchical manner related to those of [9] but embedded in the wavelet theory which allows us to 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 the previous approach [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 in an anytime fashion.

There exists a large panel of dense mapping techniques: amongst the other popular representations of dense 3D data are raw data points [4], triangle mesh [11] [12], elevation maps [13], [14] or 2^d-tree based representations [15], [9]. 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 range 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 most serious 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 range 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 does not come 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 makes it possible to represent grids in a compact but flexible format. In pyramid maps representations [8], [9], information

is stored at each scale and there is a lot of redundancy but multiscale information is available. Conversely, probabilistic 2^d-trees record data at the leaves [15] [9] and the whole depth of the tree must be traversed to update the representation (fig. 3). Wavelet occupancy grids synthesize the advantages of both approaches: there is no redundancy and they make it possible to have multiscale editing by storing at finer scale only the differences with the previous coarser scale (see section II-B). Furthermore this representation allows compression by the elimination of redundant information where there are no significant additional details such as for empty or uncertain spaces with a theoretical analysis of information loss. In addition, this paper describe a real-time algorithm for hierarchical updating of the occupancy representation in 3D for the first time.

In order to build the map, a standard approach, [10], 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 areas of the environment and at the proper scale. In a first section, we will present the wavelet framework and the data structure while underlining differences with probabilistic 2^d-trees. 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 range 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 [8]. The wavelet transform known as the Mallat algorithm successively averages each scale, starting from the finest scale (fig. 1, from right to left). This produces an oracle predicting the information stored in 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. It is beyond the scope of this paper to give details about wavelet theory; references can be found in [16] [17] [18].

A. Notations

Wavelets are built from two sets 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 functions or coefficients 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 (from left to right) or compression (from right to left) processes.

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.

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 a 2D space or an octree for a 3D space. Each representation hierarchically maps the whole explored 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 in fig. 2. Only 3 coefficients in 2D remain at each leaf while at each node is recorded the mean occupancy of the underlying area. In a standard quadtree, 4 coefficients are necessary and, for example in [9], the storage of the mean is redundant, whereas in the wavelet representation it is not.

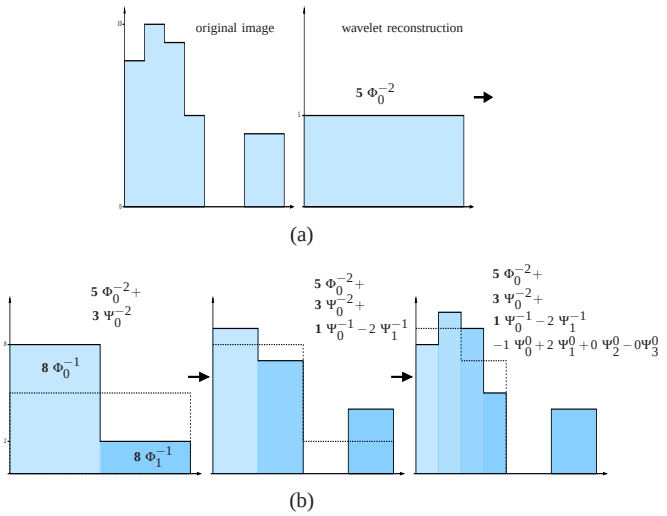


Fig. 1. The 1D image (upper, left) is: $[8,10,9,5,0,0,4,4]$, and its unnormalized (used here because it is simpler to display) Haar representation is: $[5,3,1,-2,-1,2,0,0]$. The image is then reconstructed one level at a time as follows: $[5] \rightarrow [5+3,5-3] = [8,2] \rightarrow [8+1,8-1,2-2,2+2]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

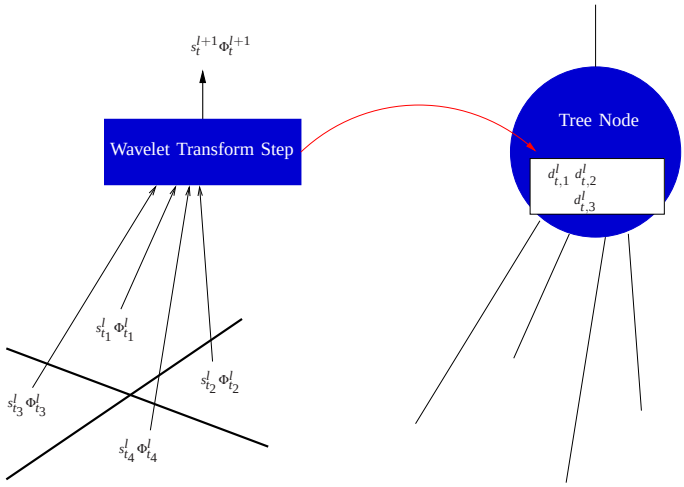


Fig. 2. A key step in a Haar wavelet transform in 2D. Four scaling samples at scale l generate 1 coarser scaling coefficient at scale $l+1$ and 3 detail coefficients at scale l that are stored in a wavelet tree node. In general the tree node has 4 children that describe finer resolutions for each space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

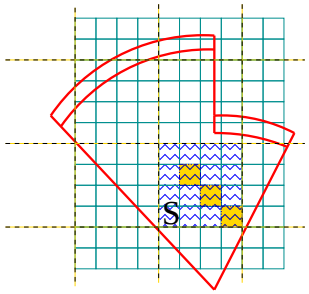


Fig. 3. Hierarchical updates are possible with wavelets: in a probabilistic quadtree all the pale/yellow squares are to be updated since they fall into the empty space. With wavelets the scale of the large square with waves is the last one that requires updating; therefore, the computation is efficient. The frontier of the coarse cells of the quadtree are marked in dashed/black lines.

The Haar wavelet data structure is exactly a 2^d-tree for its topology but not for the encoded data. Therefore the indexing of a cell with wavelet OG is as fast as with probabilistic 2^d-trees, however the retrieving of occupancy needs a small number of inverse wavelet transform operations[1]. Furthermore it not only stores spatially organized data, but also summarizes the data at different resolutions, which enables hierarchical updating. For example, fig. 3, the occupancy of all finest squares inside the empty area (fig. 6) decreases of the same amount, thus in the coarsest cell of the quadtree with waves the update is a constant. Also, the mean of the update over the waved square equals the value of the update for each finer cell, so all finer wavelet coefficients of the update are zero. As the update process is just the sum of the map wavelet representation with the update wavelet representation (section III), it produces efficient updates in areas that are coherent in the observation. Coherent areas are those that are adjacent to each other and have the same occupancy.

At the top of the structure, 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 or the bounding box of the observed places.

III. Occupancy Grids and Range Sensor Models

OG is a very general framework for environment modelling associated with range sensors such as laser range-finders, sonar, radar or stereoscopic video camera. Each measurement of the sensor consists of the range to the nearest obstacle for a certain direction. Thus a range measurement divides the space into three areas: 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 [8]. 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:

[1]The number of operations is a small constant (4 sums and 2 multiplications in 2D, 7 sums and 3 multiplications in 3D) per scale and the number of scales is the depth of the tree which is logarithmic ($\ln_4$ in 2D and $\ln_8$ in 3D) in the number of cells.

- Z a random variable[2] 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 the sensor model assumes cell independence. This leads to the following expression for the 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 4:

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

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 ([8], [19]). The results in [20] prove that for certain choices of parameters[3] these functions are piecewise constants:

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

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

where 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[4] (product and quotient). Thus a better form is necessary to operate updates on the wavelet form of occupancy functions.

B. Log-ratio form of occupancy update

As 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:

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

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:

[2]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.

[3]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 nearly empty. Only the last parameter is relevant for establishing that the functions are piece-wise constant [20].

[4]The product and the quotient operators are not base inner operators of a vector space.

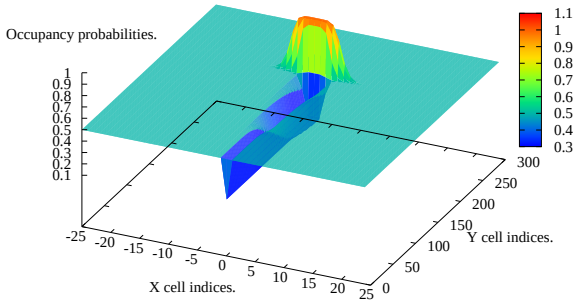


Fig. 4. Update of a 2D OG after a sensor reading, initially each cell occupancy was unknown, *i.e.* 0.5 probability. The sensor beam has an aperture of 7 degrees. The sensor is positioned in (0,0).

$$\begin{aligned} \log \frac{p(\text{occ}|z)}{p(\text{emp}|z)} &= \log \frac{p(\text{occ})}{p(\text{emp})} + \log \frac{p(z|\text{occ})}{p(z|\text{emp})} \\ &= \text{log-odd}_0 + \text{log-odd}(z) \end{aligned} \tag{7}$$

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 [8], 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 thus:

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

where ρ is the range of the cell (x,y), way to define each constant is given in [20][5]. 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.

A. Problem statement

Given the sensor position, the beam geometry and the measured ranges, it is possible to define the polygon (fig. 6) or polyhedron viewed by the sensor within the grid. Each time the sensor position changes or measured ranges change a new

[5]in the experiments: $\text{log-odd}_{\text{emp}} = -5.46$ and $\text{log-odd}_{\text{occ}} = 16.97$

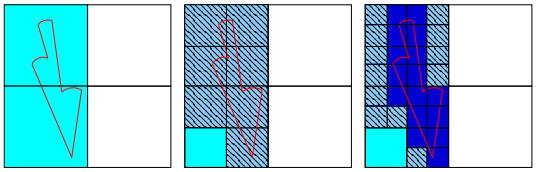


Fig. 5. The hierarchical process of updating the grid: from the coarsest scale to the finest. To save computing time, areas that are outside the polygon of view or totally included inside the areas classified as empty are detected and processed early in the hierarchy.

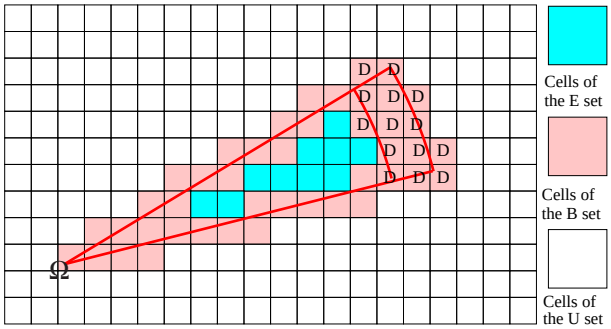


Fig. 6. A range-finder beam. The range finder is located at Ω and its field of view is surrounded by red/thick boundaries. It defines the three kinds of cell types. The band within the obstacle lies is at the top right end of the field of view. Thus the cells marked with a "D" stand for cells where a detection event occurs.

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 traversal method induces difficulties in calculating the coverage area 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. 6 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. 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. 5). The following describes an algorithm that performs the correct iteration through the grid in an efficient manner through the use of wavelets.

B. Exhaustive hierarchical space exploration

The key idea in the exploration of the grid space (fig. 5) is to define a predicate, *existIntersection*, which is true if a given set of grid cells intersect the volume defined by the field of view of the sensor beams (blue/dark gray plus red/medium gray cells in fig. 6). The absence of intersection indicates that the given set of cells are outside the sensor field of view and do not need updating. When *existIntersection* returns true, a special sub case needs to be considered in addition: if the set of cells is totally included in the sensor field of view, all the cells belong to E (blue/dark gray cells in fig. 6) and their occupancy is decreased by the same amount of log-odd$_{\text{emp}}$, eq. 7.

As the algorithm is able to detect uniform regions recursively, the grid representation should allow to update those 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 an 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)

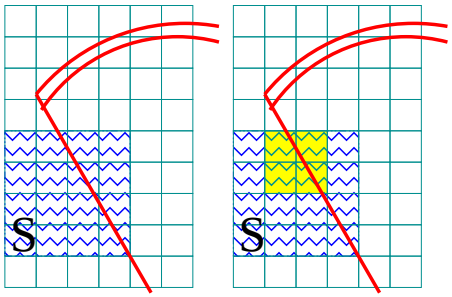
for each subspace i of S: $i = 0,\ldots,n$ **do**
  if sizeof(i) = minResolution **then**
    $v_i =$ evalOccupancy(i)
  else if existIntersection(i, B) **then**
    if $i \in E$ **then**
      $v_i = \text{log-odd}_{\text{emp}}$ /*eq. 8*/
    else
      $v_i =$ HierarchicalWavRaster(i, B)
    end if
  else
    $v_i = 0$ /*$i \in U$*/
  end if
end for
$\{s_S^{l+1,obs}, d_{f_1,S}^{l,obs}, \cdots, d_{f_n,S}^{l,obs}\} =$waveletTransform$(\{v_0, \cdots, v_n\})$
for each $d_{f,S}^{l}$: **do**
  $d_{f,S}^{l} \leftarrow d_{f,S}^{l} + d_{f,S}^{l,obs}$ /*update inference*/
end for
returns the scaling coefficient $s_S^{l+1,obs}$

Algorithm 1 gives the pseudo-code of the exhaustive hierarchical grid traversal. Here n is the maximum index of the space subdivisions that a node contains at one finer scale *i.e.* 3 for 2D and 7 for 3D Haar wavelet transforms. 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 gets 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 the whole perimeter of the polygon of view in 2D (red/medium gray cells in fig. 6). Equivalently in 3D, the explored part is all the surface of the polyhedron of view and it is far too large to be explored in real-time. That is why a better algorithm is required.

C. Improved hierarchical space exploration

Most of the space where a robot is to move about is largely empty. Thus it is not efficient to begin with a map initialized with a probability of 0.5 since this probability will decrease almost everywhere 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 investigates finer iterations through the grid only if an update is required.



(a) (b)

Fig. 7. Two different cases for the iteration along a boundary of the field of view that separates the E set and the U set. Fig. 7(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. 7(b) the separation brings information about the state of a part of an obstacle: the yellow/pale square area that is inside the field of view (on the right of the red/thick boundary).

An update is almost always required for cells that are in the obstacle neighborhood (cells marked with 'D' in fig. 6) so iteration is always performed in areas that contain such a cell. But for boundaries that separate cells that belong to the U set and to the E set, iteration is required only if the E set corrects the knowledge in the grid (fig. 7(b)); otherwise the iterations can stop early in the hierarchy (fig. 7(a)).

In algorithm 2 three main differences appear: 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*

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: $i = 0, \ldots, n$ **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**
$v_i = \text{log-odd}_{\text{emp}}$ /**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 $s_S^{l+1,obs} = s_S^{l+1} - \delta_S^{l+1}$

only if the subspace intersects 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 no longer possible 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 the information concerning unknown areas is used all the same in the definition of the polygon of view, so that occlusions are handled when observations are processed.
One of the most important parts of the previous algorithms are the intersection queries: the definition of *existIntersection*. These functions must be really optimized in order to retrieve fast algorithms. Each kind of range 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 one extensively here for lack of space. In our own implementation we have used an explicit representation of a polygon or polyhedron of the sensor view with vertices and edges and implicit representation of a grid cell with its index. Then the polygon-polygon or the polyhedron-polyhedron intersection is computed, if this test fails an inclusion test is performed to test if one object is included in the other.

The beams where a max-range reading occur, which could be produced by non-reflective surfaces, are safely replaced by totally unknown area. Therefore in presence of such readings the polygon/polyhedron is splitted into several parts connected by the laser-scanner origine.

V. Experiments

We performed experiments[6] on 2D real data sets and on 3D simulated data with noise. In the 3D simulations a rotating sick was used. For all the experiments the position of the laser were given. For real experiments, corrected data sets were used[7], whereas for simulated ones[8], the simulator provided us with the correct sensor position. For 2D and 3D both, we use data sets that contain only static obstacles and data sets that contain moving obstacles also. We test first and second algorithm on 2D data sets and only second algorithm on 3D data sets.

A. Methodology

c) Memory: for all experiments we give the memory used by an OG, a probabilistic-tree with and without mean stored and a wavelet OG at the end of the mapping. As probabilistic-tree and wavelet OG cover grids with size of power of two, we do not use OG with equivalent size to compare which would have been unfair. Instead, we compute the bounding box of all the range measurements and use an OG of the same size of that bounding box.

d) Computing time: in 2D we compare two things: first the use of the polygon rasterization algorithm against ray tracing with a Bresenham algorithm and OG rasterization using inverse sampling with a standard OG, second the hierarchical algorithms with probabilistic-trees and wavelet OG. We do not give results for other algorithms than polygon rasterization on hierarchical representations. Since all the cell accesses with the other algorithms are random accesses witch is totally inefficient with hierarchical rasterization, the results are not of interest. In 3D we only present the results with the wavelet OG representation. The mean, min and max of the update time per scan are computed.

e) Map quality: : in 2D, we evaluate map quality by comparing the resulting map with the one obtained by the OG rasterization using inverse sampling by computing the l_2 norm of the difference of the 2 grids. In 3D the quality is just visually estimated.

B. Results

f) Memory: These results show that the wavelet and probabilistic-tree performs the same concerning memory saving, witch follows the theory. As predicted, the Probabilistic-trees with the mean are however a bit more expansive. Both representations saves, in average, about 91% for 2D grids and 94% for 3D grids of the required memory compared with a classic OG representation. The amount of memory saved is larger in 3D than in 2D because the proportion of empty space is far more important.

[6]All experiments were done with an Intel$^{(R)}$ Pentium$^{(R)}$ IV CPU 3.00GHz.
[7]CSAIL (MIT), FR-Campus, FR079, FR101, thanks to Cyrill Stachniss [21]
[8]Inria static parking, Inria parking with a moving car

g) Computing time: For the comparison of update algorithm on the same OG representation, polygon rasterization and Bresenham performs almost the same witch is interesting since Bresenham does not handled beam width and is therefore far less accurate than polygon rasterization. They both performs far better than the inverse sampling. The second algorithm performs better for both representations: probabilistic-tree and wavelet OG, even if an inverse wavelet transform is computed in the last case (10 times faster in 2D, and 20 times faster in 3D). The probabilistic-tree performs better on static environments, although the difference is not of importance. Probabilistic-tree with mean performs only slightly better on static environments than wavelet OG, since, as wavelet OGs, they compute the mean. Concerning a dynamic environment, wavelet OG is slightly faster, which we reward the true multi-scale nature of this representation.

h) Quality: For 2D and 3D grids, comparisons with a standard grid construction algorithm show that there are no significant differences. In the 3D results (fig. 8), part of the ground (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 to initialize the map in the future works. Then, ground measurements would only correct the *a priori* and that would save a lot of computing time, as the ground is the main obstacle.

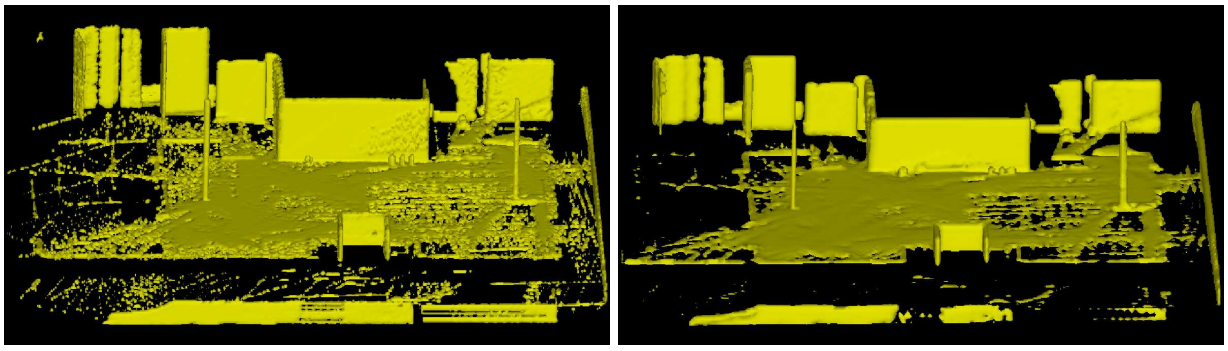
VI. Conclusions and future works

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 do not need intermediate representation to build and fuse 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 updating algorithm which is also useful in 3D. The use of Haar wavelets bring no significant computation speed-up or pittfall compare to probabilistic-tree with mean but is slightly better in memory saving. 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.

The last two requirements and half of the first one are fulfilled by this work; thus it is now possible to consider very powerful



(a) (b)

Fig. 8. The wavelet OG obtained from a simulation of 3D data gathering with a rotating laser range-finder. Fig. 8(a) and 8(b) provide 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.

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 several major areas of improvement. 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. Another area of improvement is the kind of wavelets that 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 more 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 hierarchical rasterization algorithm. The proposed algorithm is a general one and its validity area is theoretically the set of all range sensors. We plan to apply this algorithm using other kinds of range 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] 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.

[4] 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.

[5] 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.

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

[7] A. Eliazar and R. Parr, "DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks," in *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003, pp. 1135–1142.

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

[9] G. K. Kraetzschmar, G. Pagès Gassull, and K. Uhl, "Probabilistic quadtrees for variable-resolution mapping of large environments," in *Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, M. I. Ribeiro and J. Santos Victor, Eds., July 2004.

[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] P. Payeur, P. Hébert, D. Laurendeau, and C. Gosselin, "Probabilistic octree modeling of a 3-d dynamic environment," in *Proc. IEEE ICRA 97*, Albuquerque, NM, Apr. 20-25 1997, pp. 1289–1296.

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

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

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

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

[20] 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.

[21] C. Stachniss, "Corrected robotic log-files." http://www.informatik.uni-freiburg.de/ stachnis/datasets.html.