Error-Driven Refinement of Multi-scale Gaussian Maps Application to 3-D Multi-scale map building, compression and merging

Manuel Yguel, Dizan Alejandro Vasquez, Olivier Aycard, Roland Y. Siegwart, Christian Laugier

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 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 in which the number of Gaussians should be increased. As a result, the accuracy of the map can be selectively enhanced by making efficient use 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 rather than points, as is the case for conventional algorithms. In addition to mapping, this clustering paradigm makes it possible 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

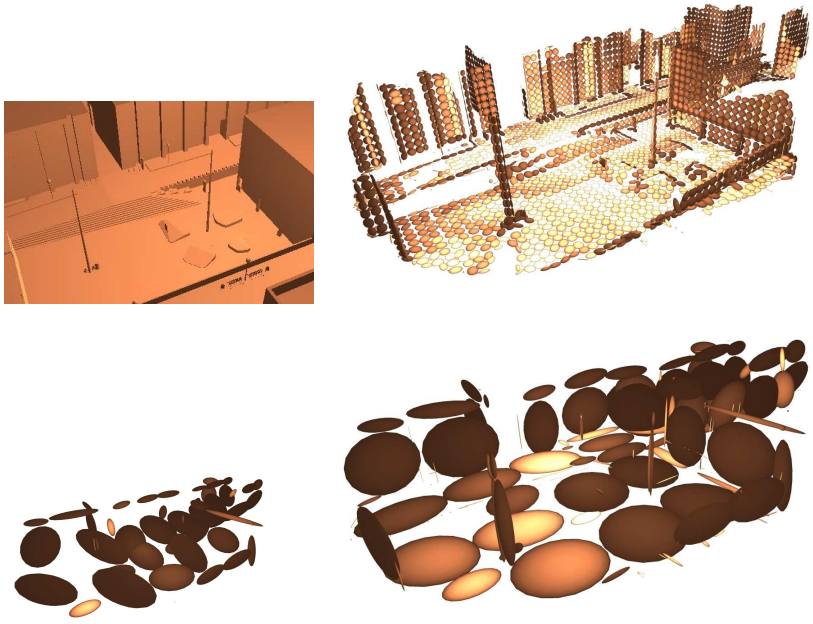

Fig. 1 Top left: simulated scene. Top right: fine scale of the map. Bottom left: coarse scale of the map, one Gaussian per cell. Bottom right: coarse scale of the refined map

at a low computational cost. Then, at a second stage, this estimate is used to initialize the fine scale search algorithms, thus accelerating convergence. For localization, this procedure also enlarges – in most cases – the convergence region. Examples of algorithms that benefit from 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. 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. 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 shape of the cell contents 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 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. 1). In this paper, we present a method to overcome this problem by allowing a coarse resolution cell to contain multiple ellip-

soids – 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 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 through a refinement process.

This paper is structured as follows: In the following section, we review related works in robotics and computer graphics. Section 3 provides an overview of our mapping framework. In section 4, we explain how to update a Gaussian map from a range image and how occupancy may be used as a learning rate. Section 5 presents our error-driven refinement algorithm for coarse scales. Section 6 discusses mapping results on simulated and real data sets. Finally, we present our conclusions and outline possible directions for future work.

2 RELATED WORKS

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. An MLS map is a 2-D grid where, for each cell, several planes are stored at different heights, together with the associated thickness. Their structure makes 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 occluded 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 [9, 10] are the most popular data structures of this kind for two and three dimensional input spaces, respectively. 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 at improving 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. 2).

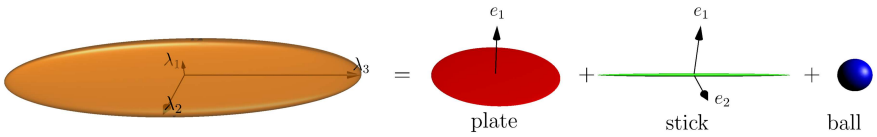

Fig. 2 Decomposition of an SSDP matrix ellipsoid into elementary shapes called tensors in [5].

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 the number of Gaussians added 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 up 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 defined by calculating 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, the error of all the affected edges is recomputed and they are reinserted in the queue. The process continues until the target budget of edges is reached.

A second class of effective simplification approaches is based on clustering. They can operate either on meshes or on point clouds. Probably the most relevant example of mesh clustering is [12] where clustering is performed on the triangles of the mesh to be simplified. An essential component of this approach is a shape metric that makes it possible to assign each triangle to its closest cluster and to compute the parameters of the cluster. Cohen and Steiner [12] consider two metrics: Garland's quadric error metric [11] and the Euclidean distance between the normals of the triangles and their cluster normals.

In [13], Pauly *et al.* have studied the simplification of point clouds of the same kind than those obtained with a laser range finder. They describe several agglomerative and partitional approaches, applying techniques proposed in [11].

Our approach is largely inspired by the work of Cohen and Steiner [12] and by the partitional 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

Our approach processes every scale in parallel, adjusting the field of view of finer scales in order to have a similar number of cells to update for each scale. As shown in Fig. 3, the approach is composed of three main components:

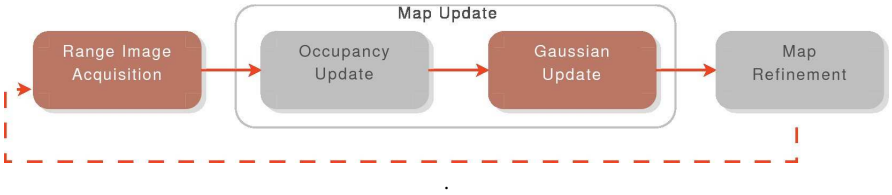

Fig. 3 Framework components. Light gray boxes are processed less than once per range image.

1. Occupancy computation: the occupancy is a measure of how often a particular cluster has been observed. In our framework is has two uses. First, it determines the plasticity of a cluster, allowing it to adapt faster in regions that have not been observed often. Second, it is used as a criterion to delete clusters, making it possible to remove dynamic objects from the map.
2. Map updating: it adapts existing clusters in order to minimize the representation error. It is similar to the standard update of conventional Gaussian maps, except that it takes into account the fact that a cell may contain several Gaussians.
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. It is worth noting that no refinement is performed on the finest scale.

4 MAP UPDATING

This section presents the procedure to update an existing map from 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 § 5.

Our goal here is to update the Gaussians' mean value μ and covariance Σ in order to minimize the representation error with respect to the input data. Every point in the range image is used to incrementally update the different scales independently. As we will explain in § 4.3, 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 reasonable it is to modify it. As we will see in § 5, occupancy is also used as a criterion to filter out dynamic objects from the map.

4.1 Computing cluster occupancy

Occupancy can be seen as a counter associated to every object. Its value gets increased when the object is visible in the range image, and decreased when the object is supposed to be visible but is in fact not. Fig. 4 illustrates the idea: if point C, is visible – *i.e.* 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, this can be considered as evidence that C is not there anymore and its occupancy should be decreased.

For visible cells, we compute occupancy in a per point basis. The occupancy of a point, C, in the map is given by comparing the range measured in the pixel of its projection, I, in the range image with its actual range, δ. For a Gaussian, the occupancy is obtained by averaging the occupancy of n points sampled from the Gaussian distribution (rejecting those that fall outside the cell).

We only need to guarantee that there are enough samples to provide a good estimate. To define n, we compute an upper bound of the number of points in the range image that can be contained in the cell. This is done using the projected bounding sphere of the cell. Let $\delta_{\min}$ and δ be the distance to the image plane in the camera coordinate system and the distance to the center of c (fig. 4), respectively. Then 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 that may be projected back into the original cluster is:

$$B \triangleq \lceil \frac{3\pi}{4p}\left(\frac{\delta_{\min}}{\delta}a\right)^2 \rceil \tag{1}$$

which is the number of samples we are looking for. So, making $n = B$ gives us a good chance to cover every range image cell that effectively contains an observation from the cluster.

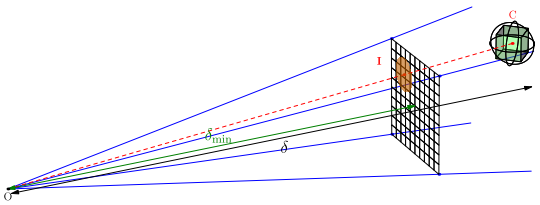

Fig. 4 Computation of occupancy in the range image of the bounding sphere of a cell.

4.2 Hierarchical culling

In order to minimize computation, we perform hierarchical visibility culling. Consider the cell c centered in C in (fig. 4). If the projection of the bounding sphere of c is outside the range image (camera field of view, blue lines in fig. 4), the finer children cells of c are not 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) are smaller than the smallest possible range for the cell, meaning that all the cell content is occluded by closer objects.

4.3 Updating Gaussian clusters from data

For every input point, a single cluster per scale will be updated. The cluster is selected by finding the cell that contains the input point 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 reference vector by a fraction of the negative gradient with respect to each input datum. 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 our approach, the value of ε depends on the occupancy, as described in § 4.4.

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 (2):

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

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. This measure has the advantage of yielding simple map update rules, since its derivative is:

$$\frac{\partial\, d(\mathsf{p},\mathsf{w})}{\partial\, \mu_{\mathsf{w}}} = -{\Sigma_{\mathsf{w}}}^{-1}(\mathsf{p}-\mu_{\mathsf{w}}) \tag{3}$$

and

$$\frac{\partial\, d(\mathsf{p},\mathsf{w})}{\partial\, \Sigma_{\mathsf{w}}} \propto -\left[(\mathsf{p}-\mu_{\mathsf{w}})(\mathsf{p}-\mu_{\mathsf{w}})^T - \Sigma_{\mathsf{w}}\right] , \tag{4}$$

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

Algorithm 1 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

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

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

$$\varepsilon(o) = \frac{\varepsilon_{\min} - \varepsilon_{\max}}{2o_{\max}}(o + o_{\max}) + \varepsilon_{\max} \tag{5}$$

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

5 ERROR-DRIVEN REFINEMENT OF COARSE SCALES

The refinement process is driven by a measure of the representation error. The map is 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 scale as input.

We periodically refine the map by adding a fixed number p of Gaussian clusters at a time. In order to choose the p vectors that have the maximum error without sorting the whole set of reference vectors, we use a priority queue of size p as was done in [11]. The following subsections provide the details of the refinement algorithm: the error metric used to build the queue is introduced in § 5.1 and § 5.2 presents the clustering algorithm.

During the mapping process it is often necessary to delete clusters that correspond to moving obstacles; this process is described in § 5.3. Finally, the application of our approach to map merging and simplification is discussed in § 5.4.

5.1 Error computation

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 the average occupancy probability of the finer cells in $\phi(c)$ is above 0.5. Furthermore, we only refine those parts of the map that are visible for the sensor.

To find the cell to refine, we compute an error value per cell. This value is basically the sum of the Mahalanobis distance between the center of the coarse cluster and the Gaussian cluster of the finer scale.

For the cells c^s of the coarse scale, s, having reference vectors (*i.e.* mean values) $\{\mathsf{w}_1, \ldots, \mathsf{w}_k\}$ and finer data at $s-1$: $\{\mathsf{z}_1, \ldots, \mathsf{z}_N\} \in G(\phi(c^s))$, we compute the average distance of each datum to its closest reference vector:

$$\mathscr{E}(c^s) = \frac{1}{N}\sum_{i=1}^{k}\sum_{j=1}^{N}(1-\varepsilon_{\mathsf{z}_j})\delta(\mathsf{w}_i,\mathsf{z}_j)\ \ (\mu_{\mathsf{w}_i} - \mu_{\mathsf{z}_j})^T \Sigma_{\mathsf{z}_j}^{-1}(\mu_{\mathsf{w}_i} - \mu_{\mathsf{z}_j})\,, \tag{6}$$

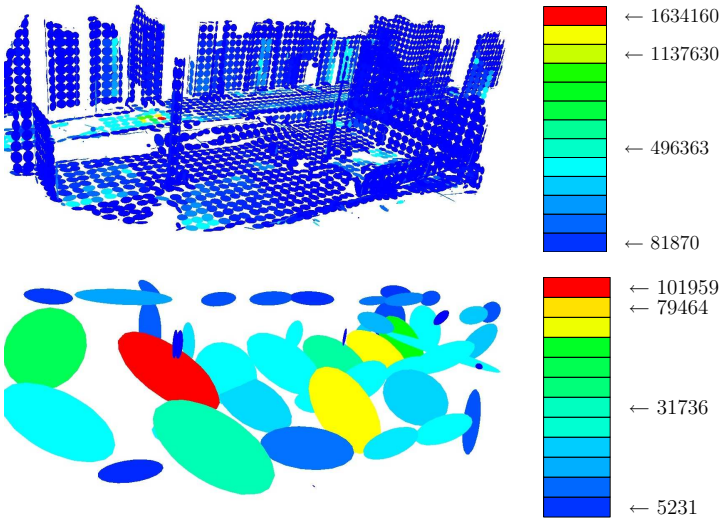

Fig. 5 Up: fine scale is colored with the magnitude of the contribution to the error at the coarser scale. Down: coarse scale, mean error. Error palettes are on the right.

where $\delta(\mathsf{w}_i, \mathsf{z}_j)$ is one if w_i is the closest reference vector to z_j using the Mahalanobis distance defined by z_j and zero otherwise. The occupancy is used through the learning rate 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.

5.2 Clustering for map refinement

Algorithm 2 describes 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 $G(\phi(c^s))$ into k classes represented by k reference vectors that minimizes the clustering distortion:

$$\mathscr{E}_{(C^*,\{\mathsf{w}_1^*,\ldots,\mathsf{w}_k^*\})} = \underset{\{C_1,\ldots,C_k\},\{\mathsf{w}_1,\ldots,\mathsf{w}_k\}}{\arg\min} \sum_{i=1}^{k} \mathscr{E}_{C_i}(\mathsf{w}_i) \tag{7}$$

This is done by 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.

Algorithm 2 Map refinement using hard clustering

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

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

// *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
  $c_{\max} \leftarrow \arg\max_{i=1,\ldots,k} d_i$
  $d_{\max} \leftarrow d_{c_{\max}}$
  $(u_{\min}, v_{\min}) \leftarrow \arg\min_{(u,v),\, u\neq v} d_{\mathscr{Z}\times\mathbb{F}}(\mathsf{w}_u, \mathsf{w}_v)$
  $d_{\min} \leftarrow d_{\mathscr{Z}\times\mathbb{F}}(\mathsf{w}_{u_{\min}}, \mathsf{w}_{v_{\min}})$
  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.
    $W^{t+1} \leftarrow (W^t \setminus \{\mathsf{w}_{c_{\min}}\}) \bigcup \{\mathsf{z}_{\max(c_{\max})}\}$
  else
    $c \leftarrow\sim \mathscr{U}([\![1;k]\!] \setminus \{c_{\max}\})$// *Draw a random candidate*
    $W^{t+1} \leftarrow (W^t \setminus \{\mathsf{w}_c\}) \bigcup \{\mathsf{z}_{\max(c_{\max})}\}$
  end if

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

  $t \leftarrow t+1$
until $\mathscr{E}_{W^t} > \mathscr{E}_{W^{t-1}}$ // *Accept the swap if the clustering distortion decreases*

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

An important drawback of k-means is that is highly dependent on initialization. Moreover, even if the algorithm is guaranteed to converge, it often gets stuck in local minima of $\mathscr{E}_{W^*}$. To get out of the local minima a so called "swap" procedure is used (line 7 to 25, alg. 2). 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. A reference vector metric is used to evaluate the similarity between clusters:

$d_{\mathbb{F}} : (\mathbb{F} \times \mathbb{F}) \rightarrow \mathbb{R}^+$. 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 16, alg. 2).

It is worth noting that this clustering framework naturally defines a hierarchy: a cluster is the parent of all the clusters of the finer scale that are closer to it than to any other cluster (see fig. 6).

5.2.1 K-Means extension for Gaussian inputs

In order for the covariance matrices of the clusters at the coarse scale to be as accurate as possible, we need to use the information provided by the covariance matrices at the finer scale. Therefore, we need a clustering algorithm that is able to properly handle Gaussian input data $\mathscr{Z} = \mathbb{F} = \mathscr{G}^3 \triangleq \{(\mu, \Sigma) | \mu \in \mathbb{R}^3 \text{ and } \Sigma \text{ is SDP}\}$ [1]. Davis [15] has proposed such an algorithm, proving that it converges. The algorithm uses the Kullback-Leibler divergence (Eq. 8) as a distance function $d_{\mathscr{Z} \times \mathbb{F}}$ for Gaussians:

$$D_{KL}(\mathsf{z} \| \mathsf{w}) = \frac{1}{2} \left[(\mu_{\mathsf{z}} - \mu_{\mathsf{w}})^T \Sigma_{\mathsf{w}}^{-1} (\mu_{\mathsf{z}} - \mu_{\mathsf{w}}) + \log \left(\frac{\det \Sigma_{\mathsf{w}}}{\det \Sigma_{\mathsf{z}}} \right) + \mathrm{Tr} \left(\Sigma_{\mathsf{z}} \Sigma_{\mathsf{w}}^{-1} \right) - d \right] \tag{8}$$

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

First the optimal mean is computed using (9):

$$\mu^* = \frac{1}{\sum_j \alpha_{\mathsf{z}_j}} \sum_j \alpha_{\mathsf{z}_j} \mu_{\mathsf{z}_j} \ , \tag{9}$$

then the covariance matrix is given by (10):

$$\Sigma^* = \left[\frac{1}{\sum_j \alpha_{\mathsf{z}_j}} \sum_j \alpha_{\mathsf{z}_j} (\Sigma_{\mathsf{z}_j} + \mu_{\mathsf{z}_j}{}^T \mu_{\mathsf{z}_j}) \right] - (\mu^*)^T \mu^* \ . \tag{10}$$

[1] SDP matrices are a subset of SSDP matrices, meaning that analysis tools such as those proposed for NDT [6], tensor voting [5] and quadric error [11] approaches, may be applied to them.

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

5.3 Cluster deletion

In order to account for dynamic objects that have been detected once and that are not there anymore, we delete those clusters whose occupation has fallen below a given threshold. As for cluster insertion, the remaining Gaussians are adjusted by running the clustering algorithm. It should be stressed that, for the sake of consistency, cluster deletion at a given scale should only happen when no corresponding clusters exist at the finer scales.

5.4 Map merging and simplification

Now, we explain how to merge two maps whose cells contain multiple Gaussian clusters. This is a form of map simplification since the goal is to delete redundant cluster centers after the union of maps.

Merging is performed through straightforward application of the clustering algorithm to every cell that appears as occupied in both maps. In order to fix the number of clusters for a given cell, we select the maximum number in the two maps. This, of course, can be later refined as described above. Thus, the main problem is the initialization of the clustering algorithm.

The idea is to take all the clusters of both maps, then to compute the inter-cluster divergences as in line 14 of Alg. 2. From there, the algorithm proceeds by replacing by a single cluster, the pair of clusters that are separated by the smallest distance, and then starting over until the target number of clusters is reached. This is done using equations 9 and 10. The procedure 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.

It is worth noting that the same process can be used to simplify the map when a reduction in the number of clusters is required.

6 RESULTS

We use the Hausdorff distance to evaluate the results. This metric may be seen as the worst-case distance. For two sets of points, it is obtained by computing, for every point in one set, the distance to closest point in the other set, and then taking the maximum of these distances. In our case, we compute this distance by sampling

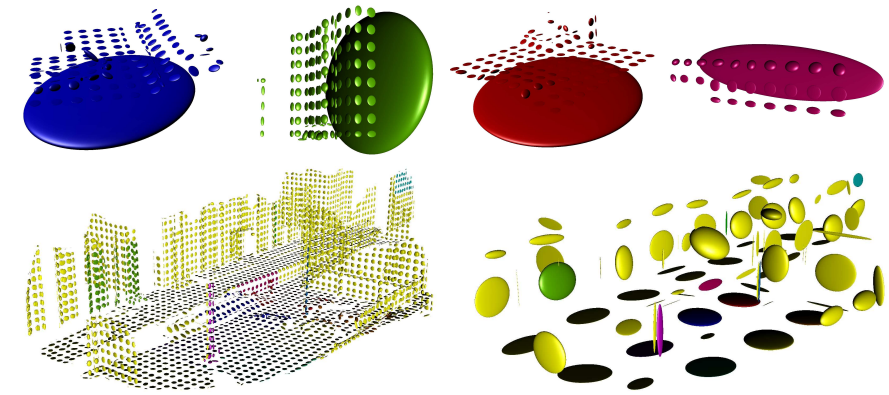

Fig. 6 Cluster hierarchy. Up: coarse clusters are translated for better display.

Gaussians and rejecting points that fall outside the cells. The samples are directly measured against point clouds in the case of real data. In the case of simulated data, the ground truth is available in the form of the original mesh; the corresponding points are obtained by representing each triangle with a number of samples. In all cases, the number of samples is proportional to the size of the object being considered: the volume of the ellipsoids in the case of Gaussians and the area in the case of triangles.

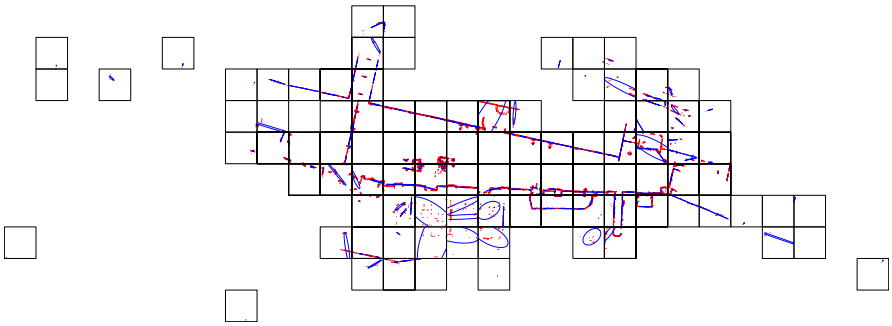

Fig. 7 Real 2-D dataset from [16]. The coarsest (blue ellipsoids) and finest (red ellipsoids) maps are displayed. Black squares represent the coarse cells. All ellipsoids correspond to the covariance matrices plotted with three times the square of the eigenvalues.

For each data set we use 3 scales; at each scale the cell side is ten times larger than the finer one. For the 2-D data set (fig. 7) the finest side is 5 cm and for the 3-D data set the finest side is 10 cm. Regarding the algorithm approach modules (Fig. 3) we set the occupancy computation to take place at acquisition rate for two dimensional data and every three acquisitions for the three dimensional case. As for

refinement, the algorithm has been configured to insert 4 clusters of the remaining Gaussian budget, every 10 acquisitions.

The results we have obtained on real and simulated data sets exhibit similar advantages and drawbacks:

- Advantages:
  - *Accuracy vs map size:* the method is able to significantly improve accuracy with a relatively small increase in the size of the model. In our experiments increasing the number of clusters by four leads to a factor of three reduction of the Hausdorff distance.
  - *Multi-scale error reduction:* the huge size reduction ratio (10^4 to 1 in 2-D and 10^8 to 1 in 3-D) between the finest and the coarsest scales is kept by the refinement process, while considerably improving accuracy. For instance, when refining the coarsest map in the 3-D data set, we increase the number of clusters from 53 to 181 and reduce the error by 3. Note that large flat clusters remain (in particular on the ground and on the walls) while a lot of detail clusters are added at poles and corners (fig. 1). This could not have been done by simply adding an intermediate scale.
- Drawbacks: the main problem we have detected is that, sometimes, the Hausdorff distance does not have a significant decrease when a cluster is added. We believe that there are two reasons for this: first, an aliasing phenomenon that arises from the fact that the underlying cells force the algorithm to artificially cut a big object in pieces, some of which can be very small with respect to other Gaussians in the same cell, leading to big error contributions because of the small size of the covariance. The second reason is that, when the 'right' number of clusters is not yet reached, the resulting Gaussian may represent two separate clusters, yielding a smaller but still important value for the Hausdorff distance.

7 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 that, for

coarse scales, 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. Middle term research will be directed to exploring the application of our approach to higher dimensional spaces that 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, 12
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, 12
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, 9, 12
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. 4, 5
14. S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, Mar 1982. 10
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. 12
16. Cyrill Stachniss. Corrected robotic log-files. http://www.informatik.uni-freiburg.de/ stachnis/datasets.html. 14