Context and Feature Sensitive Re-sampling from Discrete Surface Measurements

David M Cole and Paul M Newman
Oxford University Mobile Robotics Group, Department of Engineering Science, Oxford, OX1 3PJ, UK
Email: dmc,pnewman@robots.ox.ac.uk

***Abstract*— This paper concerns context and feature-sensitive re-sampling of workspace surfaces represented by 3D point clouds. We interpret a point cloud as the outcome of repetitive and non-uniform sampling of the surfaces in the workspace. The nature of this sampling may not be ideal for all applications, representations and downstream processing. For example it might be preferable to have a high point density around sharp edges or near marked changes in texture. Additionally such preferences might be dependent on the semantic classification of the surface in question. This paper addresses this issue and provides a framework which given a raw point cloud as input, produces a new point cloud by re-sampling from the underlying workspace surfaces. Moreover it does this in a manner which can be biased by local low-level geometric or appearance properties and higher level (semantic) classification of the surface. We are in no way prescriptive about what justifies a biasing in the re-sampling scheme — this is left up to the user who may encapsulate what constitutes "interesting" into one or more "policies" which are used to modulate the default re-sampling behavior.**

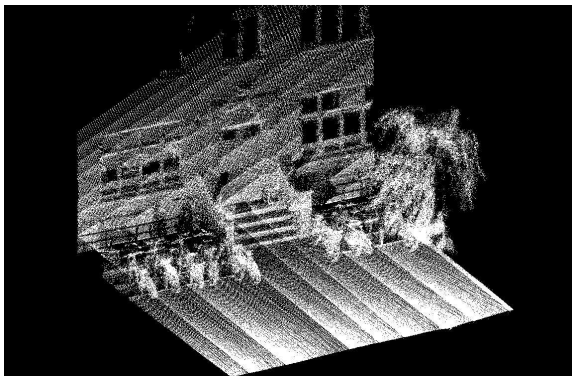

Fig. 1. An example of a large 3D point cloud generated with an outdoor mobile robot. In the foreground we can observe a rack of bicycles, to the right a large tree, and in the background a building with protruding staircase. Notice the large variation in local point density.

I. Introduction and Motivation

This paper is about scene representation using point clouds with application to mobile robot navigation. Sensors like laser range scanners and stereo pairs are becoming ubiquitous and finding substantial application to navigation and mapping tasks. They return sets of 3D points which are discrete samples of continuous surfaces in the workspace, which we shall refer to as 'point clouds'. There are two prominent and distinct schools of thought regarding how point clouds might be used to infer vehicle location and/or workspace structure. On one hand, segmentation and consensus techniques can be used to extract subsets of the data which support the existence of geometric primitives - planes, edges, splines or quadrics for example. Vehicle state inference is performed using the parameters of these primitives and the world is modelled as the agglomeration of geometric primitives. All raw data is subsequently disregarded. More recently the 'view based' approach has become very popular. Here the raw data is left untouched. Location estimation is achieved by perturbing discrete instances of the vehicle trajectory until the overlapping 'views' match in a maximum likelihood sense. The workspace estimate is then the union of all views rendered from the vehicle trajectory.

There is a stark contrast between these two approaches - one strives to explain measurements in terms of parameterizations resulting in terse, perhaps prescriptive representations, but which we can then reason with at higher levels. The other makes no attempt to explain the measurements, no data is thrown away and every data point is treated with equal importance. The ensuing work-space representations look visually rich but the maps are semantically bland.

This paper examines the ground between these two camps and contributes a framework by which raw point clouds can be used to generate new samples from the surfaces they implicitly represent in a selective way. We will refer to this as 'feature sensitive re-sampling'. The approach we describe here allows us to avoid fitting data to *a-priori* prescriptive models while still embracing the concept of a feature. Importantly it retains the ability of view based approaches to capture irregular aspects of the workspace as anonymous sets of points. The framework consists of a generalized re-sampling mechanism and a set of hooks to which any number of user-defined sampling 'policies' may be attached. The central idea is that the point cloud is a discrete sampling of continuous surfaces in the workspace. Depending on application (mapping, matching, localizing, indexing etc.), some parts of the surfaces in the workspace are more useful, interesting, salient, or descriptive than others. Such inclinations are entirely encapsulated in the individual sampling policies which have access to the local properties of the implicit workspace surfaces. For example they may preferentially select regions of a particular color, curvature or normal direction. In the light of this information each policy modulates the generalized re-sampling mechanism — biasing samples to originate from regions that, to it, are 'interesting'.

To make this abstraction more tangible we proceed with a few motivating examples. Imagine we are in possession of a substantial point cloud obtained from a 3D laser scanner like the one shown in Figure 1. Consider first the task of matching such a scan to another. There is an immediate concern that the workspace's surfaces have not been sampled uniformly - the fan-out of the laser and the oscillating nature of the device mean that close surfaces have more samples per unit area. Furthermore, surface patches close to the intersection of the workspace and the nodding axis also receive disproportionate representation. It would be advantageous to re-sample the data so that surfaces are sampled uniformly and that an ICP-like scan-matching process is not biased towards aligning regions of greater point density. Secondly, consider the case in which we are presented with both a 3D laser point cloud of a scene and color images of it from a camera - allowing the point cloud to be colored as shown in Figure 6a. A workspace region that may have appeared to be bland from a geometric perspective might now be marked with transitions in color space. We might wish to pay special attention to regions of a particular color, gradient, or neighborhood texture - the resulting data points might yield better registration results, provide discriminative indexes into appearance based databases or simply decimate the overall scene representation.

It is important to emphasize that the behavior of the framework can also be influenced by semantic information as well as low-level geometry and appearance. *A-priori* classification and segmentation of point cloud regions (glass, shrubbery or floor for example) can be used to inhibit existing policies as needs dictate. For example, a policy for maintaining regions of high curvature may be inhibited, if it is found to lie on a shrub, rather than a building - corners on a building being deemed more interesting than corners on a plant. Before proceeding, it is worth noting that the method we propose is different from that which simply sweeps through a point cloud and retains points in interesting regions. To see why, we must clarify what we mean by re-sampling. We do not draw new points from the measurements themselves but from an implicit representation of the workspace surfaces. If we constrained ourselves to re-sampling from the original point cloud we could not so easily address issues of under (or over) sampling. We would also be ignoring the fact that a single range point is just a sample from a surface - it is not an observation of an actual point in the workspace. In light of this it appears overly restrictive to limit ourselves to only working with the observed 3D points. Instead we work with the surfaces that they represent implicitly.

The rest of the paper is structured as follows: Section II begins by providing a brief summary of previous work. Section III follows, split into sub-section III-A, which provides a brief summary of the approach we have adopted and its relationship to existing literature, sub-sections III-B and III-C, which describe some of the key techniques in detail and sub-section III-D, which examines the overall framework. Section IV then shows some of the initial results obtained, and we conclude in Section V with some conclusions and final thoughts.

II. Previous Work

Over the last decade, there has been an enormous amount of high quality research in mobile robot navigation. Many papers have focused on building complex 3D environments using vision or laser scanned point clouds. However, relatively few have examined efficient representations that improve performance, or facilitate task accomplishment. One approach which has had considerable success extracts planar surfaces from the data - for example [2], [3], [9] and [10]. This has been taken a step further in [6], where a semantic net selects particular plane configurations, and forces them to conform to pre-conceived notions of how the world should look. This can correct planes corrupted by noise. However, many typical facades, surfaces and objects in realistic outdoor environments cannot be simplified using planes alone. In contrast, there has been a large amount of research in the computer graphics community on point cloud simplification, for example [7]. In the next section we take just one of these and adapt it into a context and feature sensitive re-sampling framework for mobile robotics.

III. Sampling Implicit Surfaces in 3D Point Clouds

A. Approach Summary

Let us assume we have a set of 3D points, lying on an unknown 2D manifold. The fundamental idea, a modified and extended version of Moenning and Dodgson's work in [5], is to approximate the manifold as the set of 3D grid cells which lie inside a union of ellipsoids, where an ellipsoid is fitted to each data point and a fixed number of its neighbors. This could be considered to represent a 'crust' over the data points, and is similar to the approach of Memoli and Sapiro in [4]. After randomly selecting a small number of 'seed' points, to initiate the process, we propagate fronts from each of their corresponding grid cells. This is performed using the Fast Marching Level Set Method [8] (described later in Section III-C), and continues until fronts collide or reach the edge of the problem domain. By storing the time at which a front arrived in each cell, we effectively generate a distance function over the cells considered (ie. the manifold). Furthermore, by looking for local maxima over this function, we are able to extract the vertices of a Voronoi diagram painted over the manifold. This enables us to select a location for a new re-sample point as prescribed by the Farthest Point Strategy in [1] (which is described further in Section III-B). After selection, this process can be repeated to find as many new points as required, or until the surface density reaches a predefined threshold. Note however, on subsequent iterations, it is not necessary to recompute the distance function (Voronoi diagram) over the entire problem domain. By maintaining front arrival times over all cells, only the front belonging to the most recently generated point needs propagating.

Another attractive and significant feature of this approach is that fronts can move at different speeds through different regions of the crust. These speeds can be found using the set of re-sampling 'policies' or user-defined hooks (described

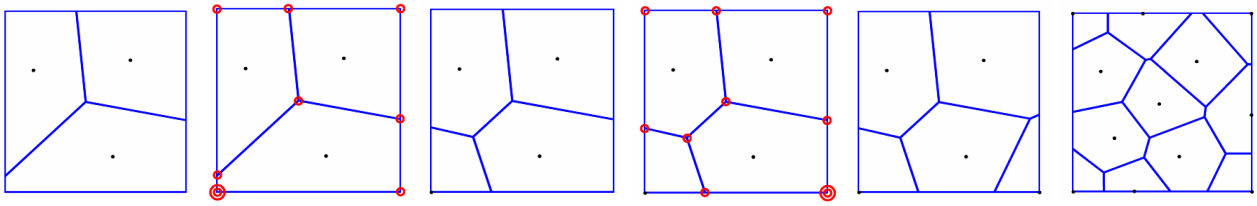

Fig. 2. Given a square image with three initial sample locations, one can construct a corresponding BVD, as shown in (a). Sub-figure (b) shows the results of a BVD vertex extraction (shown as circles), which are efficiently searched to yield the vertex with greatest distance to its closest point (circled twice). This forms the next iteration's sample point, which can be observed in the BVD in (c). Sub-figures (d) and (e) progress similarly. Sub-figure (f) shows the sample points generated after 9 iterations - note the distribution is becoming increasingly uniform.

earlier) which incorporate local surface attributes together with local class information. Consequently, when the generalized re-sampling mechanism (described above) comes to choose a new site, it extracts a vertex of a *weighted* Voronoi diagram, which is biased to be in the vicinity of a user-defined feature and/or class.

B. The Farthest Point Sampling Strategy

Farthest Point Sampling [1] was originally developed as a progressive method for approximating images. After initially selecting a set of 'seed' pixels, the idea is to select each subsequent 'sample' pixel to lie in the least known region of the image. This progressively generates high quality image approximations, *given* the current number of 'sample' pixels available. Furthermore, it can be shown that the next 'sample' pixel to select lies on a vertex of the previous iteration's Bounded Voronoi Diagram (BVD) - which provides an efficient means for future sample selection, as shown in [1]. This is illustrated in Figure 2. Given a square image with three initial sample locations, one can construct a corresponding BVD, as shown in (a). Sub-figure (b) shows the results of a BVD vertex extraction (shown as circles), which are efficiently searched to yield the vertex with greatest distance to its closest point (circled twice). This forms the next iteration's sample point, which can be observed in the BVD in (c). Sub-figures (d) and (e) progress similarly. Sub-figure (f) shows the sample points generated after 9 iterations - note the distribution is becoming increasingly uniform.

C. The Fast Marching Level Set Method

Level Set Methods [8] are efficient techniques for calculating the evolution of propagating fronts. This is achieved by considering a front Γ, moving normal to itself in $\mathbf{x}$, at time t, with speed function F, as the zeroth level set of a function $\Phi(\mathbf{x}, t)$ (ie. $\Phi(\mathbf{x}, t) = 0$). Given that the front's initial position is known, for the purpose of initialization, we can write the function as:

$$\Phi(\mathbf{x}, t = 0) = \pm d \tag{1}$$

where d is the distance from $\mathbf{x}$ to Γ and the plus or minus sign indicates whether the point is outside or inside the front

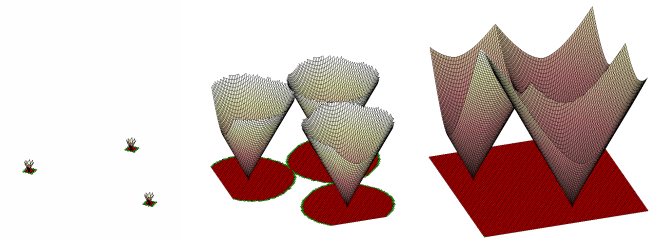

Fig. 3. These examples illustrate how the Fast Marching Level Set Method can be used to generate a 2D distance function. Sub-figure (a) shows 3 initial points with fronts propagating from each. The surface being generated above the plane represents each cell's arrival time, whilst the colors on the plane show the status of each cell. Red signifies 'dead', whilst green signifies 'active'. Sub-figure (b) shows how the front propagates over time, until the full distance function is generated in (c).

respectively. Assuming we observe a 'particle' at position $\mathbf{x}(t)$, which always lies on the propagating front, we can write:

$$\Phi(\mathbf{x}(t), t) = 0 \tag{2}$$

After differentiation, this becomes:

$$\frac{\partial \Phi(\mathbf{x}(t), t)}{\partial t} + \nabla \Phi(\mathbf{x}(t), t).\frac{\partial \mathbf{x}(t)}{\partial t} = 0 \tag{3}$$

Given that the function F gives the speed normal to the front, $F = \frac{\partial \mathbf{x}(t)}{\partial t}.\mathbf{n}$, and that $\mathbf{n} = \nabla\Phi/|\nabla\Phi|$, this can be simplified to:

$$\frac{\partial \Phi(\mathbf{x}(t), t)}{\partial t} + F|\nabla \Phi(\mathbf{x}(t), t)| = 0 \tag{4}$$

$$\text{given } \Phi(\mathbf{x}, t = 0) \tag{5}$$

Generalized Level Set Methods proceed by iterating over a grid in $(\mathbf{x}, \Phi)$ space to solve this initial value partial differential equation at each time step (ensuring that a careful approximation is made for the spatial gradient to generate a realistic weak solution). This implicitly yields the zeroth level set or propagating front for all times considered. As well as being straightforward to numerically solve, one of this technique's major advantages is that it can easily cope with sharp discontinuities or changes in front topology - as 'slices' of a smooth, higher dimensional volume.

As an example, consider a two dimensional, initially circular front. The initial three dimensional surface in $(\mathbf{x}, \Phi)$ space would form a cone. At each time step, after sufficient iterations, the solution would represent the cone moving and/or 'morphing' within this space. Simultaneously, the zeroth level set (or intersection with the plane, $\Phi(\mathbf{x}, t) = 0$) would show how the true front propagates, correctly modelling discontinuities or changes in front topology.

Unfortunately, given the dimension of a typical $(\mathbf{x}, \Phi)$ space and a reasonable temporal and spatial resolution, this can be computationally expensive. However, under the assumption that the speed function F is always positive, we can guarantee that the front passes each point in $\mathbf{x}$ once and once only (ie. Φ is single valued). This means that the Level Set Equation (Equation 4) can be expressed in terms of a front arrival time function $T(\mathbf{x})$, as an example of a stationary Eikonal equation:

$$|\nabla T(\mathbf{x})|F = 1 \tag{6}$$

This time, $T(\mathbf{x})$ need only be solved over a discrete grid in $\mathbf{x}$ (rather than $\mathbf{x}$ *and* Φ). Additionally, with careful choice of spatial gradient operator, updates can become 'one way' (rather than depending on *all* neighboring cells) - whilst still enforcing the realistic weak solution. This facilitates the efficient Fast Marching algorithm, which is only required to 'process' *each* grid cell once. In effect, it is able to calculate the front's arrival time, by working outwards over the discrete grid from its initial position.

If we now assume the front propagates in 3D (ie. $\mathbf{x} \in \mathbb{R}^3$) and substitute an expression for the spatial gradient operator, $\nabla T(\mathbf{x})$, into Equation 6, we can write the 3D Fast Marching Level Set Method's update equation as follows:

$$\begin{aligned}&[\max(\max(D^{-x}_{ijk}T, 0), -\min(D^{+x}_{ijk}T, 0))^2 \\ &+ \max(\max(D^{-y}_{ijk}T, 0), -\min(D^{+y}_{ijk}T, 0))^2 \\ &\quad + \max(\max(D^{-z}_{ijk}T, 0), -\min(D^{+z}_{ijk}T, 0))^2] = \tfrac{1}{F^2_{ijk}}\end{aligned} \tag{7}$$

Where i,j and k are grid cell indices along the x,y and z axes respectively, $T_{i,j,k}$ refers to the value of T at grid cell i, j, k and $D^{+x}_{ijk}T$ refers to the finite difference $+x$ gradient of T at grid cell i, j, k, which can be expressed as $D^{+x}_{ijk}T = (T_{i+1,j,k} - T_{i,j,k})/\Delta x$, where Δx is the unit grid cell length in the x direction. F_{ijk} is then the speed of wave propagation through grid cell i, j, k.

Before describing how to put this into practice, let us assume our domain is discretized into grid cells. We initialize the process by marking all cells which coincide with the front's initial position as 'active' and giving them a T value of zero (whilst all others are marked 'far', with T undefined). The front can then start marching forward, progressively calculating $T(\mathbf{x})$ for each 'active' cell's neighbors (except the 'dead' ones), using the update shown in Equation 7. If one of the neighboring cells is 'active', it will already possess a value for $T(\mathbf{x})$. In this instance, the smallest value of the existing and newly calculated $T(\mathbf{x})$ should be accepted, as it can be assumed this front reached that cell first. Conversely, if the neighboring cell is 'far', it should be made 'active' and assigned a value for $T(\mathbf{x})$. After all neighbors are dealt with, the processed 'active' cell should itself be considered 'dead'. It is important to note that when moving to process the next 'active' cell (ie. one on the current front) it is important to always process the one with a minimum value of $T(\mathbf{x})$ first. This ensures that any of the other 'active' cells could not possibly have had any influence on the processed cell's $T(\mathbf{x})$ (as they have greater arrival time values) - allowing (after its neighbors $T(\mathbf{x})$ values are calculated) it to be declared 'dead' (ie. fixed). This process continues until all cells are declared 'dead' and consequently all have $T(\mathbf{x})$ values. As an example, Figure 3 shows snapshots of this process propagating fronts on a plane. The full 3D technique is then formally described in Algorithm 1 where the fronts propagate over a grid $\mathbf{G}$ in which each cell $g(i, j, k)$ has the following properties: a front speed

Inputs : Grid $\mathbf{G}$, with cells $g(i, j, k)$, $i \in [1 : I]$, $j \in [1 : J]$ and $k \in [1 : K]$. Set of cells in initial front Γ.

Output: Arrival times: $g(i, j, k).T \quad \forall g(i, j, k) \in \mathbf{G}$

Initialization:
for $g(i, j, k) \in \Gamma$ **do**
  $g(i, j, k).active \leftarrow 1$;
  $g(i, j, k).T \leftarrow 0$;
  $\mathcal{H}$.PUSH($g(i, j, k)$);
end
while $\neg\mathcal{H}$*.empty()* **do**
  $g(i_{min}, j_{min}, k_{min}) \leftarrow \mathcal{H}$.POP();
  $\mathcal{N} \leftarrow$ NEIGHBOURHOOD($i_{min}, j_{min}, k_{min}$);
  for $g(i, j, k) \in \mathcal{N}$ **do**
    if $\exists g(i, j, k) \wedge \neg g(i, j, k).dead$ **then**
      $F_{ijk} \leftarrow g(i, j, k).F$;
      if $g(i, j, k).active$ **then**
        $\tau \leftarrow$ SOLVEPDE($F_{ijk}, \mathcal{N}$)
        $g(i, j, k).T \leftarrow \min(g(i, j, k).T, \tau)$;
        $\mathcal{H} \leftarrow$ HEAPIFY($\mathcal{H}$);
      else
        $g(i, j, k).Active \leftarrow 1$;
        $g(i, j, k).T \leftarrow$ SOLVEPDE($F_{ijk}, \mathcal{N}$);
        $\mathcal{H}$.PUSH($g(i, j, k)$)
      end
    end
  end
  $g(i_{min}, j_{min}, k_{min}).dead \leftarrow 1$;
end

Algorithm 1: Pseudo-code for an efficient implementation of the 3D fast marching algorithm for propagating fronts. The algorithm propagates a front Γ through a set of cells recording the time at which the front first passes through each of them. These times can be used to calculate the vertices of a bounded Voronoi diagram over the grid $\mathbf{G}$ which need not have homogenous propagation speeds.

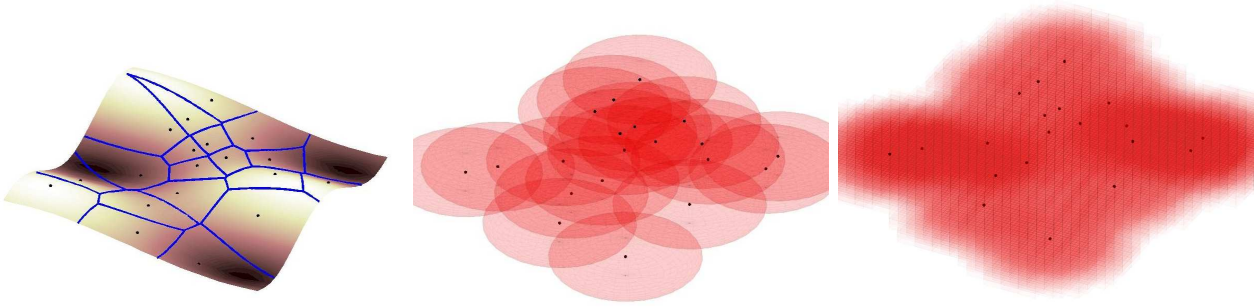

Fig. 4. Sub-figure (a) shows a 2D manifold, with a set of points on its surface. The blue lines represent surface Voronoi edges. This is what we aim to reproduce with our manifold approximation, to find where to place the next sample. Sub-figure (b) shows a series of spheres, with one centered on each original data point. If we take all 3D grid cells that lie inside this union, as in (c), we find an approximation to the original manifold.

field F, an arrival time T (initialized to ∞) and three status flags DEAD ACTIVE and FAR. The term $\mathcal{H}$ is a minheap of cells sorted by arrival time T. The function SOLVEPDE refers to solving $T_{i,j,k}$ using the discretized P.D.E. in Equation 7.

Whilst this technique has many uses, its primary function in this paper is to efficiently generate a Voronoi diagram over the original point cloud manifold, as illustrated in the example in Figure 4(a). This is implemented by approximating the manifold with a thin 'crust' of 3D grid cells. These can be found by locating spheres at each original point, as illustrated in Figure 4(b), and taking all those cells which lie within this union (as shown in Figure 4(c)). However, this approach tends to introduce large amounts of lateral noise — especially if regions of the true surface are truly planar. Consequently, we choose to fit ellipsoids to each point and its local neighborhood, and take the union of these instead. This approach seems to work particularly well, though some care does need taking in very sparse regions, when a fixed number of nearest neighbors span a large distance - and therefore produce excessively large ellipsoids. We can then take this bounded 3D space, initialize fronts in those cells which correspond to the set of points concerned, and propagate fronts outwards. This continues until all cells have been visited, when the resulting distance function can be processed to yield the Bounded Voronoi Diagram.

It is worth noting that this technique also allows efficient incremental construction of distance functions, when adding points to existing clouds. This can be achieved by propagating a front from each new point, until it reaches a region of the distance function which has values less than the front's predicted arrival time. Furthermore, by changing the speed at which fronts propagate through the discretized space, we are able to bias the positions of the Voronoi vertices. This will be examined in more detail in the next sub-section.

D. Imposing a Sampling Bias

Assume we have set of N_p data points, $P_{N_p} = \{p_1, p_2 .... p_{N_p}\}$, which are non-uniformly distributed samples of a 2D manifold M embedded in $\mathbb{R}^3$, each belonging to a single class of a total set of N_c classes, $C_{N_c} = \{c_1, c_2 .... c_{N_c}\}$, and each with a single associated vector of attributes, forming the total set of N_p attribute vectors, $A_{N_p} = \{a_1, a_2 .... a_{N_p}\}$. Our aim is to generate a new set of N_T points, $\tilde{P}_{N_T} = \{\tilde{p}_1, \tilde{p}_2 .... \tilde{p}_{N_T}\}$ with a local surface density at each original point proportional to a weight given by:

$$w_m = \mathcal{F}(a_m, \mathcal{C}(p_m)) \tag{8}$$

Where $\mathcal{F}$ expresses how interesting the neighborhood of p_m appears to be in the context of its geometry and appearance, a_m, and its semantic classification, c_m. This aim can be achieved by modifying Algorithm 1 in a straightforward way. The step $F_{ijk} \leftarrow g(i,j,k).F$ can be replaced with $F_{ijk} \leftarrow \mathcal{F}(a(g(i,j,k)), \mathcal{C}(g(i,j,k))$. Here we have slightly abused notation and used $a(g(i,j,k))$ to mean the attributes of the region surrounding the raw data point closest to cell $g(i,j,k)$ (and similarly for $\mathcal{C}(g(i,j,k))$). The upshot of this is that now the distance function calculated by the solution of the wave front P.D.E. will be biased (propagate slower) around interesting places. Because the wavefront determines the Voronoi vertices which in turn determine the new surface samples generated, the inclusion of Equation 8 in Algorithm 1 constitutes context and feature-sensitive sampling.

IV. RESULTS

Initial experiments have been performed using the data shown earlier in Figure 1, collected with a mobile robot. One of the key things to notice is the large variation in local point density. This is particularly evident on the floor, close to the vehicle, and on object faces sensed perpendicular to the scanning laser. The scene actually contains part of a building, a protruding staircase (which is sandwiched between two bicycle racks), and on the right, a large tree. Due to the nature of laser ranging - points are particularly dense around the tree's branches, as many mixed (multiple return) measurements were recorded. Figure 5a shows the same data, but includes semi-transparent colored and lettered control volumes for analysis. The number of points present is written adjacent to each (in a corresponding color), and is included in a bar chart (top right) for comparison. The total number of points in the entire cloud is shown bottom left.

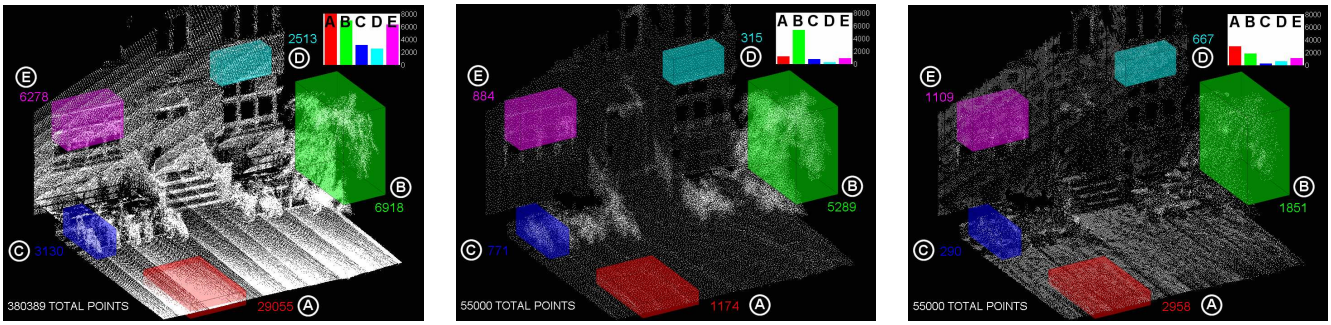

Fig. 5. Sub-figure (a) shows the original point cloud (as shown earlier in Figure 1), although this time includes semi-transparent colored and lettered control volumes for analysis. Sub-figure (b) shows the result of uniform re-sampling (maintaining a uniform weight over all cells), terminated when 55000 points had been produced. Sub-figure (c) shows the result of feature sensitive re-sampling based on modified 'normal variation', and was terminated at the same point.

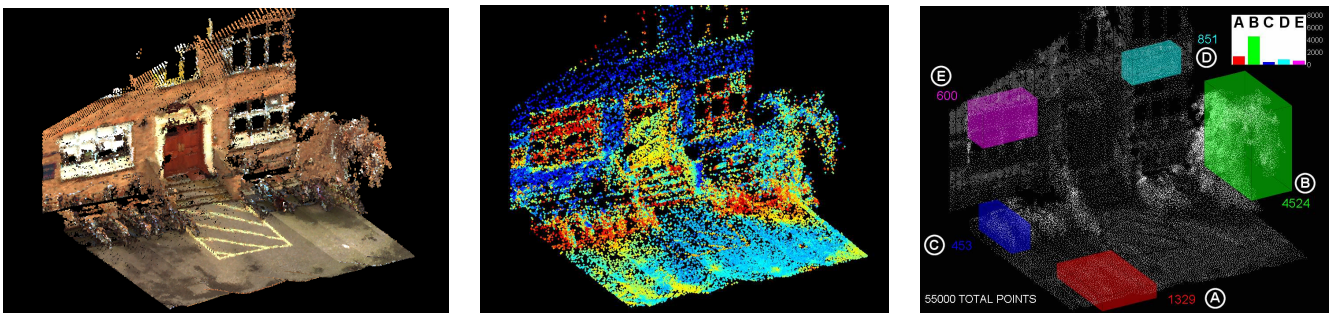

Fig. 6. Sub-figure (a) shows the result of projecting each laser point into an image of the scene and assigning it with color information. Sub-figure (b) shows a distribution of weights over the same cloud, proportional to a distance from each point (in RGB space) to a manually selected 'building' color. Sub-figure (c) then shows the result of feature sensitive re-sampling using these weights, preferentially choosing points of a particular shade.

The first experimental result is shown in Figure 5b, and later summarized in Table 1. This was produced using the re-sampling algorithm described earlier (maintaining a uniform weight over all cells), and was terminated when 55000 points had been produced. Notice that as a consequence of such weighting, the front propagated through all cells at the same speed, and results in an output which is a uniform representation of the implicit manifold. We also observe a 96% reduction in the number of points used to represent the red (A) control volume's section of floor. Similar re-sampling behavior throughout this cloud resulted in an overall representational overhead cut of 86% (from 380389 down to 55000), whilst preserving significant geometry.

The second experiment began with the same initial point cloud, and the resulting re-sampled point cloud is shown in Figure 5c. It is also summarized in Table 1. This shows that by weighting front propagation speeds through each grid cell, we can bias re-sample points to be in certain regions of the manifold. Whilst such user-defined 'policies' are generally functions of any number of local point attributes, here we make use of modified 'normal variation' alone. After generating a surface normal for each point (based on a close set of nearest neighbors), 'normal variation' can be defined as the variance of the angle between the point concerned and a fixed set of nearest neighbor surface normals. These are then transformed to ensure the distribution maps to a wide range of grid cell weights. If this is not performed, a few, very dominant points with very high 'normal variation' would tend to spread over a large range of the weights considered, leaving the majority of the grid cells with near identical propagation speeds. In this particular point cloud, the tree's foliage, the bicycles, the building's steps and its window frames all score highly. This leads to faster local front propagation and a tendency to inhibit nearby Voronoi vertices and the subsequent re-sample points. As in the last experiment, we stopped re-sampling when 55000 points were selected. Notice in particular the red (A) control volume located on the floor, and the green (B) control volume incorporating foliage, and contrast to the uniformly re-sampled cloud. The number of points in the predominantly planar red (A) control volume increased by 152% from 1174 to 2958, whilst those in the high curvature green (B) control volume decreased by 65% from 5289 to 1851.

The next series of experiments aims to highlight the versatility of our approach, as re-sampling policies can be based on any individual or set of local point cloud attributes. In this example, we choose to project each laser point into an image of the scene and assign it with color information as shown in Figure 6a. Figure 6b then shows a distribution of weights over the same cloud, proportional to a distance from each point (in RGB space) to a manually selected 'building' color. These

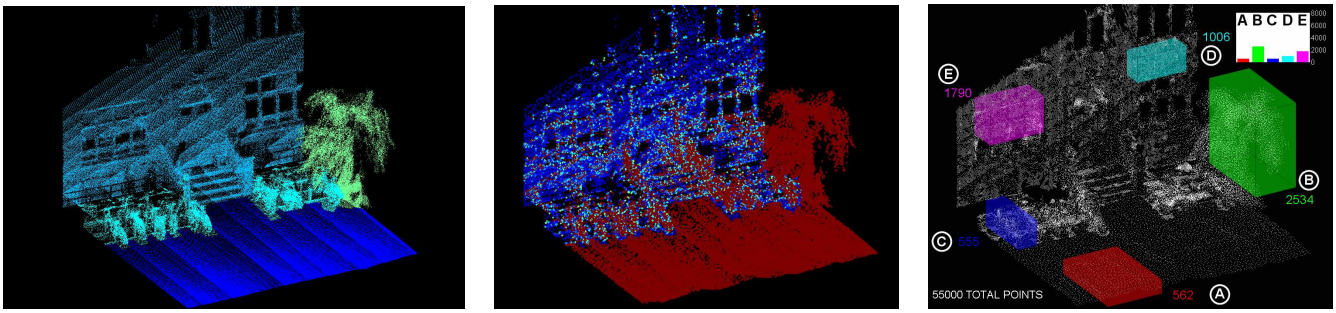

Fig. 7. Sub-figure (a) shows the point cloud from earlier, segmented into three distinct classes: 'floor', 'foliage' and 'other'. This was performed manually for the sake of demonstration. Sub-figure (b) then shows a weight distribution generated using local 'normal variation', along with the new class information. The intention here is to preferentially re-sample points on the building by artificially boosting the 'normal variation' weights of points classified as 'floor' or 'foliage'. Note that other 'non-building' points (on various objects) already have high 'normal variation' weights. Sub-figure (c) shows the result of context and feature sensitive re-sampling using these weights.

were then used to bias the generalized re-sampling mechanism, which preferentially chose points of this particular shade. Once again, re-sampling was stopped after 55000 points had been selected. The results of this process can be seen in 6c, and are summarized in Table 1. A careful examination of these results reveals some success in extracting points on the building - the number of points in the cyan (D) control volume, located on the building's wall, has increased 170% from 315 to 851 (compared to uniform re-sampling). Furthermore, the dramatic increase in the density of building points (to the left of the cyan (D) control volume, all the way across to the magenta (E) control volume) is clearly visibly, with a reduction in points on bicycles and foliage. However, a 13% increase in red (A) control volume points indicates that the color of the floor is too close to that of the building for this approach to disambiguate between the two. Additionally, we see a 32% decrease in blue (C) control volume points, as colors around the window vary dramatically compared to the main brickwork.

The third series of results demonstrates how the re-sampling process can be influenced by semantic information. Figure 7a shows the point cloud from earlier, segmented into three distinct classes: 'floor', 'foliage' and 'other'. This was performed manually for the sake of demonstration. Figure 7b shows a weight distribution generated using local 'normal variation' (described earlier), along with the new class information. The intention here is to preferentially re-sample points on the building by artificially boosting the 'normal variation' weights of points classified as 'floor' or 'foliage'. Note that other 'non-building' points (on various objects) already have high 'normal variation' weights. These were subsequently used to bias the re-sampling mechanism, which continued until the standard 55000 point target had been reached. The results generated can be seen in Figure 7c, and are also summarized in Table 1. Notice the massive increase in the number of points in the cyan (D) and magenta (E) control volumes (located on the building), when compared to the equivalent uniformly re-sampled cloud. The number of points in the cyan (D) control volume increased by 219% from 315 to 1006, whilst the number of points in the magenta (E) control volume increased by 102% from 884 to 1790. There was also a reduction in the number of points in all other control volumes (ie. those on the objects, foliage and the floor).

The final experiment highlights one use for this technique: re-sampling point clouds attached to vehicle poses in delayed state SLAM formulations. This could offer superior registration convergence properties, reduce overall processing time (if the point clouds are large or used frequently), reduce overall storage requirements, and allow efficient viewing of the maps constructed. Whilst this area does need significant further study, we include a few results to demonstrate this technique's potential. Figure 8a shows two point clouds (belonging to separate vehicle poses), each containing approximately 190,000 points. Figure 8b then shows the same point clouds uniformly re-sampled, each reduced to 50,000 points. Note that whilst we choose uniform weighting in this example (for the sake of clarity), there could be greater justification for another re-sampling policy. The pair of graphs in Figure 8c then compare the original pair of point clouds, and the re-sampled pair of point clouds when each was aligned with a typical registration algorithm. The top plot shows the normalized residual error at registration termination versus independent x, y and theta perturbations around the initial transformation estimate. Note that the re-sampled registration (shown in green) not only matches the performance of the original registration (shown in black) - the bounds of the convergence basin are actually increased. The lower plot shows corresponding processing times for the registrations, and it is apparent that with less points - the re-sampled registrations are significantly faster. Given that in this example the re-sampling process took 50 minutes (2998 seconds) for pose 1, and 23 minutes (1380 seconds) for pose 2, re-sampling would be beneficial if performing at least 6-8 registrations per point cloud (in terms of overall processing time alone). Note that these times are likely to reduce significantly when we engineer away the large time constant in our initial exploratory implementation (which scales $O(N \log N)$, where N is the number of grid cells, which is proportional to the workspace surface area - not the number of data points).

	Original	Uniform	Normals	Color	Color and Class
Entire Point Cloud	380389	55000	55000	55000	55000
Red Volume (A) (Floor)	29055	1174	2958	1329	562
Green Volume (B) (Foliage)	6918	5289	1851	4524	2534
Blue Volume (C) (Bicycle)	3130	771	290	453	555
Cyan Volume (D) (Building Wall)	2513	315	667	851	1006
Magenta Volume (E) (Building Window)	6278	884	1109	600	1790

TABLE I

EACH ENTRY IN THIS TABLE CORRESPONDS TO THE NUMBER OF POINTS IN AN ENTIRE POINT CLOUD, OR ONE OF THE CONTROL VOLUMES. THE COLUMN HEADINGS SIGNIFY WHICH EXPERIMENT PRODUCED EACH SET OF RESULTS. THE RED AND BLUE NUMBERS INDICATE WHETHER THE NUMBER OF POINTS INCREASED OR DECREASED RESPECTIVELY, COMPARED TO THE EQUIVALENT UNIFORMLY RE-SAMPLED POINT CLOUD.

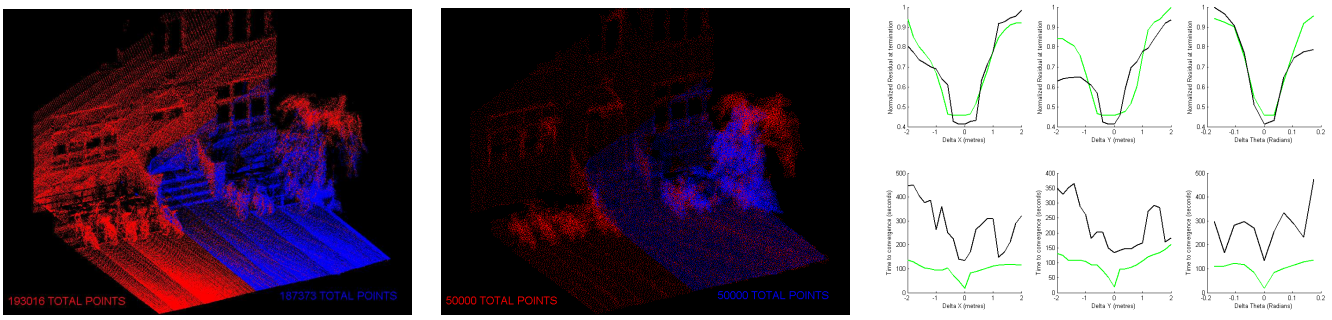

Fig. 8. Sub-figure (a) shows two point clouds (belonging to separate vehicle poses), each containing approximately 190,000 points. Sub-figure (b) shows the same point clouds uniformly re-sampled, each reduced to 50,000 points. Sub-figure (c) finishes with a pair of graphs that compare registration performance - first using the original pair of point clouds, and then using the re-sampled pair. The top plot shows the normalized residual error at registration termination versus independent x, y and theta perturbations around the initial transformation estimate, whilst the bottom one shows corresponding processing times.

V. CONCLUSIONS

In this paper we have described how one may use large 3D point clouds to generate samples from the underlying work-space surfaces. We have adopted the technique of solving a discretized P.D.E. to find a Voronoi diagram over a surface approximated by a thin crust of cells. An iterative scheme uses the Voronoi-vertices to generate new samples and update the Voronoi diagram. We have suggested in this paper that this approach has substantial value to the robotics community in terms of using 3D laser range data. By using the qualities of the original data and prior information regarding the type of surfaces being sampled to bias the solution of the governing P.D.E. we can mitigate sensor aliasing issues, select regions of interest and apply feature extraction algorithms all within a single framework. We have not yet fully explored the spectrum of uses for this technique in outdoor navigation (which is the domain which motivates this work) especially when using both laser and camera images. Nevertheless we feel it is a promising and elegant approach which comfortably occupies the middle ground between the feature-based (prescriptive) and view-based (indifferent) approaches to interpreting and using point clouds for navigation.

ACKNOWLEDGMENT

This work was supported by the Engineering and Physical Sciences Research Council Grant #GR/S62215/01.

REFERENCES

[1] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Trans. on Image Processing*, 6(9):1305 – 1315, 1997.

[2] D. Hähnel, W. Burgard, and S. Thrun. Learning compact 3D models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, 44(1):15–27, 2003.

[3] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun. Using EM to learn 3D models of indoor environments with mobile robots. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 329–336, San Francisco, CA, USA, 2001.

[4] F. Memoli and G. Sapiro. Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces: 730. *J. Comput. Phys.*, 173(2):764, 2001.

[5] C. Moenning and N. Dodgson. A new point cloud simplification algorithm. *The 3rd IASTED International Conference on Visualization, Imaging and Image Processing (VIIP)*, 2003.

[6] A. Nuchter, H. Surmann, and J. Hertzberg. Automatic model refinement for 3D reconstruction with mobile robots. *3DIM*, 00:394, 2003.

[7] M. Pauly, M. Gross, and L. P. Kobbelt. Efficient simplification of point-sampled surfaces. In *IEEE Visualization (VIS)*, Washington, DC, USA, 2002.

[8] J. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Nat. Acad. Sci.*, 93(4):1591–1595, 1996.

[9] J. Weingarten, G. Gruener, and R. Siegwart. Probabilistic plane fitting in 3D and an application to robotic mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 927–932, New Orleans, USA, 2004.

[10] D. F. Wolf, A. Howard, and G. S. Sukhatme. Towards geometric 3D mapping of outdoor environments using mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1258–1263, Edmonton Canada, 2005.