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 tesselated 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 seperately 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 localisation and mapping [3], efficient path planning algorithms [4] and occultation 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 intrinsec 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

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

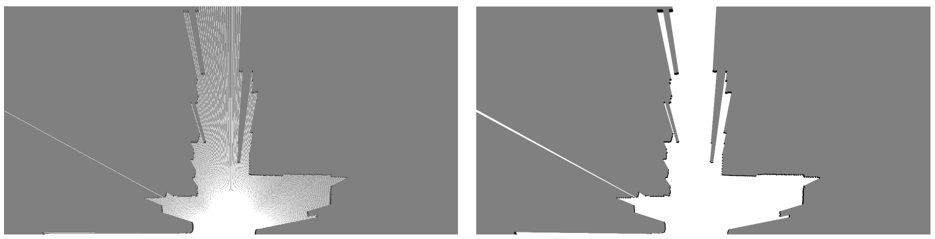
(a) (b)

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.

performances of other methods in terms of correctness, 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 intrinsec 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:

- $\overrightarrow{Z} = (Z_1, \dots, Z_n)$ a vector of n 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 $(x, y) \in \mathcal{Z}^2$.
  $\mathcal{Z}^2$ 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 realisation, and we will note $p(v)$ for $P([V = v])$ the probability of a realisation 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 independant 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 seperately. It leads to the following expression of a joint distribution for each cell.

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

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

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

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 laser range-finder

Here, the Elves and Moravec bayesian telemetric sensor models [1] are used for the 1D-case (eq. (3),(4)). An error model is used, which allows to have a logarithm expression, like in [8]. 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.

$$p(z | [O_\rho = \text{occ}]) = \begin{cases} 1/2^z \text{ if } z < \rho \\ 1/2^{\rho-1} \text{ if } z = \rho \\ 0 \text{ otherwise.} \end{cases} \quad (3)$$

$$p(z | [O_\rho = \text{emp}]) = \begin{cases} 1/2^z \text{ if } z < \rho \\ 0 \text{ if } z = \rho \\ 1/2^{z-1} \text{ otherwise.} \end{cases} \quad (4)$$

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

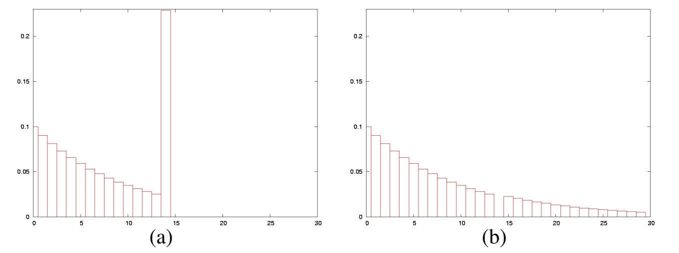
Fig. 2. (a) (resp. (b)) shape of probability distribution over the possible sensor range measurements knowing that the 14th cell is occupied (resp. empty) including an error model and *a priori* over the occupancy of cells.

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 model distributions must be switched from the Z-grid to the V-grid. In the first subsection, we give a general formalisation 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 adaptative 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 subdvision 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. Given a mesh $\mathcal{A}$ and a function f: $F(\mathcal{A}) \rightarrow E$ where $F(\mathcal{A})$ is the set of faces in $\mathcal{A}$ and E a vector space and given a mesh $\mathcal{B}$ such as $\mathcal{B} \subset \mathcal{A}$ (i.e. each point in the surface covered by $\mathcal{A}$ belongs to $\mathcal{B}$ too), we search a function g: $F(\mathcal{B}) \rightarrow E$ such as for each face $r \in F(\mathcal{B})$

$$\int_{t \in r} f(t) ds^3 = g(r).$$

If the function was differentiable and if we had an analytical expression of its derivatives, a gradient analysis would have given exact equations for the change of coordinate system. But in most cases in bayesian modelling, functions are discretized due to learning processes or as the result of bayesian inference. In our special case, we do not posess the analytical form of the sensor model (eq. (3),(4)).

[3]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.

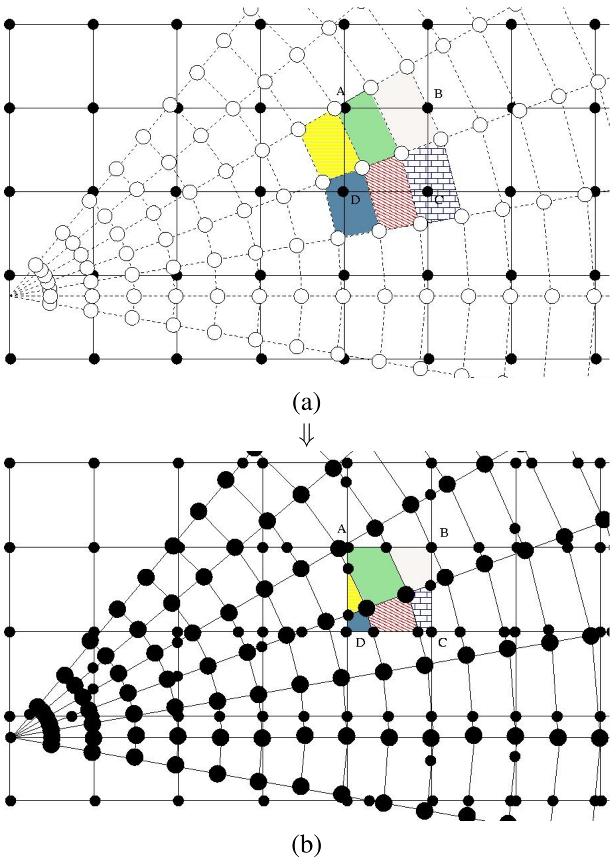

Fig. 3. (a) two subdivisions with dash lines and plain lines. In different color patterns: the different cells in the mesh $\mathcal{A}$ that intersect the ABCD cell of mesh $\mathcal{B}$ *i.e* I_{ABCD}. (b) overlaying the two subdivisions: adding vertex at each intersection of $\mathcal{A}$ and $\mathcal{B}$. 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 3a): 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:

$$g(r) = \sum_{i \in I_r} s_{i,r} f(i). \qquad (5)$$

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 folowing:

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

```
1: mapping ← array($\#(F(\mathcal{B}))$)
2: compute $C(\mathcal{A})$: a convexe approximation of $\mathcal{A}$
3: compute the map overlay of $C(\mathcal{A})$ and $\mathcal{B}$
4: for each face $f$ of the overlay do
5:     find $i \in F(C(\mathcal{A}))$ and $r \in F(\mathcal{B})$ such as $f \subset i \cap r$.
6:     compute $s = \frac{surface(f)}{surface(r)}$
7:     append $(r, s)$ to mapping[$i$].
8: 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 litterature 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 litterature. 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 catesian 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:

$$ns(x, y) = \frac{dx^2}{((\rho + \frac{d\rho}{2})^2 - (\rho - \frac{d\rho}{2})^2)d\theta} = \frac{dx^2}{\rho d\rho d\theta} \quad (6)$$

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

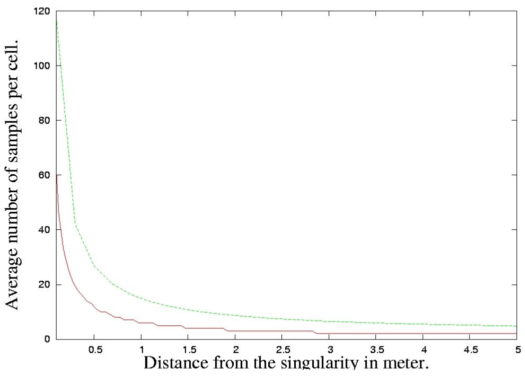

Fig. 4. In red, below: the analytical curve of the number of sample in adaptative sampling given by the ratio between cartesian and polar surface. In green, above: cardinal of the I_r sets in the overlay subdivision provided by the exact algorithm. One can notice that the adaptative sample is an approximation because the curve is below the exact one. The sampling scheme is hyperbolic in the exact and approximate case.

This approach, called adaptative 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 according weight. The average and maximal errors (tab. I) are small compared to the line drawing algorithm; calculus time is, however, more expensive. The adaptative 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. 4) 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[4]). 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 significative, considering the level of uncertainty.
In these three methods, the exacte 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 F that associates each object surface coordinate to each texture coordinate:

$$\begin{array}{cccc} F: & \mathcal{R}^2 & \rightarrow & \mathcal{R}^2 \\ & (x,y) & \mapsto & (u,v) = (\alpha(x,y), \beta(x,y)) \end{array}$$

Let $I_a(x,y)$ 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 intesity by:

$$I_a(x,y) = T_a(u,v) = T_a(\alpha(x,y), \beta(x,y)). \quad (7)$$

The problem statement is how to define on the regular grid of the image representation in computer memory this continuous fonction. This is precisely a sampling problem.
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.

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.

[4]in a slam perspective, for example

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

TABLE I
COMPARISON OF CHANGE OF COORDINATE SYSTEM METHODS.

IV. IMPLEMENTATION ON GPU

A. Change of coordinate system

The processing chain begins by defining in polar coordinates the values of the sensor models for each ray, obtaining two 2D textures, one for each state: occ and emp. Then the topological singularity is handled by computing the two previous textures at several resolutions. Thus when a cell in the cartesian grid corresponds to several cells in the polar grid a coarse resolution of the polar grid is used; this the case close to the polar origin. This process is called mipmapping and is accelerated by graphical boards. Then the change of geometry is processed by drawing the cones in the cartesian grid, each point in the cones being mapped to a texture point in each of the two sensor model textures, which have just been defined.

B. Fusion

The floating precision of GPUs is limited, so to avoid numerical pitfalls, a logarithm fusion is used. 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 marginalisation term disappear and thanks to a logarithm transformation, sums are sufficient for the inference.

$$\log \frac{p(\text{occ}|\overrightarrow{z})}{p(\text{emp}|\overrightarrow{z})} = \log \frac{p(\text{occ})}{p(\text{emp})} + \sum_{i=1}^{n} \log \frac{p(z_i|\text{occ})}{p(z_i|\text{emp})} \quad (8)$$

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 using the transparency methodology. The occupancy grid for each sensor appears as a layer where transparency decreases as the occupancy of the cell increases.

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 5.

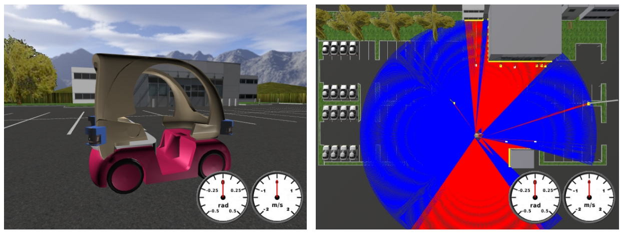

Fig. 5. Fusion view for 4 Sicks LMS-291.

A. Precision

The obtained precision is close to the exact solution, not as close as with the adaptative sampling method but far better than with the state-of-the-art method. Details close to the vehicule 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 saftey, 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 senosrs 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. 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 futur works, we plan to explore the question of 3D OG modelling using graphical hardwares. 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.