Efficient GPU-based Construction of Occupancy Grids Using several Laser Range-finders.

Manuel Yguel, Olivier Aycard and Christian Laugier
INRIA Rhône-Alpes,
Grenoble, France
email: name.lastname@inrialpes.fr

***Abstract*— Building occupancy grids (OGs) in order to model the surrounding environment of a vehicle implies to fusion occupancy information provided by the different embedded sensors in the same grid. The principal difficulty comes from the fact that each can have a different resolution, but also that the resolution of some sensors varies with the location in the field of view. In this article we present a new exact approach to this issue and we explain why the problem of switching coordinate systems is an instance of the texture mapping problem in computer graphics. Therefore we introduce a calculus architecture to build occupancy grids with a graphical processor unit (GPU). Thus we present computational time results that can allow to compute occupancy grids for 50 sensors at frame rate even for a very fine grid. To validate our method, the results with GPU are compared to results obtained through the exact approach.**

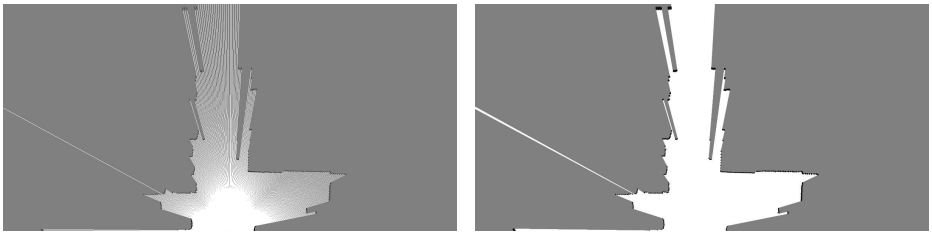
I. Introduction

At the end of the 1980s, Elfes and Moravec introduced a new framework to multi-sensor fusion called occupancy grids (OGs). An OG is a stochastic tessellated representation of spatial information that maintains probabilistic estimates of the occupancy state of each cell in a lattice [1]. In this framework, each cell is considered separately for each sensor measurement, and the only difference between cells is the position in the grid. For most common robotic tasks, the simplicity of the grid-based representation is essential, allowing robust scan matching [2], accurate localization and mapping [3], efficient path planning algorithms [4] and occlusion handling for multiple target-tracking algorithms [5]. The main advantage of this approach is the ability to integrate several sensors in the same framework, taking the inherent uncertainty of each sensor reading into account, contrary to the *Geometric Paradigm* [1], a method that categorizes the world features into a set of geometric primitives. The major drawback of the geometric approach is the number of different data structures for each geometric primitive that the mapping system must handle: segments, polygons, ellipses, etc. Taking into account the uncertainty of the sensor measurements for each sequence of different primitives is very complex, whereas the cell-based framework is generic and therefore can fit every kind of shape and be used to interpret any kind and any number of sensors. The opportunity of such a diversity is essential because it is highly useful to notice that the failure conditions are almost always different when switching from a sensor class to another. In fact, video cameras are very sensitive to changes in light conditions, laser range-finders are corrupted as soon as there is direct sun light in the receiver axis and ultra-sonic sensors are useless when the reflexion surface is very irregular. Moreover, the combination of redundant sensors limits the effects of sensor breakdown and enlarges the robot field of view.

For sensor integration OGs require a sensor model which is the description of the probabilistic relation that links a sensor measurement to a cell state, occupied (occ) or empty (emp). The objective is to build a unique occupancy map of the surroundings of an intelligent vehicle (the V-grid), equipped with several sensors that summarize all sensor information in terms of occupancy in their sensor model. As explained in section II, it requires to get a likelihood for each state of each cell of the V-grid per sensor.[1] But each sensor have its own coordinate system for recording measurements, that is with a particular topology: Cartesian, polar, spherical, etc, and a particular position and orientation in the V-grid. For example, every telemetric sensor that uses the time-of-flight of a wave, like laser range-finders, records detection events in a polar coordinate system due to the intrinsic polar geometry of wave propagation. Thus building a unique Cartesian occupancy grid involves to change from the sensor map (the Z-grid) to a local Cartesian map (the L-grid) and/then to transform the L-grid into the V-grid with the good orientation and at good position. In the following paper, a general statement of the problem is presented with an exact approach that solves this problem. In particular we obtain maps without holes, compared to the strong Moiré effect in maps obtained with the state-of-the-art line drawing method for laser range-finders [3] (Fig. 1(a)). However, the OG mapping process has obviously a computational cost that increases with the number of sensors and the number of cells; these parameters affect the precision of the representation. One of the major advantages of the OG framework is that all fusion equations are totally independent for each cell of the grid, which makes it possible to improve computing by allowing parallel algorithms. Thus in this paper we present two contributions:

- a general and exact algorithm for the switching of discrete coordinate systems, which we derived for the laser-range finder case and used as a criterion to evaluate the performances of other methods in terms of correctness,

[1] It is only necessary for the sensors that view the cell.



(a) (b)

Fig. 1. (a) 2D OG obtained by drawing lines with 1D occupancy mapping (for a SICK laser-range finder). The consequences are a Moiré effect (artificial discontinuities between rays far from origin). (b) 2D OG obtained from the exact algorithm. All the OGs are 60m × 30m with a cell side of 5cm, *i.e.* 720000 cells.

precision and computing advantages.
- a very efficient GPU implementation of multi-sensor fusion for occupancy grids including the switch of coordinate systems validated by the results of the previous method.

In the conclusions of the first study we demonstrate the equivalence between the occupancy grid sensor fusion and the texture mapping problem in computer graphics [6]. And in the second contribution, we present an efficient algorithm for graphical processor units (GPUs). Using the parallel texture mapping capabilities of GPU, we obtain a fast procedure of fusion and coordinate system switch. Thus, the experiments show that GPU allows to produce occupancy grid fusion for 50 sensors simultaneously at sensor measurement rate.

The paper is organized as follows: we present first mathematical equations of sensor fusion and the 1D equations of telemetric sensor model we use. Then we focus on the switch of coordinate systems from polar to Cartesian because for most telemetric sensors the intrinsic geometry is polar. Then we explain how to simplify the above switch of coordinate systems to improve the computational time with parallelism, taking into account precision and/or safety. Finally in the last sections we present our GPU-based implementation and the results of fusion obtained for 4 sick laser range-finders with centimetric precision.

II. Fusion in occupancy grids

A. Bayesian fusion for a grid cell and several sensors.

a) Probabilistic variable definitions:

- $\overrightarrow{Z} = (Z_1, \ldots, Z_s)$ a vector of s random variables[2], one variable for each sensor. We consider that each sensor i can return measurements from a set $\mathcal{Z}_i$.
- $O_{x,y} \in \mathcal{O} \equiv \{\text{occ}, \text{emp}\}$. $O_{x,y}$ is the state of the bin (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.

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

b) Joint probabilistic distribution: the lattice of cells is a type of Markov field and many assumptions can be made about the dependencies between cells and especially adjacent cells in the lattice [7]. In this article sensor models are used for independent cells i.e. without any dependencies, which is a strong hypothesis but very efficient in practice since all calculus could be made for each cell separately. It leads to the following expression of a joint distribution for each cell.

$$P(O_{x,y}, \overrightarrow{Z}) = P(O_{x,y}) \prod_{i=1}^{s} P(Z_i|O_{x,y}) \tag{1}$$

Given a vector of sensor measurements $\overrightarrow{z} = (z_1, \ldots, z_s)$ we apply the Bayes rule to derive the probability for cell (x, y) to be occupied:

$$p(o_{x,y}|\overrightarrow{z}) = \frac{p(o_{x,y}) \prod_{i=1}^{s} p(z_i|o_{x,y})}{p(\text{occ}) \prod_{i=1}^{s} p(z_i|\text{occ}) + p(\text{emp}) \prod_{i=1}^{s} p(z_i|\text{emp})} \tag{2}$$

For each sensor i, the two conditional distributions $P(Z_i|\text{occ})$ and $P(Z_i|\text{emp})$ must be specified. This is called *the sensor model* definition.

B. Telemetric sensor model of a time-of-flight range-finder

For the 1D-case, the sensor models, used here (eq. (7),(8)), are based upon the Elfes and Moravec Bayesian telemetric sensor models [1]. Now, is presented our own demonstration of the results of [1], which add a complete formalism to express the dependance of the sensor model to the initial occupancy of the grid cells. This initial hypothesis is called the *prior*.

The whole presentation is based upon the assumption that the telemetric sensor is of a time-of-flight type. This is an active kind of sensor which basically emits a signal with a fixed velocity v at time t_0, then receives the echo of this signal at time t_1, and then computes the distance of the obstacles from the source with: $d = \frac{t_1 - t_0}{v}$. We call Ω the source location of the emitted signal.

First, we consider an ideal case: when there are several obstacles in the visibility area, only the first one (in terms of time of flight) is detected.

1) Probabilistic variable definitions: Only one sensor is considered.

- $Z \in \{\text{“no impact”}\} \bigcup \mathcal{Z}$. Z belongs to the set of all possible values for the sensor with the additionnal value: “no impact” which means that the entire scaned region is free.
- $O_x \in \mathcal{O} \equiv \{\text{occ}, \text{emp}\}$. o_x is the state of the bin x either “occupied” or “empty”, where $x \in [|1; N|]$. N is the number of cells in the 1D visibility area for a single sensor shot.
- $G_x \in \mathcal{G}_x \equiv \{\text{occ}, \text{emp}\}^{[|1;N|]\setminus\{x\}}$. g_x represents a state of all the cells in the visibility area except the x one. G_x takes its values in the t-uples of cells $(c_1 = o_1, \ldots, c_{x-1} = o_{x-1}, c_{x+1} = o_{x+1}, \ldots, c_N = o_N)$ where c_i is the cell i (fig. 2).

2) Joint distributions: The probabilistic distribution describing the interaction between sensor values and a cell state is, following an exact Bayes decomposition:

$$P(Z, O_x, G_x) = P(O_x)P(G_x|O_x)P(Z|G_x, O_x)$$

- $P(O_x)$ is the prior: this is the probability that in a cell lies a surface that is reflective for the telemetric sensor used. In this case the cell is called occupied. We note the probability that a cell contains no reflective surface (empty): u.
- $P(G_x|O_x)$ is the probability that, knowing the state of a cell, the whole visibility area is in a particular state. Here, we make a strong assumption: we assume that the state of the cell x is non informative for the states of the other cells. So formally: $P(G_x|O_x) \equiv P(G_x)$. However not any hypothesis about the probability of some particular state of G_x is made. Then: the sole hypothesis is that $P(G_x)$ only depends on the number of empty or occupied cells[3].
- $P(Z|O_x, G_x)$ depends of the sensor, but forall $(o_x, g_x) \in [|1; N|] \times \mathcal{G}_x$, the distribution over Z depends only of the first occupied cell. Then we suppose that knowing the position of the first occupied cell q_k in the sensor view g_x, $P(Z|o_x, g_x)$ behaves as if there were only c_k occupied in all the area. We call this particular distribution over Z: the *elementary sensor model* $\mathring{P}_k(Z)$.

To compute $P(Z, O_x)$ we derive, now, equations for the marginalisation over all the possible states of G_x.

3) Marginalisation sum in the discrete 1-D case : The heart of the problem is to deal with the visibility of a bin. Considering a perfect case, the first occupied cell in the visibility area causes a detection. So knowing that the cell x is occupied, that cell is the last one which can cause a detection. Therefore we give the next definition.

a) Definition : we define A_x^k as the set of all t-uples of G_x type: $(c_1, \ldots, c_{x-1}, c_{x+1}, \ldots, c_N) \in \{\text{occ}, \text{emp}\}^{N-1}$ (Fig. 2) where:

[3]which is a more general modelling than the uniform choice made in [1].

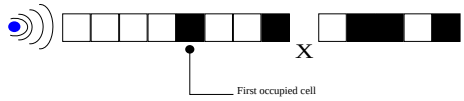


Fig. 2. In white (resp. black) the empty (resp. occupied) cells. An element of A_9^5, here $k < x$.

- $c_i = \text{emp}\ \forall i < k$
- $c_k = \text{occ}$

To derive the equations of the sensor models we use the following properties:

b) Properties:

1) $\forall (i, j),\ i \neq j,\ A_x^i \bigcap A_x^j = \emptyset$
2) $\bigcup A_x^k = \mathcal{G}_x \setminus \{\ (c_p)_{p \in [|1,N|]\setminus\{x\}} \mid \forall p, c_p = \text{emp}\ \}$
3) if $k < x$ there are k determined cells: the $k-1$ first cells: $(c_1, \ldots, c_{k-1})$, which are empty, and the kth: (c_k), which is occupied.
   Then $p(A_x^k) = u^{k-1}(1-u)$.
4) if $k > x$ there are $k-1$ determined cells: the $k-2$ first cells: $(c_1, \ldots, c_{x-1}, c_{x+1}, \ldots, c_{k-1})$ which are empty and the $(k-1)$th.: (c_k) which is occupied.
   Then $p(A_x^k) = u^{k-2}(1-u)$.

c) Distributions $P(Z|\ldots)$: the probability distribution over Z expresses the following semantic. Knowing that the cell x is occupied, the sensor measurement can only be due to the occupancy of x or of a cell before x in terms of visibility (Eq.4). So the probability that the measurement is 0, comes from the probability that the first cell is occupied, which is $1-u$ and produces a measurement in 0: $\mathring{P}_1([Z=0])$, and from the probability that the first cell is empty (u) and the second one is occupied and produces a measurement in 0: $\mathring{P}_2([Z=0])$ and so on ... Then we split the marginalisation sum into two complementary subsets of $\mathcal{G}_x$: the set of A_x^k such as x is not the first occupied cell and its complement (Eq.3). Then it leads to the following formula:

occupied case:

- if $Z \neq$ “no impact” :

$$\begin{aligned} p(Z|[O_x = \text{occ}]) &= \sum_{g_x \in \mathcal{G}_x} p([G_x = g_x])p(Z|[O_x = \text{occ}], [G_x = g_x]) \\ &= \sum_{k=1}^{x-1} p(A_x^k)\mathring{P}_k(Z) + (1 - \sum_{k=1}^{x-1} p(A_x^k))\mathring{P}_x(Z) \quad (3) \\ &= \sum_{k=1}^{x-1} u^{k-1}(1-u)\mathring{P}_k(Z) + u^{x-1}\mathring{P}_x(Z) \quad (4) \end{aligned}$$

As mentionned above eq. 4 has two terms: the left term in the sum that comes from the possibility that a cell before x is occupied and the right term that comes from the agregation of all the remaining probabilities around the last possible cell that can produce a detection event: x. In the case of a dirac *elementary sensor model*, the precision

is perfect and the agregation is completed at x fi g. 3(a). The "no impact" case ensures that the distribution is normalized.

- if $Z =$ "no impact":
  $p([Z = \text{"no impact"}]|[O_x = \text{occ}])$
  $= 1 - \sum_{r \neq \text{"no impact"}} p([Z = r]|[O_x = \text{occ}])$

- empty case:
  if $Z \neq$ "no impact" :
  we note open $= \{ \ (c_p)_{p \in [|1,N|] \setminus \{x\}} \mid \forall p, c_p = \text{emp} \ \}$

$$p(Z|[O_x = \text{emp}]) \quad (5)$$
$$= \sum_{g_x \in \mathcal{G}_x} p([G_x = g_x]) p(Z|[O_x = \text{emp}], [G_x = g_x])$$
$$= \sum_{k=1, k \neq x}^{N} p(A_x^k) \mathring{P}_k(Z) + p(\text{open}) \delta_{Z=\text{"no impact"}}$$
$$= \sum_{k=1}^{x-1} u^{k-1}(1-u) \mathring{P}_k(Z)$$
$$+ \sum_{k=x+1}^{n} u^{k-2}(1-u) \mathring{P}_x(Z) + u^{n-1} \delta_{Z=\text{"no impact"}} \quad (6)$$

There is three terms in the empty case: before the impact, after and the term "no impact". What is very interesting is that in both occupied and empty model the term before impact (left term) is exactly the same fi g. 3(a) and fi g. 3(b). As above, the "no impact" case ensures that every case is considered.

- if $Z =$ "no impact":
  $p([Z = \text{"no impact"}]|[O_x = \text{emp}])$
  $= 1 - (\sum_r p([Z = r]|[O_x = \text{emp}])) + u^{N-1} \delta_{Z=\text{"no impact"}}$

4) Some elementary sensor models *:*

- Dirac model: when the sensor has a standard deviation that is far smaller than the cell size in the occupancy grid, it is suitable to model $\mathring{P}(Z)$ with a dirac distribution[4] (Fig. 3(a)-3(b)):
  $\mathring{P}_k([Z = z]) = \begin{cases} 1.0 \text{ if } z = k \\ 0.0 \text{ otherwise.} \end{cases}$

- Gaussian models: as all telemetric sensors are far from perfect, the Dirac model is obviously inappropriate in many cases. At this point the traditional choice [**?**], [**?**], [**?**] favors gaussian distributions, centered on k and with a variance that increases with k.
  It models the failures of the sensor well. However in the case of a telemetric sensor the possible values for the measurements are always positive, but Gaussian assign non zero probabilities to negative values. Worst, close to the origin i.e. $z = 0$, this distribution assigns high values to the negative measurements. Therefore we propose the following discrete distribution (Fig. **??**) based on the gaussian[5] $\mathcal{N}(\mu, \sigma)$:

$$\mathring{P}_k([Z = z]) = \begin{cases} \text{if } z \in [0;1] : \\ \int_{]-\infty;1]} \mathcal{N}(k-0.5, \sigma(k-0.5))(u)du \\ \text{if } z \in]1;n] : \\ \int_{\lfloor z \rfloor}^{\lfloor z \rfloor + 1} \mathcal{N}(k-0.5, \sigma(k-0.5))(u)du \\ \text{if } z = \text{"no impact"} : \\ \int_{]n;+\infty[} \mathcal{N}(k-0.5, \sigma(k-0.5))(u)du. \end{cases}$$

Where $\sigma(x)$ is an increasing function of x. We notice that the probability of "no impact" is increased by the integral of the gaussian over $]n;+\infty]$, which means that all the impact surfaces beyond the sensor fi eld of view are not detected.
An other gaussian-based modelling was suggested in [8], to take into account all short reflections that could drive the echo to the signal receiver before the sensor has fi nished transmitting. These kind of telemetric sensor listen and emmit by the same channel, so the sensor cannot stand in both states: receiver and emmiter, at same time.

$$\mathring{P}_k([Z = z]) = \begin{cases} \text{if } z = \text{"no impact"} : \\ \int_{]-\infty;1] \cup]n;+\infty[} \mathcal{N}(k-0.5, \sigma(k-0.5))(u)du \\ \text{else:} \\ \int_{\lfloor z \rfloor}^{\lfloor z \rfloor + 1} \mathcal{N}(k-0.5, \sigma(k-0.5))(u)du. \end{cases}$$

Thus, we notice that the probability of "no impact" is increased by the integral of the gaussian over $]-\infty, 1]$. In this two modelling, introducing the special case of "no impact" is necessary to take the missed detections into account.

- lognormal model:
  To take into account the particular topology of a telemetric sensor model, we propose to defi ne $\mathring{P}(Z)$ with a lognormal based distribution:

$$\mathring{P}_k([Z = z]) = \begin{cases} \text{if } z = \text{"no impact"} : \\ \int_{]n;+\infty[} \mathcal{L}(\log(k-0.5), \sigma'(k-0.5))(u)du \\ \text{else:} \\ \int_{\lfloor z \rfloor}^{\lfloor z \rfloor + 1} \mathcal{L}(\log(k-0.5), \sigma'(k-0.5))(u)du \end{cases}$$

  where: $\mathcal{L}(M,S)(x) = 1/(S\sqrt{2\pi}x) e^{-(\ln(x)-M)^2/(2S^2)}$.
  It could be surprising to choose this type of modelling because the mean of $\mathring{P}_k([Z = z])$ is not in the middle of the k cell as with the other modelling. But, the median and the highest probability are in the middle of the k cell, and that is what we expect really for a telemetric sensor, i.e. there is as likelihood that the sensor return a measurement before the kth cell than after.

[4]Here we suppose that z is an integer which represents the cell index, which the sensor measurement corresponds to: if z is real it is $\lfloor z \rfloor + 1$.

[5]Here we assume that k is the index of the cell which represents all the points with radial coordinate in $]k-1;k]$, i.e. we assume a length of 1 for cell, for simplicity.

5) Consequences: in the dirac *elementary sensor models* case, the equations for the cell number ρ are:

$$p(z|[O_\rho = \text{occ}]) = \begin{cases} u^{z-1}(1-u) \text{ if } z < \rho \\ u^{\rho-1} \text{ if } z = \rho \\ 0 \text{ otherwise.} \end{cases} \quad (7)$$

$$p(z|[O_\rho = \text{emp}]) = \begin{cases} u^{z-1}(1-u) \text{ if } z < \rho \\ 0 \text{ if } z = \rho \\ u^{z-2}(1-u) \text{ otherwise.} \end{cases} \quad (8)$$

when $z \neq$ "no impact". Thus the equations of [1] holds if the uniform prior hypothesis $u = 1 - u = 1/2$ is used. It is very interessant to notice, that in the Dirac case, only three values are used to define the values of a sensor model all along the sensor field of view. For the other *elementary sensor models* proposed, only more values are needed close to the cell where a detection event occurs.

When $P(O_x)$ is uniform, the inference calculus gives:

$$p(\text{occ}|z) = \frac{p(z|\text{occ})}{p(z|\text{occ}) + p(z|\text{emp})}.$$

Thus in the case of all the above *elementary sensor models*, the following qualitative properties apply:

- if $x \ll r$ and $\forall k \in [|1, x|],\ \mathring{P}_k([Z = r]) \simeq 0$ which is the case for gaussian elementary sensor model, according to eq. 4, fig. 3(a) $p(z|\text{occ}) \simeq 0$ while according to eq. 6, fig. 3(b) $p(z|\text{emp}) > 0$.
  So $p([Z = r]|[O_x = \text{emp}]) \gg p([Z = r]|[O_x = \text{occ}])$ therefore:
  $$p(\text{occ}|r) \simeq 0$$
  It means that, if there is a measurement in r, there is no occupied cell before r.
- if $x \gg r$ then, almost only the left term in eq. 4 and eq. 6 are used to calculate the posterior and they are identical. Thus $p(\text{occ}|r) \simeq 0.5$
  That what ensures that after the impact all the cells have the same probability, which means: no state occupied of empty is preffered. That is the required behaviour because those cells are hidden. The equality holds in the dirac case but for other *elementary sensor models* it depends on the uncertainty in the location of the cell that produces the impact. For example, for gaussian *elementary sensor models* the equality numerically holds far enough -$4\sigma(k - 0.5)$ is enough- behind the impact location.

6) Error model: For modelling false alarm and missed detections it is possible to use a confidence model (or error model) like in [8]. A different confidence model can be used for each sensor so that it is possible to deal with the information about the amount of errors produced by each sensor. The principle is to consider a new variable:

- $D_i \in \mathcal{D} \equiv \{\text{on}, \text{off}\}$. D_i is the state of the measurement, either correct ("on") or wrong ("off").

Now, the joint distribution to define is:

$$P(0_x, \overrightarrow{Z}, \overrightarrow{D}) = P(O_x) \prod_{i=1}^{s} P(D_i) P(Z_i|O_x, D_i) \quad (9)$$

that is defining $P(D_i)$ and defining $P(Z_i|O_x, \text{off})$ and $P(Z_i|O_x, \text{on})$. Defining $P(D_i)$ corresponds to define $P([D_i = \text{off}])$ which is simply the probability that the ith sensor produced a wrong measurement. The previously defined $P(Z_i|O_x)$ is assign to $P(Z_i|O_x, \text{on})$ because it models the correct behaviour of the sensor. For $P(Z_i|O_x, \text{off})$, without any kind of information, a non-informative distribution which assign the same probability to each sensor measurement, is chosen for the two possible states, o_x, of the cell.
If there is no information about the current behaviour of the sensor, the used distribution is just the marginalization over all the possible state of each measurement:

$$P(O_x, \overrightarrow{Z}) = P(O_x) \prod_{i=1}^{s} \sum_{\mathcal{D}} P(D_i) P(Z_i|O_x, D_i) \quad (10)$$

Finally it is equivalent, to remplace each sensor model $P(Z_i|O_x)$ by the distribution:

$$p(\text{on}) P(Z_i|O_x) + p(\text{off}) \mathcal{U}(Z_i) \quad (11)$$

where $\mathcal{U}(Z_i)$ is a uniform distribution over $\mathcal{Z}_i$.
This kind of transformation of the sensor model adds a certain inertia related to the probability of wrong measurement. It means that a good sensor measurement must be received several times to be considered by the system as relevant as a sensor measurement without error model. This inertia is the price for the robustness added by the fault modelling.
An other very important feature added by the error model is that it implies that all the probabilities are non zero. Thus in eq. (2), $p(o_{x,y})$ is never zero neither for occ, nor for emp. If not, the cell occupancy would remain always the same whatever a sensor measurement is received. The consequence would be an occupancy building very sensitive to sensor failure. And it is also possible to use logarithm representation for probabilities which increases the numerical stability and is required for the implementation of the fusion process.

C. From 1D to 2D

The sensor model is defined in a ray (1D), and each cell in this ray is defined by its radial coordinate ρ, the telemetric sensor is supposed to return the cell number z where a detection event occurs.

The objective of the following section is to compute $P(Z|[O_{x,y}])$ from $P(Z|[O_\rho])$ for a ray with efficient and precise algorithms.

III. Change from polar to Cartesian coordinate system

To compare the measurements of two sensors at different positions on a vehicle, each of them providing measurements in its own coordinate system, the sensor information must be switched to a common frame. In the OG case, all the sensor

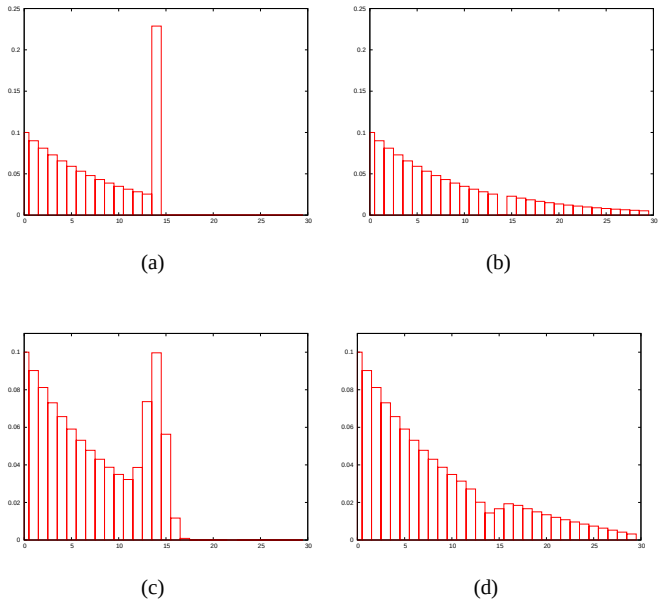


Fig. 3. probability distributions over the possible sensor range measurements knowing that the 14th cell is occupied (a),(c) (resp. empty (b),(d)), *a priori* over the occupancy of cells, the prior (u) is set to 0.1. (a),(b) dirac *elementary sensor models*; (c),(d) gaussian *elementary sensor models*.

model distributions must be switched from the Z-grid to the V-grid. In the first subsection, we give a general formalization of this problem which leads us to present an implementation of the exact solution. Finally we compare our exact algorithm with the classical approach in robotic and an adaptive sampling approach that leads us to present the equivalence of OG building with a texture mapping problem in computer graphics.

A. Problem statement

We use the word mesh for a planar subdivision of space whose geometric components are vertices, vertices that make edges and edges that make faces that are equivalent to cells in the OG formalism. We define a discrete coordinate system switch as the transformation that allows to define the same function for different meshes of the same space S. Given a mesh $\mathcal{A}$, origin, a mesh $\mathcal{B}$ goal where $\mathcal{B} \subset \mathcal{A}$ (i.e. each point in the surface covered by $\mathcal{B}$ belongs to $\mathcal{A}$ too) and 2 functions:

1) f: $F(\mathcal{A}) \rightarrow E$ where $F(\mathcal{A})$ is the set of faces in $\mathcal{A}$ and E a vector space,
2) h: $S \rightarrow S$ which makes a bijective transformation from a point of the goal to a point of the origin.

Thus it is possible to associate a point x of a certain face c in $\mathcal{B}$ to a point u of a certain face c' of $\mathcal{A}$.

the problem is to find a function g: $F(\mathcal{B}) \rightarrow E$ such as for each face $r \in F(\mathcal{B})$

$$\int_{t \in r} f(t)dt^6 = g(r).$$

[6]Here, we consider, for the integral, the Lebesgue measure for simplicity, but the formalism is general as soon as the measure of the intersection between any face of $\mathcal{A}$ and any face of $\mathcal{B}$ is well defined.

If there exists an analytical expression of f, and if h is differentiable and analytical expression of its derivatives exist, a gradient analysis gives exact analytic equations for the change of coordinate system through the following equation:

$$g(r) = \int_{t \in r} g(t)ds = \int_{t \in r} f \circ h(t)|Dh(t)|dt. \qquad (12)$$

where $Dh(t)$ is the Jacobian matrix of h in t and $|Dh(t)|$ its determinant. But in most cases in Bayesian modeling, functions are discretized due to learning processes or as the result of Bayesian inference. In our special case, we do not possess the analytical form of the sensor model (eq. (7),(8)).

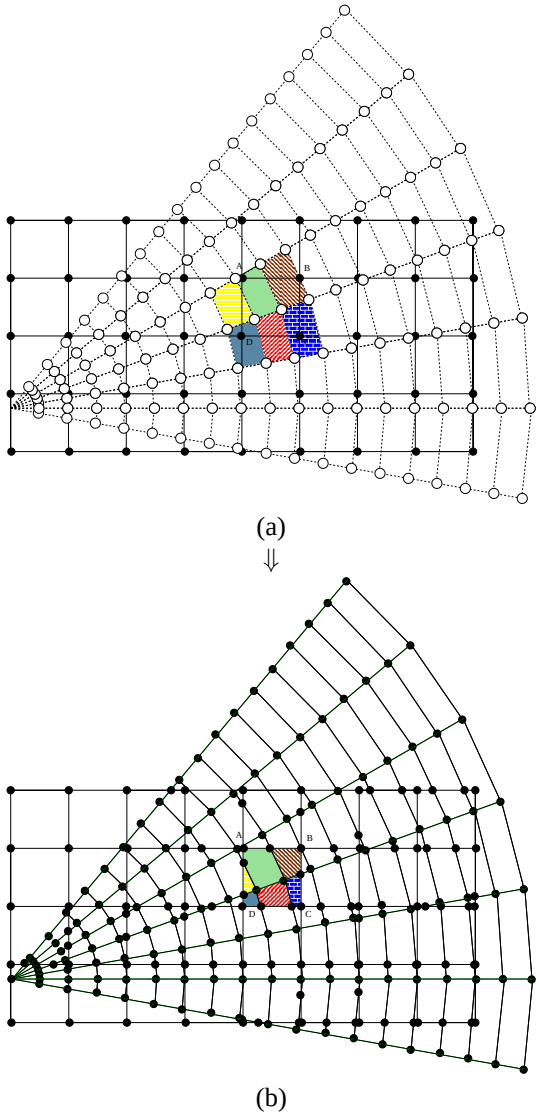


Fig. 4. (a) two subdivisions with dash lines and plain lines. In different color patterns: the different cells in the mesh $\mathcal{A}$ that intersect the ABCD cell of mesh $\mathcal{B}$ *i.e* I_{ABCD}. (b) overlaying the two subdivisions: adding vertex at each intersection of $\mathcal{A}$ and $\mathcal{B}$. The colored cells are the parts of the colored faces above that are included in ABCD.

1) The map overlay approach: the exact manner to compute $g(r)$ is to search all the faces of $\mathcal{A}$ that intersect r (Fig 4a):

let $I_r = \{u \in F(\mathcal{A}) | u \cap r \neq \emptyset\}$.
For each face i of I_r, let compute the surface, s, of $i \cap r$ and the surface, s_r, of r and keep their quotient $\frac{s}{s_r}$ (noted $s_{i,r}$). Then we obtain $g(r)$ with the following exact formula:

$$g(r) = \sum_{i \in I_r} s_{i,r} f(i). \tag{13}$$

So the problem comes down to computing, for each face r, its set I_r. This problem is called the map overlay problem in the computational geometry literature [9]. The complexity of the optimal algorithm [10] that solves this problem is $O(n \log(n) + k)$ in time and $O(n)$ in space where n is the sum of the numbers of segments in both subdivision $\mathcal{A}$ and $\mathcal{B}$ while k is the number of intersection points in both subdivisions. In the case of simply connected subdivisions the optimal complexity is $O(n + k)$ in time and space [11], and for convex subdivisions the optimal complexity is $O(n + k)$ in time and $O(n)$ in space [12]. This computation is very expensive, even in a simply connected subdivision and to use this approach a pre-computed map overlay is calculated off line.

2) Exact algorithm: To pre-compute the switching of coordinate systems an adaptation of the algorithm of Guibas and Seidel is used in order to obtain for each map of $\mathcal{B}$, the set of indexes of faces of $\mathcal{A}$ that intersect it and the surface of each of these intersections. We choose to work with convex subdivisions only, because it is easier to compute the surface of the intersections which therefore are also convex. Then for the switch from polar to Cartesian coordinate system, the algorithm is the following:

Algorithm 1 CoordinateSystemSwitch(polar $\mathcal{A}$, Cartesian $\mathcal{B}$)

```
mapping ← array(#(F(B)))
compute C(A): a convex approximation of A
compute the map overlay of C(A) and B
for each face f of the overlay do
    find i ∈ F(C(A)) and r ∈ F(B) such as f ⊂ i ∩ r.
    compute s = surface(f) / surface(r)
    append (r, s) to mapping[i].
end for
```

With this algorithm we have computed the map for the switch from polar to Cartesian coordinate system. It is possible to compute the two transformations, the one relative to topology and the one relative to position at the same time ,just by setting the relative positions of the two meshes.

B. Comparing the methods

In the next paragraphs, two methods are reviewed and are compared with the exact algorithm. We give quantitative and qualitative comparisons: the output probabilities values are compared and the maximal and average differences are shown, the average calculus time on a CPU is given then we focus on correctness and the possibility to have parallel algorithms. Our contribution in these comparisons is that, to the best of our knowledge, the exact algorithm was never used before.

1) The classical solution and the Moiré effect: as far as we know, all the OGs shown in the literature resort to line drawing to build the sensor update of laser range-finders [5], [3]. This method is simple to implement with a Bresenham algorithm and is fast because the whole space is not covered. But it presents several drawbacks. An important part of the map (all the cells that fall between two ray) fails to be updated. This is a well known problem, called the Moiré effect (fig. 1(a)) in computer graphics literature. This effect increases with the distance to the origin, and if the aim of the mapping is to retrieve the shape of objects and scan matching algorithms are used, the holes decrease the matching consistency. The maximal error (tab. I) is important because space is not well sampled and cells close to the origin are updated several times because several rays cross them. This ray overlapping induces bad fusion that makes some small obstacles appear or disappear.
This is an important issue: the V-grid has a certain resolution, *i.e.* a cell size and each sensor has its own resolution, thus a good OG building system must handle these differences. Interestingly, the OG building system allows the system to scale the grid locally to match the sensor resolution if precise investigations are needed, which means that all the the available information can be used.

2) Sampling approaches: The sampling approach is a common tool in computer graphics: in each cell of the Cartesian mesh a set of points is chosen, then the polar coordinates of those points are calculated, then the original values of the function f in those coordinates. Finally a weighted mean is calculated for the different values of f and is assigned to the Cartesian cell. Here the polar topology requires a non-regular sampling, *i.e.* the number of samples ns for each Cartesian cell is adapted according to the surface ratio of Cartesian and polar surfaces:

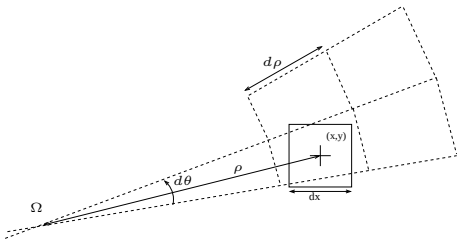


Fig. 5. Notations for the polar and Cartesian grids.

$$ns(x,y) = \frac{dx^2}{((\rho + \frac{d\rho}{2})^2 - (\rho - \frac{d\rho}{2})^2)d\theta} = \frac{dx^2}{\rho d\rho d\theta} \tag{14}$$

where ρ is a range associated to the point (x, y) and $d\rho, d\theta, dx$ are the steps of the two grids (Fig. 5).

This approach, called adaptive sampling, solves the problem of the singularity near the origin but still makes an approximation in choosing the location of the sample points and the

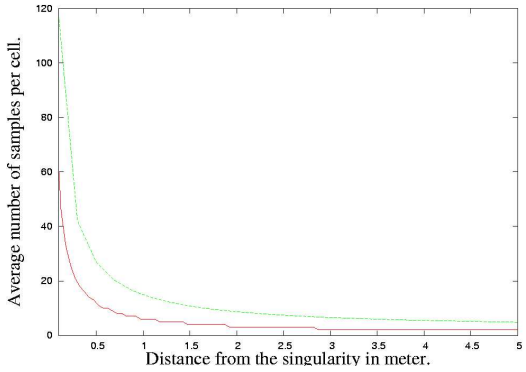


Fig. 6. In red, below: the analytical curve of the number of sample in adaptive sampling given by the ratio between Cartesian and polar surface. In green, above: cardinal of the I_r sets in the overlay subdivision provided by the exact algorithm. One can notice that the adaptive sample is an approximation because the curve is below the exact one. The sampling scheme is hyperbolic in the exact and approximate case.

according weight. The average and maximal errors (tab. I) are small compared to the line drawing algorithm; calculus time is, however, more expensive. The adaptive sampling is very close to the exact solution, in terms of the average number of samples per Cartesian cell, and of the repartition of the samples according to the distance with the singularity (fig. 6) and it is also closer int terms the quantitative error. Moreover the sampling method offers two advantages. From a computational point of view, it does not require to store the change of coordinate map, *i.e.* for each Cartesian cell the indexes of the polar cells and the corresponding weights that the exact algorithm requires. This gain is important not due to memory limitation but because memory access is what takes longest in the computation process of the above algorithms. From a Bayesian point of view, the uncertainties that remain in the evaluation of the exact position of the sensor in the V-grid have a greater magnitude order than the error introduced by the sampling approximation (this is even more so with an absolute grid[7]). The exactness in the switch of Z-grid to L-grid is relevant only if the switch between the L-grid and the V-grid is precise too. Thus in this context, a sampling approach is better because it is faster and the loss of precision is not significant, considering the level of uncertainty.

In these three methods, the exact algorithm and the sampling approach are parallel because each Cartesian cell is processed independently, whereas the line algorithm is not because the Cartesian cells are explored along each ray. The results in the tab. I are computed for a fine grid resolution: cell side of 0.5cm and a wide field of view: $60m \times 30m$, *i.e.* 720000 cells and one sick sensor that provides 361 measurements. The absolute difference between the log-ratios of occupancies are calculated to evaluate both average and maximal errors. The CPU used is an Athlon XP 1900+.

3) Equivalence with texture mapping: in computer graphics, texture mapping adds surface features to objects, such as a pattern of bricks (the texture) on a plan to render a wall. Thus the texture is stretched, compressed, translated, rotated, etc to fit the surface of the object. The problem is defined as a transformation problem between the texture image coordinates and the object coordinates [6]. The main hypothesis is that there exists a geometric transformation H that associates each object surface coordinate to each texture coordinate:

$$\begin{array}{rccc} H: & \mathcal{R}^2 & \rightarrow & \mathcal{R}^2 \\ & (x,y) & \mapsto & (u,v) = (u(x,y), v(x,y)) \end{array} \quad (15)$$

Let $g_a(x,y)$ the intensity of the final image at (x,y) and $T_a(u,v)$ the intensity of the texture at location (u,v) in continuous representation, the final intensity is linked to the texture intensity by:

$$g_a(x,y) \quad = \quad T_a(u,v) = T_a(u(x,y), v(x,y)). \quad (16)$$

The problem statement is how to define on the regular grid of the image representation in computer memory this continuous function. This is precisely a sampling problem and the geometric function H is a particular case of the h function above.

Just considering the problem in OG: for the occ state of the cells (for example) and for a certain measurement z in a ray, the sensor model of each polar cell can be considered as a texture: $p(z|[O_{(u,v)=(\rho,\theta)} = \text{occ}])$ that only depends of the (ρ,θ) coordinates. Thus the problem is to map this polar texture on its corresponding Cartesian space: a cone. The transformation function is the mapping between the Z-grid and the V-grid.

Method	Avg. Error	Max. Error	avg. time
exact	0	0	1.23s (CPU)
line drawing	0.98	25.84	0.22s (CPU)
sampling	0.11	1.2	1.02s (CPU)
GPU	0.15	1.8	0.049s on MS 0.0019s on board

TABLE I
COMPARISON OF CHANGE OF COORDINATE SYSTEM METHODS.

The equivalence between texture mapping and occupancy grid building, is part of a strong link between images and OG [1], and it suggests to investigate methods and hardware used in computer graphics to process this key step of OG building, as done in the following.

IV. CHANGE OF COORDINATE SYSTEM ON GPU

The GPU components for texture mapping rely on basic approximation procedures and for each of them it often exists dedicated process units that achieve the associated calculus. The basics of the process are

- the way to choose the cells in $\mathcal{A}$ that are used in the calculus for a cell $r \in \mathcal{B}$. That is the sampling step.

[7] in a slam perspective, for example

- the way to choose the weight associated with the different chosen cells in $\mathcal{A}$. That is the interpolation step.

When defining the mapping for a cell of the goal mesh, $\mathcal{B}$, two great cases arise depending of the number of original cells needed to deduce the value for a goal cell. An intuitive evaluation for this number could be made with the ratio between the surface of the cell in the goal mesh and mean surface of the associated cells in the original mesh, $\mathcal{A}$. Cells in the goal mesh can have a surface:

1) far lower (Fig 7(a));
2) comparable (Fig 7(b));
3) far greater (Fig 7(c))

than the corresponding cells in the original mesh.

The two first cases are handled identically with a *magnification* texture mapping and required only the fundamental part of sampling and interpolation processes that is described in subsection IV-A. The third case, called *minification*, corresponds to what happens close to the polar origin in the change of coordinate system from polar to Cartesian. The main idea to process this case is to find a transformation of the original mesh to get back a magnification case. And the dedicated transformation to achieve that process is described in the subsection IV-B.

A. Sampling and interpolation schemes for magnification

In graphical boards, all the information stored are mapped on matrices of bytes. Therefore all the definitions of the choice of sampling points and the choice of interpolation weights are given for transformations between matrices. Thus origin and goal cell values are accessed via integer coordinates in rectangular matrices. And all the geometrical transformations are inside the definition of F (Eq. 15) which transform continuous matrice coordinates into other continuous matrice coordinates.

1) One dimensional interpolation: let us consider two 1D-regular meshes (fig 8). The two meshes just differ by a translation and the problem is to evaluate the function defined on $\mathcal{A}$ on the center of the cells of $\mathcal{B}$. Let x the real coordinates of the center of a cell in $\mathcal{B}$ and $u = F(x)$ the real coordinates of x in the memory representation of $\mathcal{A}$.
This is a very simple case of magnification because each of the cells have identical shape, in this case a cell of $\mathcal{B}$ could overlap at most two cells of $\mathcal{A}$. The coordinates of the overlapped cells are:

- $i_0 = \lfloor u - 1/2 \rfloor$
- $i_1 = i_0 + 1$

In this simple case a linear interpolation realizes an exact change of coordinate system, the weights are defined by:

$$\begin{aligned} w_0 &= 1 - frac(u - 1/2) \\ w_1 &= 1 - w_0 \end{aligned}$$

and the value in $\mathcal{B}$ is:

$$g(x) = w_0 T_a(i_0) + w_1 T_a(i_1) \tag{17}$$

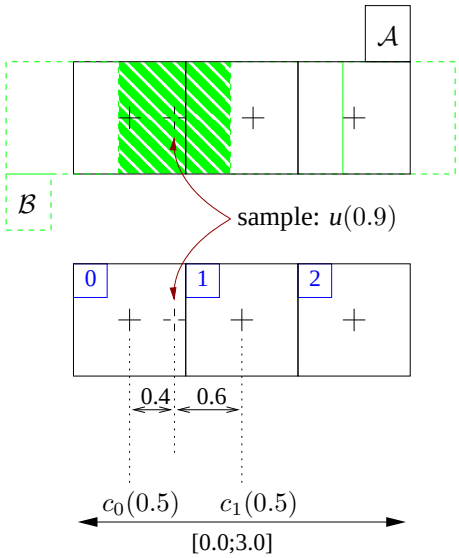


Fig. 8. The center of the cell of $\mathcal{B}$ is the sample, u is its coordinate in the $\mathcal{A}$ memory frame. The fractional part of $u - 1/2$, $w = 0.4$ is exactly the distance from u to the center of the cell 0 of $\mathcal{A}$ and $1 - w = 0.6$ is the distance from u to the center of the following cell 1 of $\mathcal{A}$. A linear interpolation keeps the greatest weight from the closest cell of the sample: thus 0.6 from c_0 and it remains 0.4 from c_1.

2) Two dimensional interpolation: the process in 2D is a combination of the 1D-process on the rows and on the columns. Thus it provides four samples and four weights and it is again exact for Cartesian regular meshes that differ only by a translation. In the other cases, it gives a good approximation because in a magnification case a cell of $callB$ overlap almost between one and four cells of $\mathcal{A}$ and the interpolation process guarantees that the closest cell of the sample is the main contributor to the value for the goal cell. Let (x, y) the real coordinates of the center of a cell in $\mathcal{B}$ and $(u, v) = F(x, y)$ the real coordinates of (x, y) in the memory representation of $\mathcal{A}$. Since the goal cell has smaller or equal size compare to the original cells, the approximated number of original cells overlapped by the goal cell is fixed to 4. Thus sampling is defined with four points whose coordinates are: $\{(i_0, j_0); (i_0, j_1); i_1, j_0); (i_1, j_1)\}$ where

- $i_0 = \lfloor u - 1/2 \rfloor$ and $j_0 = \lfloor v - 1/2 \rfloor$
- $i_1 = i_0 + 1$ and $j_1 = j_0 + 1$

The weights in the interpolation follow a linear model: the closer to the sample (u, v), the larger they are. They are based upon the two numbers:

$$\begin{aligned} w_\alpha &= 1 - frac(u - 1/2) \\ w_\beta &= 1 - frac(v - 1/2) \end{aligned}$$

where $frac(x)$ denotes the fractional part of x. The final interpolated value for the 2D function is then:

$$\begin{aligned} g(x, y) = & \\ & w_\alpha w_\beta T_a(i_0, j_0) + (1 - w_\alpha) w_\beta T_a(i_1, j_0) \\ + & \; w_\alpha (1 - w_\beta) T_a(i_0, j_1) + (1 - w_\alpha)(1 - w_\beta) T_a(i_1, j_1) \end{aligned} \tag{18}$$

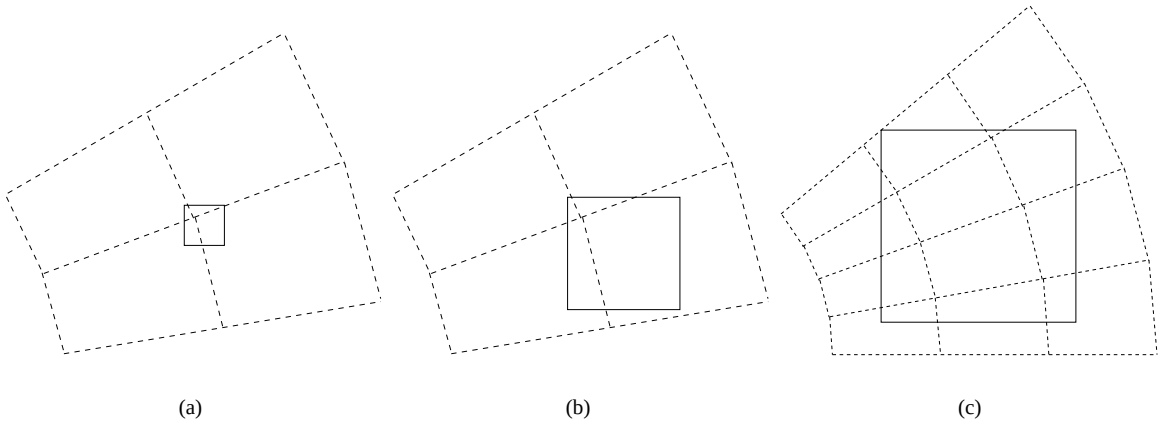


Fig. 7. (a) and (b) cases of magnification: the goal cell overlap with few cells in the original mesh. (c) case of minification: the goal cell overlap many cells in the original mesh.

B. Minification and mipmapping

The definition of magnification given above was approximate, thus let us give a precise one now.

1) Minification definition: let us define ν:

$$\nu = \max\left\{\sqrt{\frac{\partial u}{\partial x}^2 + \frac{\partial v}{\partial x}^2}; \sqrt{\frac{\partial u}{\partial y}^2 + \frac{\partial v}{\partial y}^2}\right\}$$

This value is related with the Jacobian (Eq. 19) in the continuous change of coordinate equation (Eq.12) for the 2D case:

$$|DH(x,y)| = \begin{vmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{vmatrix}. \quad (19)$$

It gives the maximum of the norms of each of the column of the Jacobian (Fig. 9).

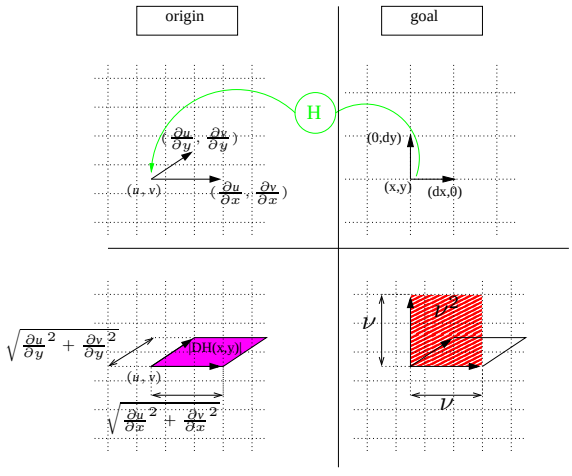


Fig. 9. Up right: an elementary displacement in the goal. Up left: the equivalent displacement in the original space given by the derivatives of the backward transformation H. Bottom left: in purple, a geometric view of the Jacobian. Bottom right: in red, a geometric view of the surface value chosen by the graphical board for defining the number of samples: it is a upper bound since the parallelogram has a smaller area than the square constructed from its larger side.

The Jacobian is the area of the image of the surface defined by an elementary displacement (dx, dy) in the goal space by H (Fig. 9). So ν is the maximum distance covered in the original space for an elementary displacement in the direction x or in the direction y in the goal space. Thus ν^2 is an upper bound for the area covered in the original space by an elementary displacement (dx, dy).

To define if there is minification: a comparison is made between the areas in the origin and in the goal through ν. If ν is lower than $\sqrt{2}$ there is magnification otherwise minification. The natural choice would be to compare ν to 1.0 but it is not optimal since four samples are used to evaluate the function in the magnification process.

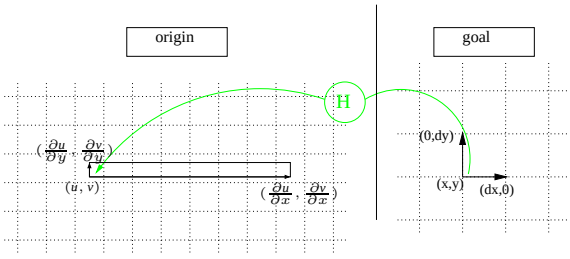


Fig. 10. When a derivative is large whereas the other is small, the area of the cell in the origin can be small. But as a large part of the original mesh is covered a large number of samples is necessary to compute the goal value thus minification sampling is required.

In the calculus of ν the choice of the maximum instead of a product, for an area, is important to avoid the case where the derivative in a dimension is very large while in the other dimension the derivative is very small (Fig. 10). In this case the Jacobian is small but the number of cells covered in the original mesh is important. In last generation of graphical board this problem is handled specifically with anisotropic sampling and there are two kind of ν, one for each dimension. Derivatives in the graphical board are approximated by calculating finite differences using the

evaluation of H for each corner of the goal cell.

a) For example in the Cartesian to polar mapping: :

$$\left| \begin{array}{cc} \frac{\partial \rho}{\partial x} & \frac{\partial \rho}{\partial y} \\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial y} \end{array} \right| = \left| \begin{array}{cc} \cos(\theta) & \sin(\theta) \\ -\sin(\theta)/\rho & \cos(\theta)/\rho \end{array} \right| = \frac{1}{\rho} \qquad (20)$$

and

$$\nu^2 = \max \left\{ (\cos^2(\theta) + \frac{\sin^2(\theta)}{\rho^2}); (\sin^2(\theta) + \frac{\cos^2(\theta)}{\rho^2}) \right\}$$

Eq. 20 explains the number of sample choices in the adaptive sampling (Eq. 14).

2) Sampling and interpolation formulas: mipmapping: in a case of minification a cell in the goal grid overlap several cells in the original one. Thus the solution chosen by the graphical board is to compute an appropriate coarse resolution of the mesh $\mathcal{A}$ which gives a new grid with which magnification could be used. Then the process of texture mapping is:

1) pre-compute several resolutions of the texture $\mathcal{A}$;
2) calculate for each pixel of $\mathcal{B}$ the two closest pre-computed resolutions;
3) for the two resolutions calculate the magnification values;
4) interpolate linearly between the two magnification values based on the distance between appropriate resolution and pre-computed ones.

To change from one resolution to a coarser one, each dimension is divided by a power of 2. That provides a pyramid of textures and the appropriate resolution is given by:

$$\lambda(x, y) = \log_2[\nu(x, y)].$$

Then the two closest resolution are given by:

- $d1 = \lfloor \lambda(x, y) \rfloor$
- $d2 = d1 + 1$

The magnification rules for sampling and interpolating are then applied to each of the selected texture, yielding two corresponding values $g^1(x, y)$ for the d_1 resolution and $g^2(x, y)$ for d_2 resolution. The final value for the cell (x, y) is then found as a 1D interpolation between the two resolutions:

$$g(x, y) = (1 - frac(\lambda(x, y)))g^1(x, y) + frac(\lambda(x, y))g^2(x, y)$$

The process of using several resolution of the original mesh is called mipmapping and is accelerated by graphical boards. These texture mapping schemes are part of the OpenGL2.0 specification, [13].

Thus it just remains to the programmer to define H which is the change of coordinate function. This definition could be done by providing to the graphical board the result of each of the required evaluation of H. It is also possible to draw geometric primitives: triangles, quadrilaterals or polygons which vertex are given in the goal coordinate system and are also associated with corresponding coordinates in the original coordinate system. Between each vertex interpolation is made by the graphical board to deduce all required coordinates.

Fig. 11. A map of the values of the occupied sensor model in polar coordinates, θ in abscissa and ρ in ordinate at different scales. They coorespond to 5 level of mipmaps and are used to calculate the change of coordinate system. The three colors represents the three possible values in a ray as explained in II-B.5 (they do not represent the real numerical values).

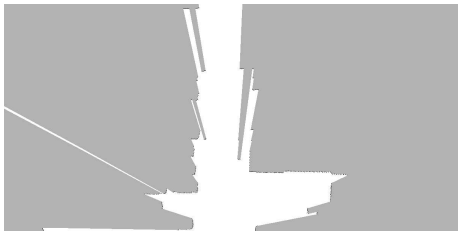


Fig. 12. Occupancy grid generated by the GPU. Compared to fig. 11 the geometric transformation apply, and each column in fig. 11 is transformed in a triangle in the final grid. The result weakly differs from fig. 1(b) with a calculus time many order faster.

Therefore the first method is more precise but computationally more expensive than the second.

In the case of polar to Cartesian change of coordinate system two matrices are drawn with (ρ, θ) coordinates one for occ and one for emp. Let the *polar sensor model grids*those matrices. Each column of a matrix corresponds to one angle and one range measurement and in this column is plotted the sensor model corresponding to the current range measurement. Then mipmaps of the two matrices are computed. Finally the change of geometry is processed by drawing geometric primitives: for each range measurement the corresponding quadrilateral is drawn in the Cartesian grid, each of the vertex of the quadrilateral is associated with the corners of the line of the 1-D sensor-model.

V. Results: comparison between exact sampling and GPU sampling

We test all the algorithms and in particular the GPU one on real data, *i.e.* 2174 sick scans. We obtain the results that are summarized in the last line of tab I. We made a simulation

with 4 Sicks to compare fusion results too and we obtained the following results: Fig 13.

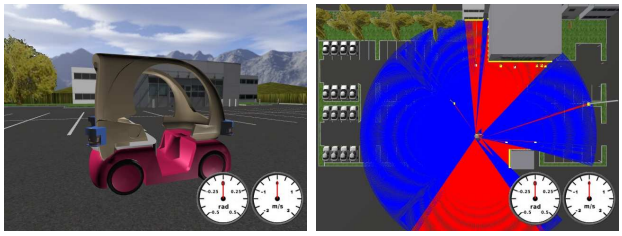


Fig. 13. Fusion view for 4 Sicks LMS-291.

A. Precision

The obtained precision is close to the exact solution, not as close as with the adaptive sampling method but far better than with the state-of-the-art method. Details close to the vehicle are well fit and any kind of resolution could be achieved for the OG. To avoid the infinite increasing of the required precision close to the sensors and for safety, we choose to consider the worst occupancy case for every cell that lies within a $30cm$ radius around the sensor. Outside this safety area the remaining error is almost null so that when considering these particular grids, precision is very close to that obtained with the exact algorithm.

B. Performance

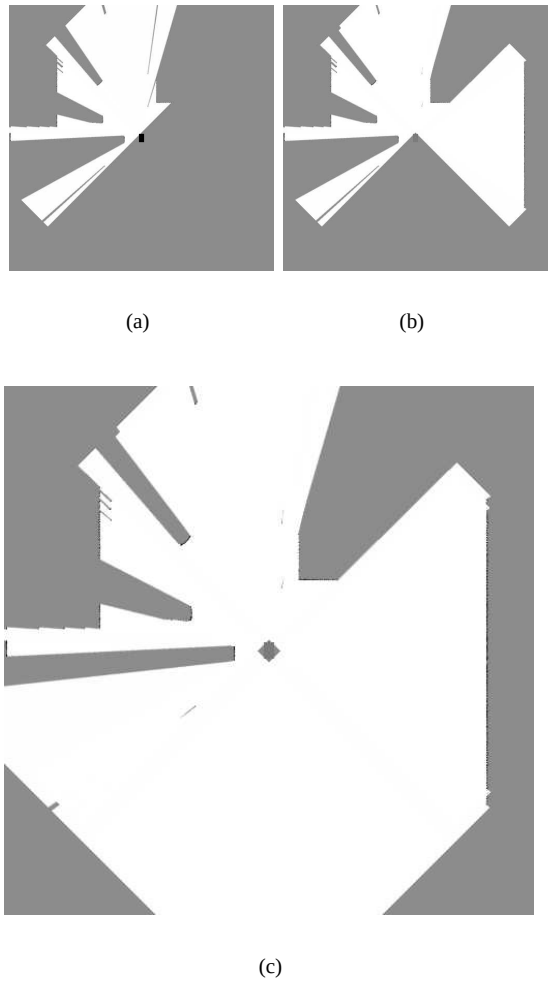
To evaluate the results an NVIDIA GeForce FX Go5650 for the GPU is used (tab I). For the GPU, two calculus times are given: first the computation time with the result transfer from the GPU to the CPU in memory main storage (MS) and second without this transfer. The difference is important and in the first case most of the processing time is devoted to data transfer, so if further computations were made on GPU, a lot of time could be saved. In this case the amazing number of 50 sensors can be computed at real-time with the GPU. It is important to note that, as only the result of the fusion needs to be sent to the main storage, a little more than half a second remains to compute OGs for other sensors and fusion when using a $10Hz$ measurement rate. So in the current conditions, 12 others sensors can be processed at the same time because the fusion process takes about as long as the occupancy computation, *i.e.* $2ms$.

VI. Fusion on GPU

Floating number have a limited precision, so to avoid numerical pitfalls, a logarithm fusion is very often used. On the actual graphical boards, floating precision is restricted to 32 bits, so this is an important issue of the sensor fusion on GPU, thus the logarithm fusion is presented here. 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 quotient makes the marginalization term disappear and thanks to a logarithm transformation, sums are sufficient for the inference.

$$\log \frac{p(\text{occ}|\overrightarrow{z})}{p(\text{emp}|\overrightarrow{z})} = \log \frac{p(\text{occ})}{p(\text{emp})} + \sum_{i=1}^{n} \log \frac{p(z_i|\text{occ})}{p(z_i|\text{emp})} \quad (21)$$

For each sensor, the two appropriate *polar sensor model grids*are constructed with the associated set of mipmaps. For each Cartesian cell, the two sensor models at the right resolution are fetched then an interpolated value is calculated from samples of each of them, then the log-ratio is calculated. The final value is added to the current pixel value. This process uses the processor units dedicated to transparency in the graphical board. The occupancy grid for each sensor appears as a layer where transparency decreases as the occupancy of the cell increases.



(a) (b)

(c)

Fig. 14. (a) V-grid with only the first sensor measurements. (b) V-grid with the fusion of the two first sensors measurements. (c) V-grid with the fusion of the four sensors.

And the final grid, fusion of all sensors, is just the sum of all the layers.

VII. Conclusion

Building occupancy grids to model the surrounding environment of a vehicle implies to fusion the occupancy information

provided by the different embedded sensors in the same grid. The principal difficulty comes from the fact that each sensor can have a different resolution, but also that the resolution of some sensors varies with the location in the field of view. This is the case with a lot of telemetric sensors and especially laser range-finders. The need to switch coordinate systems is a frequent problem in Bayesian modelling, and we have presented a new approach to this problem that offers an exact solution and which is absolutely general. This has lead us to evaluate a new design of OG building based upon a graphic board that yields high performances: a large field of view, a high resolution and a fusion with up to 13 sensors at real-time. The quality of the results is far better than with the classic method of ray tracing and the comparison with the exact results shows that we are very close to the exact solution. This new design of OG building is an improvement for environment-modelling in robotics, because it proves, in a theoretical and practical way, that a chip hardware can handle the task of fusion rapidly. The gain of CPU-time can therefore be dedicated to other tasks, and especially the integration of this instantaneous grid in a mapping process. In future works, we plan to explore the question of 3D OG modeling using graphical hardware. We will also investigate whether GPUs are suitable for other low-level robotic tasks. Videos and full results could be found at `http://emotion.inrialpes.fr/~yguel`.

REFERENCES

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

[2] 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.

[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] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on sub-optimality," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2004.

[5] D. Schulz, W. Burgard, D. Fox, and A. Cremers, "People tracking with a mobile robot using sample-based joint probabilistic data association filters," *International Journal of Robotics Research (IJRR)*, 2003.

[6] P. S. Heckbert, "Survey of texture mapping," *IEEE Comput. Graph. Appl.*, vol. 6, no. 11, pp. 56–67, 1986.

[7] S. Z. Li, *Markov Random Field Modeling in Image Analysis*. Springer-Verlag, 2001, series: Computer Science Workbench, 2nd ed., 2001, XIX, 323 p. 99 illus., Softcover ISBN: 4-431-70309-8. [Online]. Available: http://www.cbsr.ia.ac.cn/users/szli/MRF_Book/MRF_Book.html

[8] K. Konolige, "Improved occupancy grids for map building." *Auton. Robots*, vol. 4, no. 4, pp. 351–367, 1997.

[9] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwartzkopf, *Computational Geometry: Algorithms and Applications*. Springer, 1997.

[10] I. J. Balaban, "An optimal algorithm for finding segments intersections," in *SCG '95: Proceedings of the eleventh annual symposium on Computational geometry*. New York, NY, USA: ACM Press, 1995, pp. 211–219.

[11] U. Finke and K. H. Hinrichs, "Overlaying simply connected planar subdivisions in linear time," in *SCG '95: Proceedings of the eleventh annual symposium on Computational geometry*. New York, NY, USA: ACM Press, 1995, pp. 119–126.

[12] L. Guibas and R. Seidel, "Computing convolutions by reciprocal search," in *SCG '86: Proceedings of the second annual symposium on Computational geometry*. New York, NY, USA: ACM Press, 1986, pp. 90–99.

[13] M. Segal and K. Akeley, *The OpenGL Graphics System: A Specification*, Silicon Graphics, Inc., 10 2004. [Online]. Available: http://www.opengl.org/documentation/specs/version2.0/glspec20.pdf