Error Driven Refinement of Multi-scale Gaussian Maps

Application to 3-D Multi-scale map building, compression and merging

Manuel Yguel, Dizan Vasquez, Olivier Aycard, Roland Siegwart, Christian Laugier

Abstract The accuracy of Grid-based maps can be enhanced by combining them with a Gaussian representation, putting a Gaussian in every cell of the map. However, this solution works poorly for coarse discretizations in multi-scale maps. This paper proposes a method to overcome the problem by allowing several Gaussians per cell at coarse scales. We introduce a multi-scale approach to compute an error measure for each scale with respect to the finer one. This measure constitutes the basis of an incremental refinement algorithm where the error is used to select the cells where the number of Gaussians should be augmented. As a result, the accuracy of the map can be selectively enhanced making an efficient usage of computational resources. Moreover, the error measure can also be applied to compress a map by deleting the finer scale clusters when the error in the coarse ones is low.
The approach is based on a recent clustering algorithm that models input data as Gaussians as opposed to points, for conventional algorithms. In addition to mapping this clustering paradigm permits to perform map merging and to represent feature hierarchies under a sound theoretical framework. Our approach has been validated with both real and simulated 3-D data.

1 INTRODUCTION

The idea of producing multi-scale grids has been present since the very first works on grid-based representations, [1]. Coarse maps are used in path planning [2, 3] or localization [4] algorithms in order to obtain a rough trajectory or position estimate

Yguel, Laugier
INRIA Rhône-Alpes, Grenoble e-mail: firstname.lastname@inrialpes.fr

Vasquez, Siegwart
ETHZ, Zürich e-mail: name@mavt.ethz.ch

Aycard,
UJF, Grenoble e-mail: firstname.lastname@inrialpes.fr

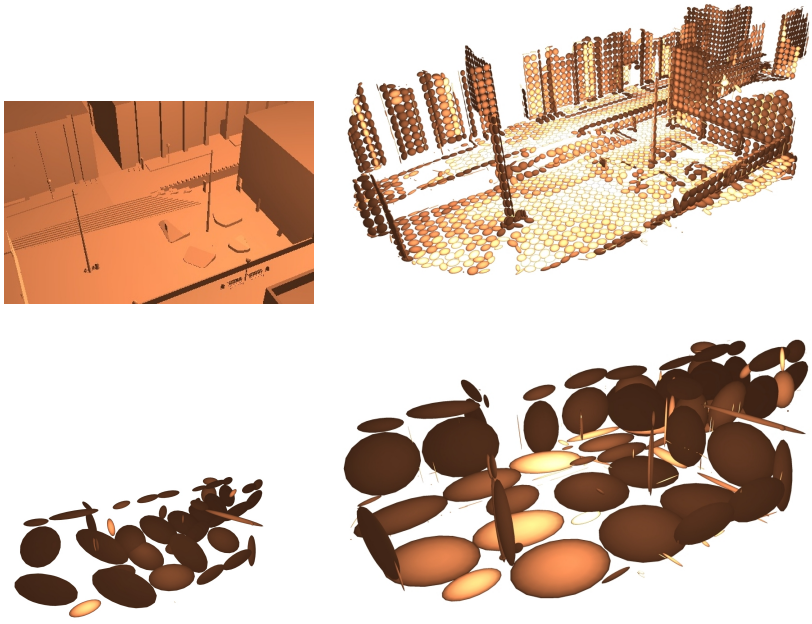

Fig. 1 Up,left: simulated scene. Up, right: fine scale of the map. Bottom, left: coarse scale of the map, one Gaussian per cell. Bottom, right: coarse scale of the refined map

at a low computational cost. Then, in a second stage, this estimate is used to initialize the fine scale search algorithms, thus accelerating convergence. In the case of localization, this procedure also enlarges the convergence region of most of the algorithms. Examples of algorithms that benefit of such an approach include sampling based, gradient based and line-to-point or plane-to-point ICP based algorithms.

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. Moreover, all information concerning the shape of the cell contents is lost since coarse resolution cells are, in general mostly empty. A way to alleviate this problems consists in attaching to every occupied cell some sort of statistical shape description. Two seminal works in this direction are tensor maps [5] and the Normal Distribution Approach (NDT) [6], these approaches significantly improve accuracy by approximating the cell contents' shape with ellipsoids that are encoded as symmetric semi-definite positive (SSDP) matrices. The accuracy of these approaches, together with their relative simplicity has contributed to making them very popular in the map building community [7, 4].

That being said, a single ellipsoid is still a very 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. In this paper, we present a method to

overcome this problem by allowing a coarse resolution cell to contain multiple ellipsoids – more specifically, Gaussians. 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 representation complexity and accuracy. In particular, from now on, we will assume that there is a given budget of Gaussians per coarse scale that needs to be allocated in an optimal way by a refinement process.

In the following section, we review related works in robotics and computer graphics. Then, we explain how Gaussian maps may be extended to handle additional clusters per cell, extending the concept to a theoretically sound formulation of map merging. In section 5, we explain how to update a Gaussian map using either points or Gaussians as input. In section 6, we explain our use of occupancy as a learning rate to update the map. Section 7, presents our error driven refinement algorithm for coarse scales. Section 8 discusses mapping results on simulated and real data sets. Finally, we present our conclusions as well as possible directions for future work.

2 RELATED WORK

2.1 Related mapping approaches in robotics

An interesting approach to grid refinement are multi-level surface maps (MLS) [8], which can be considered as a refinement of an elevation map. A MLS map is a 2-D grid where, for each cell, several planes are stored at different heights, together with their associated thickness. Their structure make them particularly well suited to represent traversability information, as shown by their impressive results on real data sets. However, they share the aliasing related problems of 2-D grids particularly in the horizontal plane. Moreover, due to their lack of merging mechanisms they often fail to represent tall vertical structures as a single element if those structures were partially obstructed during early observations.

A different approach to cope with cell aliasing is to use a multi-scale grid map which is refined where features are denser. Tree-based representations, such as quadtrees and octrees are the most popular data structures of this kind for two and three dimensional input spaces, respectively [9, 10]. Nevertheless, these structures also suffer from the aliasing problem because of their cubic cell shape, which makes them inappropriate to represent curves or surfaces.

Tensor voting and NDT aim to improve geometric accuracy by representing all the points that fall into a given cell by an ellipsoid, whose orientation and shape are such that the representation error is minimized. In both cases, the ellipsoid is encoded as an SSDP matrix (Fig. 3).

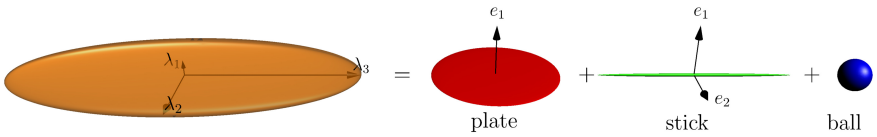

Fig. 2 Decomposition of an SSDP matrix ellipsoid into elementary shapes called tensors in [5].

In both tensor voting and NDT, having one cluster per cell produces large ellipsoids (called junctions in the tensor voting framework) as soon as the resolution is too coarse and the cell encompasses several distinct objects or the object geometry is not linear. This problem has been addressed in the original NDT paper [6] by storing four overlapping maps shifted by half a cell. However this approach is expensive in terms of number of Gaussians and the advantages are unclear when compared to a map with twice the resolution.

2.2 Related approaches in computer graphics

In the computer graphics community the problem of 3-D model simplification has received a lot of attention. The objective in this case is to obtain simpler models to streamline manipulation and speed rendering when high accuracy is not required – *e.g.* when objects are far from the virtual camera or moving rapidly. This is deeply related to refinement as it is, essentially, the inverse problem.
The seminal work in this field is the paper of Garland *et al.* [11] where edge contraction is performed on the mesh edges that introduce the smallest amount of *quadric error* in the model. As indicated by its name , this metric is computed by computing a special type of quadric on the vertices, described by SSPD matrices. The simplification algorithm uses a priority queue. At every iteration, the edge having the lowest error is contracted and the error of all the affected edges is recomputed and reinserted in the queue. The process continues until the target budget of edges is reached.
The second class of effective simplification approaches is based on clustering. They can operate either on meshes or on point clouds. Our approach is closest in spirit to the work in [12], where clustering is performed on the triangles of the mesh to be simplified. As set of triangles is represented by the plane (called shape proxies) that fit them the best. Then the models are simplified by keeping vertices that belongs to proxy intersections and averaging their position between neighbour proxies. An essential component of these works is a shape metric that allows to assign each triangle to its closest cluster and to compute the parameters of the shape proxy. Cohen and Steiner[12] consider two metrics:Garland's quadric error metric [11] and the Euclidean distance between the normals of the triangles and the plane proxy. Re-

garding point clouds, in [13] several agglomerative and partitional approaches are tested.

Our approach is largely inspired by the work of Cohen and Steiner [12] and by the region splitting algorithm of Pauly [13]. The main difference lies in the fact that we do not restrict our main representation to surfaces because at coarse scales many important features such as poles, trees trunks and towers may appear as one dimensional curves rather than surfaces.

3 APPROACH OVERVIEW

Fig.

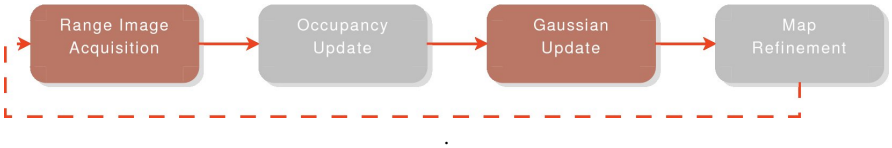

Fig. 3 Framework components. Light gray boxes are processed less than once per range image.

4 CLUSTERING FOR MAP REFINEMENT

In this section, we explain the process of refining the map by adding new clusters. Let c^s be a cell at scale s that contains $(k-1)$ Gaussian clusters and let $\phi(c^s)$ be the set of all the non-empty cells at the finer scale $(s-1)$ that are included into c^s.

If a Gaussian z at the finer scale belongs to a cell c_{z}^{s-1} of $\phi(c^s)$, we say that $\mathsf{z} \in G(\phi(c^s))$.

In the following discussion we assume that, given a set C_i with N elements of the finest scale which belong to the data space $\mathscr{Z}$, we know how to compute the reference vector w^* from the feature space, $\mathbb{F}$, that minimizes the cluster distortion $\mathscr{E}_{C_i}$:

$$\mathsf{w}^* \triangleq \arg\min_{\mathsf{w}\in\mathbb{F}} \mathscr{E}_{C_i}(\mathsf{w}) \triangleq \arg\min_{\mathsf{w}\in\mathbb{F}} \sum_{j=1}^{N} d_{\mathscr{Z}\times\mathbb{F}}(\mathsf{z}_j, \mathsf{w}) \tag{1}$$

where $d_{\mathscr{Z}\times\mathbb{F}}(\mathsf{z}_j, \mathsf{w})$ measures the distance between the element z_j and the closest reference vector. The distortion can be understood as a measure of the error introduced by replacing z_j with the closest reference vector w.

4.1 k-means clustering

We use a hard clustering approach, that is we search a partition $C^* = \{C_1^*, \ldots, C_k^*\}$ of the $G(\phi(c^s))$ into k classes represented by k reference vectors that minimizes the clustering distortion:

$$\mathscr{E}_{(C^*,\{\mathsf{w}_1^*,\ldots,\mathsf{w}_k^*\})} = \underset{\{C_1,\ldots,C_k\},\{\mathsf{w}_1,\ldots,\mathsf{w}_k\}}{\arg\min} \sum_{i=1}^{k} \mathscr{E}_{C_i}(\mathsf{w}_i) \tag{2}$$

This is done using the well known k-means clustering algorithm [14].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 basic Lloyd algorithm, both input data and reference vectors are simply points in feature space (3-D space in our case) $\mathscr{Z} = \mathbb{F} = \mathbb{R}^3$ and the distance function, $d_{\mathscr{Z}\times\mathbb{F}}(\mathsf{z},\mathsf{w}) = \|\mathsf{w}_i - \mathsf{z}\|^2$ is the square Euclidean distance.

An important drawback of k-means is that it is very sensible to initialization.

K-means clustering always converges but unfortunately it is only guaranteed that the solution is a local minimum of $\mathscr{E}_{(C^*,\{\mathsf{w}_1^*,\ldots,\mathsf{w}_k^*\})}$. To get out of the local minima a so called "swap" procedure is used (line 8 to 25, alg. 1). One cluster is choosen, 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 point of 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. A reference vector metric is used to evaluate the similarity between clusters: $d_{\mathbb{F}} : (\mathbb{F}\times\mathbb{F}) \to \mathbb{R}^+$. If $\mathscr{Z} = \mathbb{F}$ it is possible to use $d_{\mathbb{F}} = d_{\mathscr{Z}\times\mathbb{F}}$, to compute both the distance between clusters and the distortion between a datum and its corresponding cluster (line 17, alg. 1).

It is worth noting that this clustering framework naturally defines a hierachy: a cluster is the parent of all the clusters of the finer scale that are closer to it than to any other cluster.

4.2 Use of the Kullback-Leibler divergence

In order for the covariance matrices of the clusters at the finest scale to be as accurate as possible, we need a clustering algorithm that is able to properly handle Gaussian input data $\mathscr{Z} = \mathbb{F} = \mathscr{G}^3 \triangleq \{(\mu,\Sigma) | \mu \in \mathbb{R}^3 \text{ and } \Sigma \text{ is SDP}\}$. Davis [15] has proposed such an algorithm, proving that it converges. The algorithm uses the Kullback-Leibler divergence (Eq. 3) as a distance function $d_{\mathscr{Z}\times\mathbb{F}}$ for Gaussians:

$$D_{KL}(\mathsf{z}\|\mathsf{w}) = \frac{1}{2}\left[(\mu_\mathsf{z}-\mu_\mathsf{w})^T\Sigma_\mathsf{w}^{-1}(\mu_\mathsf{z}-\mu_\mathsf{w}) + \log\left(\frac{\det\Sigma_\mathsf{w}}{\det\Sigma_\mathsf{z}}\right) + \mathrm{Tr}\,\left(\Sigma_\mathsf{z}\Sigma_\mathsf{w}^{-1}\right) - d\right] \tag{3}$$

Algorithm 1 Map refinement using hard clustering

$Z = \{\mathsf{z}_j | j = 1,\dots,N\} \leftarrow$ the N Gaussians of the fine scale s
$A = \{\alpha_j | j = 1,\dots,N\} \leftarrow$ the non negative weights of the fine Gaussians
$W = \{\mathsf{w}_1^0,\dots,\mathsf{w}_k^0\}[k-1] \leftarrow$ the $k-1$ Gaussians of the coarse scale $s+1$
$d_{\mathscr{Z}\times\mathbb{F}}(\cdot,\cdot) \leftarrow$ the distance function

$\{\mathsf{w}_1^0,\dots,\mathsf{w}_k^0\} \leftarrow \{\mathsf{w}_1^0,\dots,\mathsf{w}_k^0\}[k-1] \bigcup \{\mathsf{z}_{\max}^0\}$ *// Init. with the data of maximum distortion*
$\{(C_i^1,\mathsf{w}_i^1)|i=1,\dots,k\}, \mathscr{E}_{(C_i^1,\mathsf{w}_i^1)} \leftarrow \text{kMeans}(Z,A,W,d_{\mathscr{Z}\times\mathbb{F}})$

$t \leftarrow 1$
// Simulate a swap
repeat
  for all C_i^t **do**
    $\mathsf{z}_{\max(i)} \leftarrow \arg\max_{\mathsf{z}_j \in C_i^t} d_{\mathscr{Z}\times\mathbb{F}}(\mathsf{z}_j,\mathsf{w}_i)$
    $d_i \leftarrow d_{\mathscr{Z}\times\mathbb{F}}(\mathsf{z}_{\max(i)},\mathsf{w}_i)$
  end for
  $c_{\max} \leftarrow \arg\max_{i=1,\dots,k} d_i$
  $d_{\max} \leftarrow d_{c_{\max}}$
  $(u_{\min},v_{\min}) \leftarrow \arg\min_{(u,v),\ u\neq v} d_{\mathscr{Z}\times\mathbb{F}}(\mathsf{w}_u,\mathsf{w}_v)$
  $d_{\min} \leftarrow d_{\mathscr{Z}\times\mathbb{F}}(\mathsf{w}_{u_{\min}},\mathsf{w}_{v_{\min}})$
  if $d_{\max} < d_{\min}$ **then**
    $c_{\min} \leftarrow$ the cluster, $C_{u_{\min}}^t$ or $C_{v_{\min}}^t$, with the smallest number of elements.
    $\{\mathsf{w}_1^{t+1},\dots,\mathsf{w}_k^{t+1}\} \leftarrow \left(\{\mathsf{w}_1^t,\dots,\mathsf{w}_k^t\} \setminus \{\mathsf{w}_{c_{\min}}\}\right) \bigcup \{\mathsf{z}_{\max(c_{\max})}\}$
  else
    $c \leftarrow\sim \mathscr{U}(\llbracket 1;k \rrbracket \setminus \{c_{\max}\})$*// Draw a random candidate*
    $\{\mathsf{w}_1^{t+1},\dots,\mathsf{w}_k^{t+1}\} \leftarrow \left(\{\mathsf{w}_1^t,\dots,\mathsf{w}_k^t\} \setminus \{\mathsf{w}_c\}\right) \bigcup \{\mathsf{z}_{\max(c_{\max})}\}$
  end if

  $\{(C_i^{t+1},\mathsf{w}_i^{t+1})|i=1,\dots,k\}, \mathscr{E}_{(C_i^{t+1},\mathsf{w}_i^{t+1})} \leftarrow \text{kMeans}(Z,A,\{\mathsf{w}_1^{t+1},\dots,\mathsf{w}_k^{t+1}\},d_{\mathscr{Z}\times\mathbb{F}})$

until $\mathscr{E}_{(C_i^{t+1},\mathsf{w}_i^{t+1})} > \mathscr{E}_{(C_i^t,\mathsf{w}_i^t)}$ *// Accept the swap if the clustering distortion decreases*

return $\{(C_i^t,\mathsf{w}_i^t)|i=1,\dots,k\}, \mathscr{E}_{(C_i^t,\mathsf{w}_i^t)}$

where d is the dimension. The metric is composed by 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 together fine scale clusters does not only depend on their position, but also on their size an orientation. 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 difference in angle between them.

As explained in [15] the computation of the optimal reference vectors from a set of Gaussians $\{\mathsf{z}_j = (\mu_{\mathsf{z}_j},\Sigma_{\mathsf{z}_j})|j=1,\dots\}$ weighted by positive reals (α_{z_j}), is done in two steps:

First the optimal mean is computed using (4):

$$\mu^* = \frac{1}{\sum_j \alpha_{z_j}} \sum_j \alpha_{z_j} \mu_{z_j} \ , \tag{4}$$

then the covariance matrix is given by (5):

$$\Sigma^* = \left[\frac{1}{\sum_j \alpha_{z_j}} \sum_j \alpha_{z_j} (\Sigma_{z_j} + \mu_{z_j}{}^T \mu_{z_j}) \right] - (\mu^*)^T \mu^* \ . \tag{5}$$

It is interesting to remark that, if the weights are defined to be the number of samples used to compute the Gaussians of the fine scales of the cluster, then the optimal Gaussian is given by the sample mean and covariance of those samples.

4.3 Merging or simplification

We explain no how to merge two Gaussian whose cells contain multiple Gaussian clusters. This is a form of map simplification since the goal is to delete the redundant cluster centers after the union of maps.

The approach consist in the straightforward application of the clustering algorithm to each cell that is occupied in both maps. Here we will assume that the number of clusters per cell in the final map is given, we will discuss later how to choose it (see sec. 7). In addition, we suppose also that the finer scale is already merged (see sec. 7 for the complete algorithm). Thus, the only remaining problem is how to initialize the clustering algorithm.

The idea is to take all the clusters of both maps, then to compute the inter-cluster divergences as in line 15 of Alg. 1. From there, the algorithm proceeds by replacing the pair of clusters having smallest distance by a single cluster, and then starting over until the target number of reference vectors is reached. This is done using equations 4 and 5 which is very efficient because no access to the finer scales is required.

After finishing the merging step, a run of the clustering algorithm is executed to smooth out the result.

5 MAP UPDATING

The off-line algorithm presented so far is not suitable for real-time mapping. Therefore, we have developed a stochastic gradient descent version of the clustering algorithm. This section presents the incremental version, considering both Gaussian

inputs – as for the off-line algorithm –, and point inputs, which are more adapted to the kind of data actually provided by most range sensors.

5.1 Gaussian-based update

Given a cell containing several Gaussians and an input Gaussian z, the algorithm first finds the closest Gaussian – according to the KL divergence – and then updates it. This is similar to the hard assignment made by off-line k-means.

For the update, the principle of a stochastic gradient descent algorithm is to update the reference vector by a fraction of the negative gradient of the distortion for each incomming data. As more and more samples are processed, the magnitude of the adaptation should decrease (typically faster than $1/n$) to ensure convergence. A good example is the on-line computation of the sample mean:

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

where n represents the name of samples processed so far, and $\mathsf{z}^n - \mu^{n-1}$ 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 ε.

In the case of the KL divergence, the gradients for the mean and the covariance are given by (6) and (7), respectively:

$$\frac{\partial\, D_{KL}(\mathsf{z}\|\mathsf{w})}{\partial\, \mu_\mathsf{w}} = -{\Sigma_\mathsf{w}}^{-1}(\mu_\mathsf{z} - \mu_\mathsf{w}) \tag{6}$$

$$\frac{\partial\, D_{KL}(\mathsf{z}\|\mathsf{w})}{\partial\, \Sigma_\mathsf{w}^{-1}} = \Sigma_\mathsf{z} + (\mu_\mathsf{z} - \mu_\mathsf{w})^T(\mu_\mathsf{z} - \mu_\mathsf{w}) - \Sigma_\mathsf{w} \tag{7}$$

So, in order to get a zero covariance gradient, one has to follow the direction given by Eq.(8) (*c.f.* [16]):

$$\frac{\partial\, D_{KL}(\mathsf{z}\|\mathsf{w})}{\partial\, \Sigma_\mathsf{w}} \propto -(\Sigma_\mathsf{z} + (\mu_\mathsf{z} - \mu_\mathsf{w})^T(\mu_\mathsf{z} - \mu_\mathsf{w}) - \Sigma_\mathsf{w}) \tag{8}$$

Since the formulae are simple, it is possible to accelerate convergence using a Gauss-Newton procedure, yielding the following update algorithm:

Algorithm 2 Map update with Gaussians: gaussianUpdate

$\{\mathsf{w}_1,\ldots,\mathsf{w}_k\} \leftarrow$ the k Gaussian reference vectors of the cell
$\{\varepsilon_j | i = 1,\ldots,k\} \leftarrow$ the associated learning rates
$\mathsf{z} = (\mu_\mathsf{z}, \Sigma_\mathsf{z}) \leftarrow$ the Gaussian observation

$n \leftarrow \arg\min_{i=1,\ldots,k} D_{KL}(\mathsf{z} \| \mathsf{w}_i)$
$\mu_{\mathsf{w}_n} \leftarrow \mu_{\mathsf{w}_n} + \varepsilon_n(\mu_\mathsf{z} - \mu_{\mathsf{w}_n})$
$\Sigma_{\mathsf{w}_n} \leftarrow \Sigma_{\mathsf{w}_n} + \varepsilon_n \left[\Sigma_\mathsf{z} + (\mu_\mathsf{z} - \mu_{\mathsf{w}_n})^T(\mu_\mathsf{z} - \mu_{\mathsf{w}_n}) - \Sigma_{\mathsf{w}_n}\right]$

As it can be seen, the learning rates are not defined here, their computation is detailed in section 6.

5.2 Point-based update

In the case of points, a distance metric between a point and a Gaussian should be used. We have chosen to use a spherical measure given by (9):

$$d(\mathsf{p},\mathsf{w}) \triangleq \frac{1}{2}\left[(\mathsf{p}-\mu_\mathsf{w})^T {\Sigma_\mathsf{w}}^{-1}(\mathsf{p}-\mu_\mathsf{w}) + \log\left(\det(\Sigma_\mathsf{w})\right)\right] , \tag{9}$$

This distance is the addition of the Mahalanobis distance and a volume term. Compared to the pure Mahalanobis distance, the volume term aims to compensate the fact that the Mahalanobis distance of a big cluster tends to make every point very close. This measure has the advantage of yielding simple map update rules, since its derivative is:

$$\frac{\partial\, d(\mathsf{p},\mathsf{w})}{\partial\, \mu_\mathsf{w}} = -{\Sigma_\mathsf{w}}^{-1}(\mathsf{p}-\mu_\mathsf{w}) \tag{10}$$

and

$$\frac{\partial\, d(\mathsf{p},\mathsf{w})}{\partial\, \Sigma_\mathsf{w}} \propto -\left[(\mathsf{p}-\mu_\mathsf{w})(\mathsf{p}-\mu_\mathsf{w})^T - \Sigma_\mathsf{w}\right] \tag{11}$$

Giving the following gradient descent algorithm for point-based update:

Algorithm 3 Map update with points: pointUpdate

$\{\mathsf{w}_1,\ldots,\mathsf{w}_k\} \leftarrow$ the k Gaussian reference vectors of the cell
$\{\varepsilon_j | i = 1,\ldots,k\} \leftarrow$ the associated learning rates
$\mathsf{z} = \mathsf{p} \leftarrow$ the observed point

$n \leftarrow \arg\min_{i=1,\ldots,k} d(\mathsf{z},\mathsf{w}_i)$
$\mu_{\mathsf{w}_n} \leftarrow \mu_{\mathsf{w}_n} + \varepsilon_n(\mathsf{p} - \mu_{\mathsf{w}_n})$
$\Sigma_{\mathsf{w}_n} \leftarrow \Sigma_{\mathsf{w}_n} + \varepsilon_n \left[(\mathsf{p} - \mu_{\mathsf{w}_n})^T(\mathsf{p} - \mu_{\mathsf{w}_n}) - \Sigma_{\mathsf{w}_n}\right]$

6 Occupancy

A central element of our approach that has been discussed yet is occupancy. It plays two fundamental roles. First, it provides a criterin to filter dynamic objects out of the map. Second, it constitutes the basis for determining the local plasticity of the map through its use to compute the learning rate of the clustering algorithm.

For dynamic obstacle filtering, we use occupancy in log-ratio space as presented in [17]. From this point of view, occupancy can be seen as a counter associated to every object. Its value gets increased when the object is visible in the range image, or decreased when the object is supposed to be visible but it is not. Fig. 4 illustrates the idea: if the point C, is visible – the dashed red line is free from obstacles between I and C – then the value of the range image at cell I will correspond to the distance r_C. If, on the other hand, the value of cell I is greater than r_C, then C violates the visibility rules and its occupancy should be decreased.

We attach an occupancy value to each Gaussian cluster of the map and when its occupancy decreases below a given threshold, the cluster is removed from the map.

In the followin sections, we detail occupancy computation and its use as the learning rate of the update algorithms.

6.1 Computing occupancy

We compute occupancy in a hierarchical fashion to perform early supression of clusters that are not visible. For a given cell c of a given scale s, we project the bounding sphere onto the range image (see fig. 4). We consider that the range image is acquired through a perspective range camera [1]. To compute the position of a point in the range image, a simple affine transformation is performed then and the real range of the projected point is associated to the image range cell. If the projection is outside the range image (camera field of view, blue lines in fig. 4), the finer children cells of c are no further explored. The search is also finished if all the ranges observed in the disc of the projected bounding sphere (orange disc in the image plane

[1] Which is always a correct approximation as long as the range image as a small enough angular field of views. If it is not the case it is possible to splitt the image into smaller sub images

fig. 4) are smaller than the smallest possible range for an object contained in the bounding sphere of the cell (that is the range of the center C of c, minus the radius of the bounding sphere).

For the visible cells, we compute occupancy in a per point basis. The occupancy of a point is given by comparing the range in the pixel of the projected point in the range image with the actual range of the point. For a Gaussian, we sample it with n points (rejecting points that fall outside the cell) and we compute the Gaussian occupancy as the average of the observed occupancy of each of its point. To define n, we compute an upper bound of the number of data in the range image that can be contained in the cell using the projected bounding sphere of the cell. If $\delta_{\min}$ is the distance of the image plane in the camera coordinate system, and δ the distance of the center of c (fig. 4), the projection of the bounding sphere of c occupies an area of $\frac{3\pi}{4}\left(\frac{\delta_{\min}}{\delta}a\right)^2$ (orange disc in fig. 4), where a is the length of the side of c. Knowing the area of one pixel of the range image p an upper bound for the number of pixels is:

$$B \triangleq \lceil \frac{3\pi}{4p}\left(\frac{\delta_{\min}}{\delta}a\right)^2 \rceil \tag{12}$$

Then we use $n = B$ samples per cluster, which provides a good chance to cover every range image cell that effectively contains an observation from the cluster.

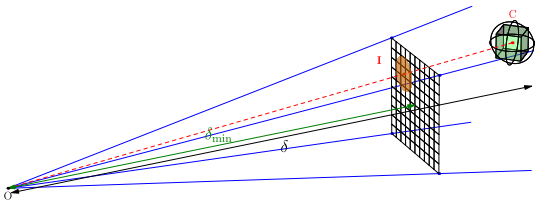

Fig. 4 Computation of occupancy in the range image of the bounding sphere of a cell.

6.2 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 it position and shape are supposed to be, thus a small value of ε 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 unsufficient 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 into $[\varepsilon_{\min}, \varepsilon_{\max}]$. For our approach we have chosen a linear mapping between both values:

$$\varepsilon(o) = \frac{\varepsilon_{\min} - \varepsilon_{\max}}{2o_{\max}}(o + o_{\max}) + \varepsilon_{\max} \tag{13}$$

In our experiments, we have set $o_{\max} = 10.0$, $\varepsilon_{\max} = 5 \cdot 10^{-2}$ and $\varepsilon_{\min} = 5 \cdot 10^{-4}$.

7 ERROR DRIVEN REFINEMENT OF COARSE SCALES

Our refinement algorithm makes use of both occupancy and distortion to determine which cell should be refined first. This section presents the details of our error criterion and then discusses the practical details of budget allocation.

7.1 Error computation

To find the cell to refine, we do not use the same KL divergence than to do the clustering. Instead of using $D_{KL}(\mathsf{z}\|\mathsf{w})$, we use rather $D_{KL}(\mathsf{w}\|\mathsf{z})$. As the KL divergence is not symmetric, this is an important change and it is relevant to achieve good results. Our choice is motivated by the fact that the coarse scale error is computed from finer scales, meaning that those finer scales are the primary source of error.

For each cell c^s of the coarse scale, s, with reference vectors $\{\mathsf{w}_1, \ldots, \mathsf{w}_k\}$ and finer data at $s-1$: $\{\mathsf{z}_1, \ldots, \mathsf{z}_N\} \in G(\phi(c^s))$ we compute the average distortion of the data to their closest reference vector with the KL divergence:

$$\mathscr{E}(c^s) = \frac{1}{N}\sum_{i=1}^{k}\sum_{j=1}^{N}(1 - \varepsilon_{\mathsf{z}_j})\delta(\mathsf{w}_i, \mathsf{z}_j)D_{KL}(\mathsf{w}_i\|\mathsf{z}_j)\ , \tag{14}$$

where $\delta(\mathsf{w}_i, \mathsf{z}_j)$ is one if w_i is the closest reference vector to z_j using $D_{KL}(\mathsf{w}_i\|\mathsf{z}_j)$ and zero otherwise. The occupancy is used through the learning rate to assign higher error weights to occupied clusters, disregarding low occupancy ones whose accuracy may still improve without the need of adding extra clusters.

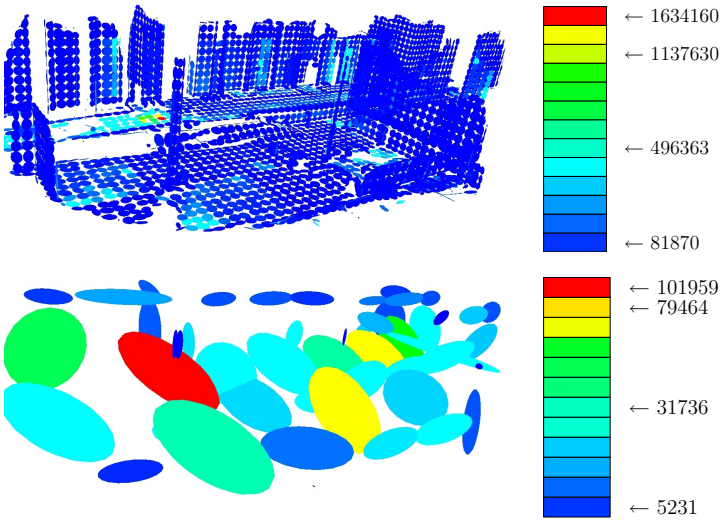

Fig. 5 Up: fine scale is colored with the magnitude of the contribution to the error at the coarser scale. Down: coarse scale, mean error. Error palettes are on the right.

7.2 Error driven updates

Most of the update time is spent at the finest scale, therefore it is interesting to search for mechanisms that avoid fine scale update. The main question here is: if a coarse scale accurately represents the measurements, why updating the finer scales? In this case, no update is done on the finer scales, saving a lot of computation.

7.3 Budget allocation

We periodically refine the map by adding a fixed number p of reference vectors at a time. In order to choose the p vectors that have the maximum distortion without sorting the whole set of reference vectors, we use a priority queue of size p as was done in [11].

Since our goal is to add clusters in places where the error is high in spite of having used many samples, we choose to refine a cell c only if the average occupancy probability of the finer cells in $\phi(c)$ is above 0.5. Furthermore, we concentrate only on the part of the map that is visible for the sensor.

8 RESULTS

To evaluate the results, we use the Hausdorff distance between two point sets. This distance calculates the distance in the worst case. It calculates the pair of closest points in both sets and take the maximum of these distances[2]. To generate an interesting point cloud for the Gaussian map, we sample each Gaussian, rejecting points that fall outside the cells. For real data, we already have points clouds and for simulated data (based upon triangle meshes) we sample each triangle by a set of points. To choose the number of samples the size of the object is considered (the volume of the ellipsoids for Gaussians and the area for the triangles).

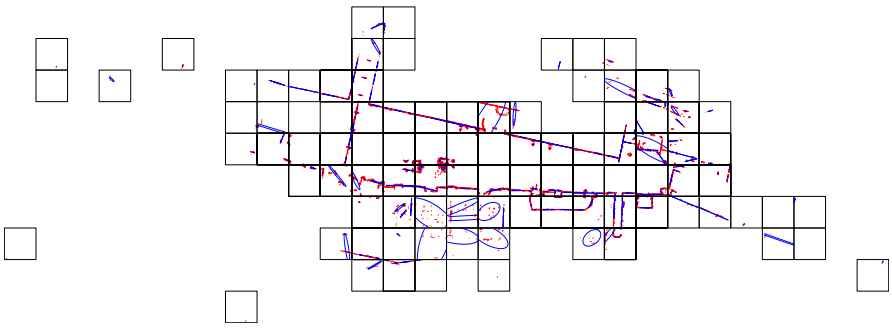

Fig. 6 2-D real dataset from [18]. The coarsest and the finest map are displayed (the image is in a vector format, so you can zoom on it to have the desired resolution). The black squares are the coarse cells, in red are the finest Gaussians (which appears like dots) and in blue are the ellipsoids of the covariances of the coarsest scale, plotted with 3 times the square of the eigenvalues.

The results, we have obtained on real and simulated data-sets share the same qualities and drawbacks. For each data set we use 3 scales, at each scale the cell side is ten times larger. For the 2-D data set (fig. 6) the finest side is 0.5 cm and for the $3-d$ data set the finest side is 0.1 cm [3]. The method makes a real good use of each added cluster since by adding just a few cluster per cell (typically less than 4), the Hausdorff distance is almost divided by 3 in each cell. The huge reduction of the number of cells (one for 10^4 in 2-D and one for 10^8 in 3-D) between the coarsest scale and the finest does not make the coarse scales useless any longer for a cheap augmentation of the number of elements. For instance we add 128 clusters in the coarsest scale of the 3-D data set, which makes 181 clusters to represent 423 818 clusters at the finest scale which corresponds to less than 0.05% of the ressources of the finest scale.

[2] Note that we cannot use this distance for refinement since it is too costly to compute in real-time.

[3] we don't show the finest scale in 3-D, since the Gaussians are too small and neither the first scale in 2-D, since it is the less interesting.

However there is still drawbacks: sometimes the Hausdorff distance does not decrease so much when a cluster is added. We think that two reasons can explain that phenomenon: the discretization can be the first problem, since as object are randomly cut by the cells, at coarse scale a cell can contain a big clusters and a small ones. In this case the error is concentrated on the small ones and when a new cluster is added, the decrease of the error is therefore small. The second problem is that the resulting Gaussian can be in the middle of two separate clusters. When sampling the Gaussian, points are sampled in the empty space in-between and errors is artificially high here.

9 Conclusions and Future Work

In this paper we have proposed a comprehensive framework to build two and 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 able to preserve important features of the environment (*e.g.* corners) that are usually smoothed out by other approaches. The framework provides a theoretically sound foundation for map merging. In order to deal with moving objects and noise, our approach makes use of occupancy to determine when to delete parts of the map that have become empty, as well as to adjust the plasticity of the map. Experiments with real and simulated data show promising results concerning the accuracy and compactness of the maps produced by our approach.

Further work includes working towards real time mapping by exploiting parallelization and studying the application of our approach to higher dimensional spaces in order to include point properties such as color.

References

1. Alberto Elfes. *Occupancy grids: a probabilistic framework for robot perception and navigation*. PhD thesis, Carnegie Mellon University, 1989. 1
2. Alex Yahja, Anthony (Tony) Stentz, Sanjiv Singh, and Barry Brummit. Framed-quadtree path planning for mobile robots operating in sparse environments. In *IEEE Conference on Robotics and Automation*, Leuven, Belgium, May 1998. 1
3. 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. 1
4. Nora Ripperda and Claus Brenner. Marker-free registration of terrestrial laser scans using the normal distribution transform. Technical report, Institute of Cartography and Geoinformatics, University of Hannover, Germany, 2005. 1, 2

5. Gérard Medioni, Mi-Suen Lee, and Chi-Keung Tang. *A Computational Framework for Segmentation and Grouping*. Elsevier Science Inc., New York, NY, USA, 2000. 2, 4
6. 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, 4
7. Giorgio Grisetti, Cyrill Stachniss, and Wolfram 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*, pages 2443–2448, 2005. 2
8. Rüdolf Triebel, Patrick Pfaff, and Wolfram Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006. 3
9. Pierre Payeur, Patrick Hébert, Denis Laurendeau, and Clément Gosselin. Probabilistic octree modeling of a 3-d dynamic environment. In *Proc. IEEE ICRA 97*, pages 1289–1296, Albuquerque, NM, Apr. 20-25 1997. 3
10. M. Yguel, C. Tay Meng Keat, C. Braillon, C. Laugier, and O. Aycard. Dense mapping for range sensors: Efficient algorithms and sparse representations. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007. 3
11. Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. *Computer Graphics*, 31(Annual Conference Series):209–216, 1997. 4, 14
12. David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. In *ACM SIGGRAPH 2004 Papers*, pages 905–914, 2004. 4, 5
13. Mark Pauly, Markus Gross, and Leif P. Kobbelt. Efficient simplification of point-sampled surfaces. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 163–170, Washington, DC, USA, 2002. IEEE Computer Society. 5
14. S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, Mar 1982. 6
15. 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. 6, 7
16. Jan R. Magnus and Heinz Neudecker. *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons, 2nd edition, 1999. 9
17. M. Yguel, O. Aycard, and C. Laugier. Efficient gpu-based construction of occupancy grids using several laser range-finders. *Int. J. on Vehicle Autonomous System*, 6(1/2):48–83, 2008. 11
18. Cyrill Stachniss. Corrected robotic log-files. http://www.informatik.uni-freiburg.de/ stachnis/datasets.html. 15