# 3D mapping of outdoor environment using clustering techniques

Manuel Yguel* and Olivier Aycard†
*University of Karlsruhe - GERMANY
Email: manuel.yguel@gmail.com
†University of Grenoble1 - FRANCE
Email: Olivier.Aycard@imag.fr

***Abstract*— The goal of mapping is to build a map of the environment using raw data provided by some sensors embedded on an intelligent vehicle. This map is used by an intelligent vehicle to have knowledge about its surrounding environment to better plan its future actions.**

**In this paper, we present a method, based on occupancy grids [3], to map 3D environment. In this method, we discretize the environment in cells and the shape of each cell is approximated by one or several gaussians in order to achieve a balance between representational complexity and accuracy. Experimental results on 3D real outdoor data provided by a lidar are shown: a map of an urban environment is presented. Moreover a quantitative comparison of our method with state of the art methods is presented to show the interest of the method.**

Keywords: Sensor Data Processing, Data Modeling, Learning, Perception

## I. Introduction

An intelligent vehicle is generally defined as a vehicle designed to help driving automatically or to monitor a human driver and assist him in driving. To solve these tasks, they are equipped with sensors to perceive their surrounding environment and with actuators to act in this environment. To make the link between sensors and actuators, an intelligent vehicule generally requires 3 fundamental components: i) perceive and model the environnement where it is moving, ii) reason and decide about future actions to execute iii) and finally perform these actions. Perception plays a fundamental role in construction of an intelligent vehicle as it constitutes its first component and provides informations to other components. Its objective is to interpret noisy and raw data of different sensors embedded on a vehicle to model environment (ie, build a map of the environment) and understand the current situation in order to provide necessary informations to decide future actions to execute. The quality of perception processing has an impact on the quality of the whole process.

The choice of map representation is an important step when dealing with the perception problem. Popular methods for representing maps of the environment include: feature-based approach [5], grid-based approach [3] and direct approach [7]. Because of advantages over others, nowadays, occupancy grids have become the most common choice among map representation methods, particular for applications in outdoor environments [8]: whereas grid-based approaches typically require an important amount of memory, but they are able to represent arbitrary features and provide detailed representations. In this representation, the environment is subdivided into a regular array or a grid of rectangular cells. In addition to this discretization of space, a probabilistic measure of occupancy is estimated for each cell of the grid which indicate that the cell is occupied by an obstacle or not given the raw data provided by the sensors. An example of an occupancy grid map representation, built using 2D raw laser data is shown in figure 1, where white regions correspond to free cells, black regions to occupied cells and grey regions to unknown cells.

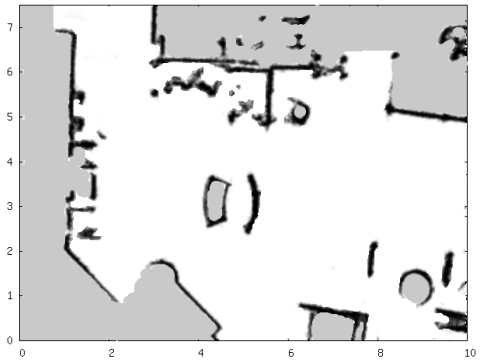


Fig. 1. Example of grid map representation.

The resolution of the environment representation directly depends on the size of the cells. The idea of producing multi-resolution grids has been present since the very first works on grid-based representations [3]. Coarse grids are used in path planning [9] or localization [10] algorithms in order to obtain a rough trajectory or position estimate at a low computational cost. Then, at a second stage, this estimate is used to initialize the fine resolution search algorithms, thus accelerating convergence. However, as the resolution becomes coarser, the aliasing effect of the geometry of the cells becomes more evident and it can no longer be neglected. When considering a cell as a full block, all the information concerning the shape of the cell contents is lost. A way to alleviate this problem is to attach some sort of statistical shape description to every occupied cell. [1] significantly improves accuracy by approximating the shape of the cell contents with ellipsoids. But a single ellipsoid is still a poor representation when there are several objects with different

orientations in the cell, which is the case –for instance– of a pole standing on the ground (see fig. 2).

In this paper, we present a method to overcome this problem by allowing a coarse resolution cell to contain multiple ellipsoids – more specifically, Gaussian clusters. The idea is to start with a single cluster per cell, and then to *refine* it by inserting additional clusters in order to achieve a balance between representational complexity and accuracy.

In next section, we present our new map representation and our approach to overcome limitations of classical methods: multi-resolution gaussian maps. In section III, we explain how to update a multi-scale gaussian map given a set of raw data. Section IV presents our refinement algorithm to improve the quality of the representation at coarse levels. This step is our main contribution, at every refinement step new Gaussian clusters are added to the cells where the representation error is maximum. Experimental results on real 3D laserscanner raw data are reported in section V. Section VI gives some conclusion and perspectives to this work.

## II. Map Representation & Approach Overview

In this paper, we are concerned with mapping of 3D static outdoor environment. The perception of the environment is done by a 3D laserscanner embedded on an intelligent vehicle moving in this 3D static outdoor environment. The laserscanner perceives, with a frequence of about 20 hertz, a part of the environment depending on its field of view and the position of the intelligent vehicle. A perception of the laserscanner is composed of a set of 3D points: each 3D point (x, y, z) corresponds to a static part of the environment.

The size of the environment (and the corresponding grid as well) is given *apriori* and each time the 3D laserscanner is performing an acquisition of a set of 3D raw data, we update our multi-resolution gaussian map. Actually, it is composed of 3 independent grids with different resolutions. The size of cell at each resolution is given *apriori* and the number of cells for each resolution depends on these resolutions. Our approach processes every resolution independently. Each resolution is a sparse grid where only occupied cells are stored. A sparse grid is encoded using a special hash table: each occupied cell is indexed by a key built upon its integer coordinates inside the grid. Basically, the grid is seen as a 3-D array, and like for a matrix, an unique integer can be associated to a cell $(i, j, k)$:

$$\text{key} = k + w * j + h * w * i$$

where $w$ and $h$ are the number of cells in the width and the height of the grid. This supposes that the grid is contained inside a bounding box with dimensions $(hs, ws, ds)$ where $d$ is the number of cells in the depth of the grid and $s$ is the cell size in meters. Thus for each real point, one compute the 3-D coordinates of its cell, then the key of the cell in the hash table.

A cell is represented by one or more gaussian: in the fine resolution there is only one gaussian per cell and in coarser resolutions, a cell can contain several gaussians.

Figure 2 gives an illustration of our map representation for a 3D simulated scene with only one gaussian per cell.

As shown in Fig. 3, the approach is composed of three main components:

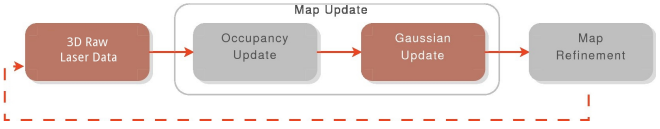


Fig. 3. Framework components.

.

1) 3D laser raw data acquisition: this is the first step of the process. It consists of collecting 3D laser raw data used by the next parts of the process;
2) Map Update: this step is composed of two parts:
   - Occupancy Update: it is the likelihood that a cluster represents a static part of the map. In our framework it allows to adapt faster the regions that have not been observed often;
   - Gaussian Update: it adapts existing clusters in order to minimize the representation error.
3) Map refinement: this step is our main contribution, at every refinement step new Gaussian clusters are added to the cells where the representation error is maximum. In particular, from now on, we will assume that there is a given budget of Gaussians per resolution (except the fine one) that needs to be allocated in an optimal way through a refinement process. It is worth noting that no refinement is performed on the finest resolution.

Light gray boxes are processed less than once per acquisition of 3D laserscanner raw data. These part of the process usually takes time to proceed and moreover there is no need to process them at each time. In our framework, these light gray boxes are processed after a given number of acquistion. This given number is defined *apriori* at the beginning of the process.

## III. Map Update

This section presents the procedure to update an existing map from 3D laser raw sensor data. At this point, we assume that the number $k$ of Gaussian clusters per cell is known. The actual estimation of $k$ is handled by the refinement algorithm that we will discuss in section IV.

Our goal here is to update the Gaussians' mean value $\mu$ and covariance $\Sigma$ in order to minimize the representation error with respect to the input data. Every observation is used to incrementally update the different resolutions independently. The basic idea is to find the cell where the input point falls and then updating the cluster in that cell that is "closest" to the input point.

As in most incremental approaches, an important question is how much to adapt the clusters – *i.e.* finding the 'right' learning rate. In the following subsection, we describe the use of the cluster's occupancy to control the adaptation. It can be intuitively explained as follows: the more a cluster has been observed, the more is known about it and the less

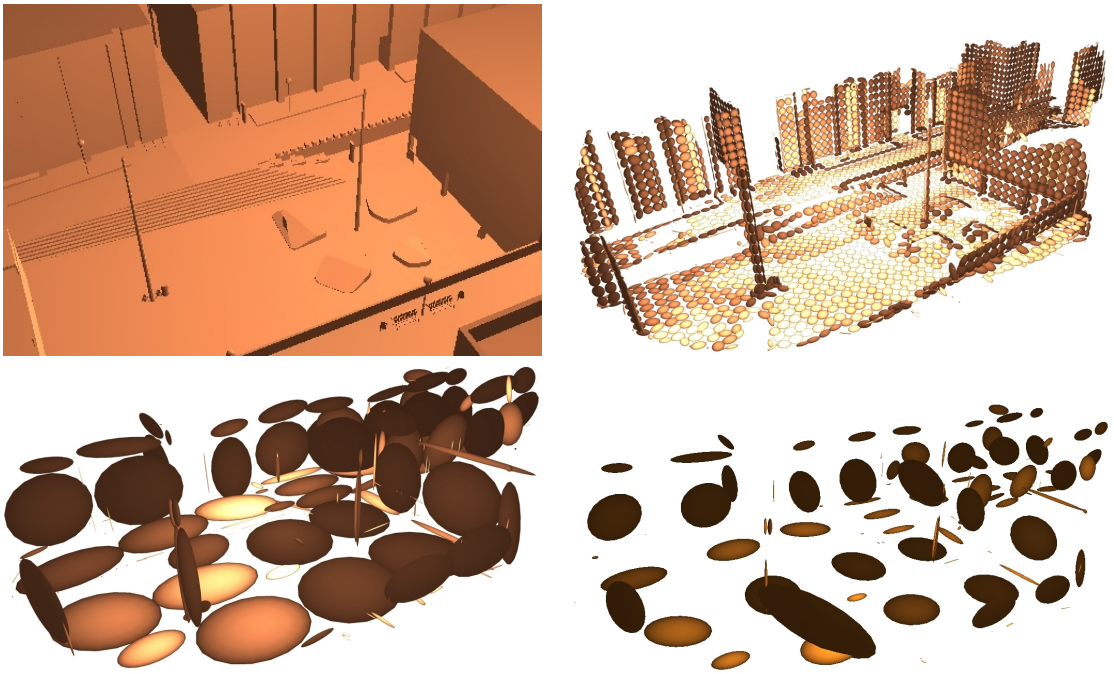

Fig. 2. Top left: 3D simulated scene. Top right: fine resolution of the refined map. Bottom left: intermediate resolution of the refined map. Bottom right: coarse resolution of the refined map

reasonable it is to modify it. So, this adaptation depends on the occupancy of a cluster. In next subsection, we describe how this occupancy is computed for a point and a cluster. Furthermore if the cluster violate visibility, i.e. one sees through that cluster in some observation, this cluster is very likely to have represented a moving obstacle (a parked car for instance). Thus we want to decrease its occupancy in such a case.

### A. Computing Cluster Occupancy

We can split that problem in two steps: the increment of occupancy for observed clusters and the decrement of occupancy for clusters that have been seen through. The increment of occupancy is straightforward and is done during the update of the cells that contain an observation. For the second step, we use the 3-D Bresenham algorithm [4] that traverses all the cells that are seen through and decreases the occupancy of the clusters they may contain.

### B. Updating Gaussian Clusters from Data

For every observation, a single cluster per resolution will be updated. The cluster is selected by finding the cell that contains the observation and then finding the cluster having the minimum distance to that point.

Once the cluster has been selected, its parameters are updated by means of a stochastic gradient descent algorithm. The principle is to update the cluster by a fraction of the negative gradient with respect to each observation. As more and more samples are processed, the magnitude of the adaptation should decrease to ensure convergence. A good example is the on-line computation of the sample mean:

$$\boldsymbol{\mu}^n = \boldsymbol{\mu}^{n-1} + \tfrac{1}{n}(\mathbf{z}^n - \boldsymbol{\mu}^{n-1})$$

where $n$ represents the number of samples processed so far, and $\mathbf{z}^n - \boldsymbol{\mu}^{n-1}$ can be understood as the negative gradient, and $\frac{1}{n}$ the fraction of the gradient to be taken into account. This decreasing weight is called the learning rate and is noted $\epsilon$. In our approach, the value of $\epsilon$ depends on the occupancy, as described in section III-C.

In the case of points, a distance metric between a point and a Gaussian should be used. We have chosen to use the probability measure given by (1):

$$d(\mathbf{p},\mathbf{w}) \triangleq \frac{1}{2}[(\mathbf{p}-\boldsymbol{\mu}_\mathbf{w})^T {\boldsymbol{\Sigma}_\mathbf{w}}^{-1}(\mathbf{p}-\boldsymbol{\mu}_\mathbf{w}) + \log\left(\det(\boldsymbol{\Sigma}_\mathbf{w})\right)] \ , \tag{1}$$

This distance is the addition of the Mahalanobis distance and a volume term. Compared to the pure Mahalanobis distance, the volume term aims at compensating the fact that the Mahalanobis distance of a big cluster tends to make every point very close.

### C. Learning Rate

Our idea is to define the learning rate from the occupancy: the higher the occupancy of a cluster, the better the accuracy of its position and shape is supposed to be; thus, a small value of $\epsilon$ should be used. If, on the other hand, the occupancy is low, the current estimated state of the reference vector can be assumed to be based on insufficient statistics and the learning rate should be high to permit the reference vector to adapt itself.

In log-ratio the occupancy typically is bounded in $[-o_{\max}, o_{\max}]$ and the learning rate varies within

$[\epsilon_{\min}, \epsilon_{\max}]$. For our approach we have chosen a linear mapping between both values:

$$\epsilon(o) = \frac{\epsilon_{\min} - \epsilon_{\max}}{2o_{\max}}(o + o_{\max}) + \epsilon_{\max} \ . \tag{2}$$

## IV. Map Refinement

The idea is to start with a single gaussian per cell, and then to *refine* it by inserting additional gaussian clusters in order to achieve a balance between representational complexity and accuracy. This refinement is only performed for intermediate and coarse resolution. We aim at adding clusters only in those regions where the Gaussian shapes have already converged to their final shapes, which can be deduced from its occupancy. Accordingly, we choose to refine a cell $c$ only if its occupancy probability is above $0.5$. In particular, from now on, we will assume that there is a given budget of Gaussians per resolution that needs to be allocated in an optimal way through a refinement process. The refinement process is driven by a measure of the representation error. The map is periodically refined by inserting a new cluster in the cell that has the maximum error. After every insertion, the shape of the other clusters in the same cell should be modified accordingly; this is done by running a clustering algorithm using the cells of the finer resolution as input.

The following subsections provide the details of the refinement algorithm: the error metric used to find where to add a new gaussian cluster is introduced in next subsection and secondly we present the clustering algorithm.

### A. Error Computation

To find the cell to refine, we compute an error value per cell for the intermediate and coarse resolution. For a given cell, this value is basically the sum of errors between each coarse or intermediate gaussian and the corresponding fine gaussians. So for a given cell, the value is the sum of errors of each coarse or intermediate gaussian. More formally, this value is the sum of the Mahalanobis distance between the center of each gaussian cluster and the Gaussian clusters of the finer resolution.

For the cells $c$ of the coarse or intermediate resolution, having reference gaussian clusters $\{\mathbf{w}_1, \ldots, \mathbf{w}_K\}$ and the finest corresponding gaussians: $\{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$, we compute the average distance $\mathcal{E}(c)$ of each gaussian to its closest gaussian reference at the finest resolution:

$$\begin{aligned}\mathcal{E}(c) = \frac{1}{N}\sum_{i=1}^{k}\sum_{j=1}^{N}(1 - \epsilon_{\mathbf{z}_j})\delta(\mathbf{w}_i, \mathbf{z}_j) \\ [(\boldsymbol{\mu}_{\mathbf{w}_i} - \boldsymbol{\mu}_{\mathbf{z}_j})^T \boldsymbol{\Sigma}_{\mathbf{z}_j}^{-1}(\boldsymbol{\mu}_{\mathbf{w}_i} - \boldsymbol{\mu}_{\mathbf{z}_j})]\end{aligned} \tag{3}$$

where $\delta(\mathbf{w}_i, \mathbf{z}_j)$ is one if $\mathbf{w}_i$ is the closest reference vector to $\mathbf{z}_j$ using the Mahalanobis distance defined by $\mathbf{z}_j$ and zero otherwise. The occupancy is used through the learning rate $\epsilon_{\mathbf{z}_j}$ to assign higher error weights to occupied clusters, disregarding those whose occupancy is low and, in consequence, whose accuracy may still improve without the need of adding extra clusters.

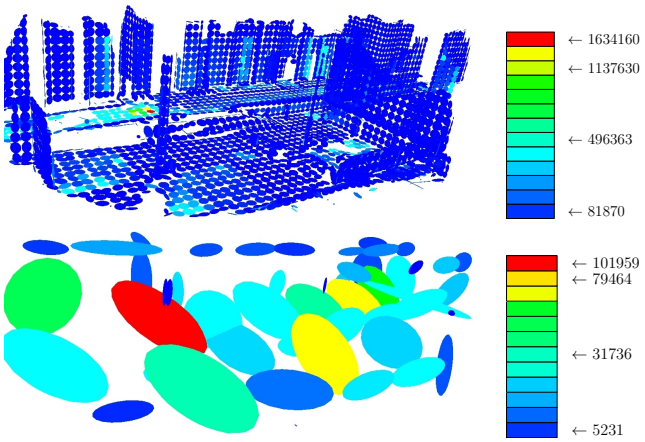


Fig. 4. Up: fine scale is colored with the magnitude of the contribution to the error at the coarser scale. Down: coarse scale, mean error. Error palettes are on the right.

Figure 4 illustrates the error computed for each cell of the coarse resolution at the beginning of the refinement process where at the coarse resolution, we only have one gaussian per cell. The top of the figure represents the contribution of each cell of the fine resolution to the error of the corresponding coarse cell. Most of cells at fine resolution have a small contribution to the error: because at this level, the gaussian is usually a good approximation of corresponding data. The most important error (shown by yellow and red colors) is due to the stairs which are not very well modelized by gaussians. At the coarse level, we see that using only one gaussian per cell gives very poor approximation. For instance, the gaussian in red is a very poor approximation of the real data (see top left figure 2): a part of stairs and a part of one pole and one background wall. In this case, the first refinement process will add one gaussian in this cell.

### B. Clustering for Map Refinement

In this section, we describe our clustering approach for map refinement. This method solves a hard clustering problem: we are looking for a partition $C^* = \{C_1^*, \ldots, C_k^*\}$ of the k gaussian clusters of a given cell $c$ into $k$ classes represented by $k$ reference vectors $\{\mathbf{w}_1, \ldots, \mathbf{w}_K\}$.

This is done by using the well known k-means clustering algorithm [6]. The optimal clusters are computed iteratively from the set of reference vectors : each datum is associated to its closest reference vector; then, the minimizer of each cluster energy is computed.

In the next two paragraphs, we explain the distance we choose for our specific problem and we detail how we deal with the problem of initialization and local minima.

*1) K-Means Extension for Gaussian Inputs:* In the basic Lloyd algorithm, both input data and reference vectors are simply points (3-D points in our case) and the distance function is the square Euclidean distance. In order for the covariance matrices of the clusters at the coarse or intermediate resolution to be as accurate as possible, we need to use the information provided by the covariance matrices at the finest resolution. Therefore, we need a clustering algorithm

that is able to properly handle Gaussian input data. [2] has proposed such an algorithm, proving that it converges. The algorithm uses the Kullback-Leibler divergence (Eq. 4) as a distance function for Gaussians:

$$D_{KL}(\mathbf{z}\|\mathbf{w}) = \frac{1}{2}[(\boldsymbol{\mu}_\mathbf{z} - \boldsymbol{\mu}_\mathbf{w})^T \boldsymbol{\Sigma}_\mathbf{w}^{-1}(\boldsymbol{\mu}_\mathbf{z} - \boldsymbol{\mu}_\mathbf{w}) + \log\left(\frac{\det \boldsymbol{\Sigma}_\mathbf{w}}{\det \boldsymbol{\Sigma}_\mathbf{z}}\right) + \mathrm{Tr}\left(\boldsymbol{\Sigma}_\mathbf{z}\boldsymbol{\Sigma}_\mathbf{w}^{-1}\right) - d]$$

where $d$ is the dimension. The metric is composed of three terms corresponding to the Mahalanobis distance, the volume and the orientation, respectively.

The use of this metric in clustering means that the decision of grouping clusters together does not only depend on their positions, but also on their sizes and orientations. This property is particularly important for mapping, since it will tend to preserve sharp features such as corners and edges because the distance between the linear components of such features will increase with the angle difference between them.

As explained in [2] the computation of the optimal reference vectors from a set of Gaussians $\{\mathbf{z}_j = (\mu_{\mathbf{z}_j}, \Sigma_{\mathbf{z}_j}) | j = 1, \ldots\}$ weighted by positive reals ($\alpha_{\mathbf{z}_j}$), is done in two steps. First the optimal mean is computed using (4):

$$\boldsymbol{\mu}^* = \frac{1}{\sum_j \alpha_{\mathbf{z}_j}} \sum_j \alpha_{\mathbf{z}_j} \boldsymbol{\mu}_{\mathbf{z_j}} \ , \tag{4}$$

then the covariance matrix is given by (5):

$$\boldsymbol{\Sigma}^* = \left[\frac{1}{\sum_j \alpha_{\mathbf{z}_j}} \sum_j \alpha_{\mathbf{z}_j} (\boldsymbol{\Sigma}_{\mathbf{z_j}} + \boldsymbol{\mu}_{\mathbf{z_j}}{}^T \boldsymbol{\mu}_{\mathbf{z_j}})\right] - (\boldsymbol{\mu}^*)^T \boldsymbol{\mu}^* \ , \tag{5}$$

where the $\alpha_j$ are the occupancy associated with each measurement.

*2) Initialization of K-Means:* An important drawback of k-means is that it is highly dependent on initialization. Moreover, even if the algorithm is guaranteed to converge, it often gets stuck in local minima.

To get out of the local minima a so called "swap" procedure is used. One cluster is chosen, either randomly or because of its short distance to a different cluster with more elements. Then, a simulation is done by reallocating that reference vector to the region of space with maximum error. If the resulting partition has a lower clustering distortion, the result is accepted and a new simulation is done. Otherwise, the result is rejected and the procedure stops.

## V. Experimental Results

### A. Data and Demonstrator

To test our approach, we had a cooperation with an industrial partner to perform 3D Mapping of outdoor environment using a laser scanner. We had a dataset of 6 Millions of laser data collected with a vehicle moving moving in this urban environment. The localization of the vehicle was provided. A camera was also mounted on the vehicle and laser data were synchronized with images. So, each laser data had 6 dimensions: x, y and z (for position) and RGB component for color.

The problem was to propose a multi-resolution model of this 6D dataset with high precision and significant reduction of size. We build a multi-resolution gaussian map of 700 meters $\times$ 200 meters $\times$ 50 meters. This map was composed of 3 resolutions: cells have a size of 20 centimeters for fine resolution, 3.2 meters for intermediate resolution and 12.8 meters for coarse resolutions.

### B. Experimental Results

With this dataset, we have experimented the use of the color as an extra clustering dimension. In this case, the points are gathered into a same cluster if they are close in terms of location (ie, they produce the smallest possible ellipsoid in the 3D space) and in terms of color (they make the smallest ellipsoid in the uv color space). Figures V-B presents our results at the fine resolution. With our approach, we were able to obtain a reduce representation of less than 0.1% of the original data but nonetheless accurate. Moreover, we can see in figure V-B that a significative distinction is made between the object in the environment based upon the color. In particular vegetation is well separated from artificial structure.

Figure 6 shows the coarse resolution with one gaussian per cell. This figure is used as a reference to compare it with our approach. We clearly see that an approximation of the shape by one gaussian is not appropriate: for instance, all the cells along the road approximate the road and the buildings as well. So, their shape and color is a mix between the shape and color of these two independent parts of the environment.

On the contrary, figure 7 provides a very good approximation of the shape of the objects present inside the environment. We can see that we have a very good separation between road and buildings, moreover the color of the gaussian corresponds to the color of the corresponding object. The coarse resolution with one Gaussian per cell contains 5457 Gaussians and the number of Gaussians in the refined coarse resolution is 42367. We added less than 8 Gaussians per cell in average and were able to obtain a significant gain, in terms of separation between objects. The large planar shapes of the road are very well approximated by large Gaussians whereas complicated shapes of the facades of the buildings and the vegetation use more Gaussian. For path planing, for collision detection, this representation allows to compute bounding ellipsoids

## VI. Conclusions and Future Work

We have proposed a comprehensive framework to build three-dimensional maps from range data. The proposed representation enhances the accuracy of previous approaches by enabling the presence of several Gaussians per cell. These Gaussians are added by means of a refinement algorithm which inserts them where the representation error is maximum. The algorithm makes use of a recent Gaussian clustering approach that uses the Kullback-Leibler divergence as a distance function, thanks to this, our algorithm is

Fig. 5. fine resolution of 3D laser data and color

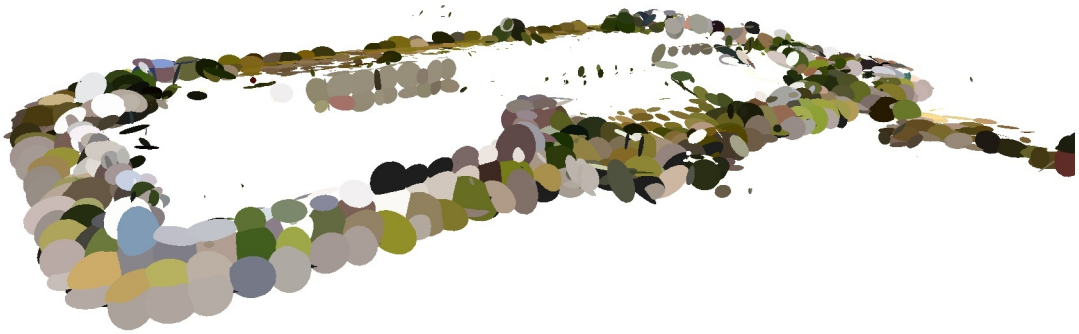

Fig. 6. coarse resolution with one Gaussian per cell

Fig. 7. coarse resolution with several Gaussians per cell

able to preserve important features of the environment (*e.g.* corners) that are usually smoothed out by other approaches. Experiments with real data show that, for coarse resolutions, significant accuracy gains may be obtaining by a small augmentation in the number of clusters. Moreover, when compared with existing approaches, the additional computational cost that is required to insert these clusters is marginal.

Further work includes working towards real time mapping of huge streams of 3-D points by exploiting parallelization and hierarchical multi-scale update.

## REFERENCES


[1] Peter Biber and Wolfgang Straßer. The normal distributions transform: A new approach to laser scan matching. In *IEEE/RJS International Conference on Intelligent Robots and Systems*, 2003.

[2] Jason V. Davis and Inderjit Dhillon. Differential entropic clustering of multivariate gaussians. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 337–344. MIT Press, Cambridge, MA, 2007.

[3] A. Elfes. *Occupancy grids: a probabilistic framework for robot percpetion and navigation*. PhD thesis, Carnegie Mellon University, 1989.

[4] J. Foley, A. VanDam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley, 2nd. ed. edition, 1990.

[5] John Leonard and Hugh Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1991.

[6] S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, Mar 1982.

[7] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 1997.

[8] M. Montemerlo and all. Junior: The stanford entry in the urban challenge. *Journal of Field Robotics*, 2008.

[9] D. K. Pai and L.-M. Reissell. Multiresolution rough terrain motion planning. In *IEEE Transactions on Robotics and Automation*, volume 1, pages 19–33, February 1998.

[10] Alex Yahja, Anthony (Tony) Stentz, Sanjiv Singh, and Barry Brummit. Framed-quadtree path planning for mobile robots operating in sparse environments. In *In Proceedings, IEEE Conference on Robotics and Automation, (ICRA)*, Leuven, Belgium, May 1998.