AtomMap: A Probabilistic Amorphous 3D Map Representation for Robotics and Surface Reconstruction

David Fridovich-Keil, Erik Nelson, and Avideh Zakhor

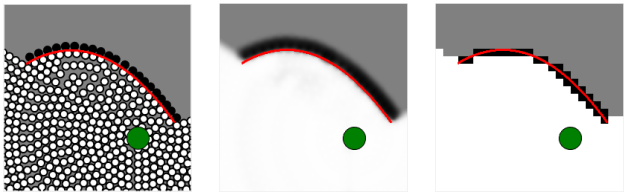
***Abstract*— We present a new 3D probabilistic occupancy map representation for robotics applications by relaxing the commonly-assumed constraint that space must be perfectly tessellated. We replace the regular structure of 3D grids with an unstructured collection of non-overlapping, equally-sized spheres, which we call "atoms." Abandoning the grid structure allows a more accurate representation of space directly tangent to surfaces, which facilitates a number of applications such as high fidelity surface reconstruction and surface-guided path planning. Maps composed of atoms can distinguish between free, occupied, and unknown space, support computationally efficient insertions and collision queries, provide free space planning guarantees, and achieve state-of-the-art memory efficiency over large volumes. This is achieved while simultaneously reducing quantization effects in the vicinity of surfaces and defining a useful implicit surface representation.**

I. Introduction

Many areas of interest in the field of mobile robotics rely on the ability to handle several core tasks, including localization and mapping, obstacle avoidance, motion planning, and exploration. Common among these tasks is the need to fuse sensor measurements together into a volumetric map which not only represents the raw range sensor data but also the volume carved out by that data.

For simplicity, 3D space is traditionally discretized and represented as a regular grid, where each voxel in the grid is independently considered either *occupied* or *free*, with occupancy probability estimated from range measurements. Formally, this data structure is called an *occupancy grid* [7]. The most obvious benefit of this cubic partition is tractability: rather than representing occupancy at each infinitesimal point in space, occupancy grids quantize the space into a finite and tunable number of regular voxels. This quantization, however, implies an important shortcoming: lacking the ability to place units of space arbitrarily, grid-type maps introduce significant errors near surfaces. In many robotics applications, e.g. obstacle avoidance, it is critically important to model the space near surfaces correctly.

In this paper we present an alternative representation, AtomMap, which replaces the regular grid of cube-shaped voxels with an unordered collection of non-overlapping, equally-sized spheres. Figure 1 demonstrates the AtomMap concept in 2D. In Fig. 1a an AtomMap is generated from simulated laser scans of a curved surface, emanating from a point source. Each sphere stores an estimate of its occupancy probability, which is interpolated across the space in Fig. 1b according to the method described in Sec. III-E. For comparison, Fig. 1c shows the result of inserting the same laser scan into an occupancy grid.



(a) (b) (c)

Fig. 1: AtomMaps (a) may be interpolated (b) to preserve the continuous nature of surfaces in the environment more accurately then grid-based map data structures (c).

Because they use arbitrary-depth k-d trees as their underlying data structure, AtomMaps enjoy memory efficiency commensurate with octree representations, and likewise support comparably efficient insertions and spatial queries. By construction, AtomMaps are able to represent surfaces more accurately than existing methods, promising to facilitate surface-based exploration strategies and high-quality renderings. Moreover, AtomMaps still support popular algorithms such as A^* path planning and abstractions such as signed distance fields [15], which are useful for a variety of graphics-related applications.

Our presentation of AtomMaps proceeds as follows. We begin with a survey of related representations from robotics and computer graphics in Sec. II. Section III introduces the AtomMap framework, detailing the usage of spherical atoms as a unit of construction, a probabilistic method for inserting information from range sensor measurements, and implementation details for achieving computational and memory efficiency. Section III-F presents an application of AtomMap to occupancy-based surface reconstruction. We then describe the optional addition of signed distance estimation within atoms in Sec. IV, and show how signed distance can be used to guide path planning in Sec. IV-A. We close with Sec. V, which presents timing and memory comparisons against a standard grid map representation [10], empirical map quality metrics, and visualizations of several large environments reconstructed as AtomMaps from collected LiDAR data.

All authors are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. D. Fridovich-Keil is supported by the NSF GRFP. The authors also acknowledge support from NSF grant no. CMMI-1427096
{dfk,eanelson,avz}@eecs.berkeley.edu

II. Related Work

Initially proposed in the late 1980s for use in 2D mapping with sonar [12, 7], occupancy grids have become the accepted standard in mobile robotics for mapping, exploration, and planning.

In two dimensions, it is possible to represent a large environment with a densely-allocated grid [8]. Although this approach has been demonstrated in three dimensions [18, 13], the maximum map size is severely limited by the available memory on standard computers used for mobile robotics applications. For example, in [13] the authors are only able to map out an area that is $6 \times 6 \times 2$ feet at roughly 0.3 inch resolution using a $256 \times 256 \times 64$ grid. Of course, Moore's law scaling has dramatically increased memory capacities since the introduction of occupancy grids; unfortunately, the memory usage of a dense 3D grid quickly becomes prohibitive even on modern computers.

There have been many attempts to improve the memory performance of occupancy grids in 3D. One of the first ideas to emerge was based on the 2.5D assumption, i.e. that the environment can be fully modeled by a floor plan labelled with surface heights above the ground plane. In [9] the authors design a legged extraterrestrial rover that represents the environment with an elevation map that stores exactly one value in each cell of a 2D grid that represents the height of the surface at that location. In [21] the authors extend the elevation map concept to account for the possibility of multiple surfaces above each 2D grid cell. This is similar to the idea behind multi-volume occupancy grids, proposed in [6]. The major disadvantage of such 2.5D representations is that they are poorly suited to irregular outdoor geometries, which do not always conform to the 2.5D assumption.

A major breakthrough in memory-efficient true 3D occupancy grid mapping came in the form of OctoMap, which partitions the 3D volume using an octree [10]. OctoMap demonstrated a dramatic memory improvement over existing methods, achieved by only allocating memory as needed at run-time and optionally by pruning the tree during operation to remove redundancy at the cost of speed.

One of the most important features of occupancy grids is that they distinguish between unknown and free space. The distinction is unnecessary for some applications, e.g. planning paths around obstacles, but it is critical for tasks that guide robots toward unexplored areas. For example, [23] uses a 2D dense occupancy grid to find *frontiers* – voxels in free space that border unexplored space – which are then used to guide autonomous exploration of an unknown scene. Increasingly popular information-theoretic approaches to autonomous exploration, e.g. [3, 4], also necessarily rely on the direct modeling of free, unknown, and occupied space. In particular, we draw attention to the approach taken in [24], in which a UAV navigates toward frontiers that lie *on surfaces in the voxel grid*. Such frontiers are subject to quantization error bounded by the maximum voxel dimension.

In addition, there is a rich mathematical theory of general lattices [5]. In [2] the authors show that the body-centered cubic lattice structure is an optimal quantizer in three dimensions, in the mean-squared sense under a uniform data distribution. Building on these results, in [20, 19] the authors investigate the use of non-cubic lattices specifically for occupancy mapping. In [20] the authors dismiss out of hand the possibility of using circular or spherical voxels since they cannot tessellate the space. In this paper, we solve this problem by revising the traditional formulation for probabilistic occupancy updates, as described in Sec. III-E.

To our knowledge, there is no existing work which uses spheres as the basis for an occupancy map structure. However, we do acknowledge that spheres have been used as the geometric building blocks for UAV path planning. In [22], the authors create a graph of connected overlapping spheres of radius equal to the distance to the nearest obstacle, and plan paths through this graph so that the vehicle is always contained in the "tunnels" between spheres. Our work is closely related, although we restrict spheres, which we call "atoms," to be non-overlapping and of uniform size, and we also compute explicit occupancy probability.

There is also a great deal of existing work related to signed distance fields (SDFs), both with application to surface modeling in computer graphics, and as a map representation for autonomous robotics. Generally, the Euclidean signed distance (ESDF) to the nearest surface is approximated on a grid, where the zero-crossings of the resulting scalar field represent the surface manifold. Various approximations are used: in particular, the Truncated SDF (TSDF) approximates the distance to the nearest surface *along the scan ray* rather than the distance to the nearest surface *in any direction*. The KinectFusion project [14] uses TSDFs to model surfaces using RGBD data, for example. Recent work [15] blends TSDFs and ESDFs for superior performance in on-line map-building and planning. Although the focus of this paper remains occupancy grid mapping, for completeness AtomMap does support a variant of the TSDF, as explained in Sec. IV.

III. Atom Mapping Framework

We begin in Sec. III-A by formulating the mathematical problem which AtomMaps are designed to solve. Section III-B then explains how AtomMaps use non-overlapping, equally-sized spheres as the fundamental unit of space.

Algorithm 1 provides an overview of how an AtomMap is generated. Upon receiving a sensor pose x and a scan z, the scan is transformed into world coordinates and surface normals are estimated at each scan point. Each scan point is then processed via several steps of raytracing described in Sec. III-C, which generates a set of candidate atoms to be inserted into the AtomMap. However, before insertion, these candidate atoms are first preprocessed by a temporary buffer in order to remove redundant information. Section III-D describes the mechanics of this preprocessing step in detail. Finally, the remaining candidate atoms are inserted into the AtomMap using a probabilistic update scheme described in Sec. III-E.

Algorithm 1 AtomMap Creation

```
map ← new AtomMap()
while map.incomplete() do
    x ← sensorPose()
    z ← newScan()
    z ← transformScan(z, x)
    n ← estimateNormals(z)
    buffer ← new Buffer()
    for all p_i, n_i ∈ (z, n) do
        atoms ← traceRays(x, p_i, n_i)
        buffer.insert(atoms)
    map.merge(buffer)
```

A. Problem Formulation

In this section, we present a formal description of the occupancy mapping problem.

Consider a robot moving through a continuous environment $\mathcal{E}$ along a discrete-time trajectory $(x_1, x_2, x_3, ..., x_n), x_i \in \mathcal{E}$. Each point in $\mathcal{E}$ is labelled by a function $\omega : \mathcal{E} \longrightarrow \{0, 1\}$ either "occupied," namely 1, or "free," namely 0. At each time index i, the robot receives sensor measurement z_i consisting of a set of points that lie on the boundary between free and occupied space in $\mathcal{E}$, i.e. on a surface. The occupancy mapping problem is to estimate the occupancy probability $\ell(p) \triangleq \mathbb{P}\{\omega(p) = 1 | x_{1:n}, z_{1:n}\}$ at all points p in the space.

Additionally, define a signed distance function $sdf : \mathcal{E} \longrightarrow \mathbb{R}$ which maps each point in the environment to a scalar whose sign represents whether the point is occupied or not, and whose magnitude corresponds to the Euclidean distance to the nearest surface.

In this work, we support separately estimating occupancy probability ℓ and signed distance field sdf, in "occupancy mode" and "signed distance mode" respectively.

B. Spherical Atoms as a Unit

Non-overlapping spherical "atoms" of fixed radius are the basic unit of our maps. These spheres are stored in a k-d tree, rather than an octree as is commonly used in voxel-grid representations. As with octrees, k-d trees are commonly used in computer graphics for efficiently indexing points in multidimensional spaces; one major difference is that k-d trees do not impose a regular cubic partition of space.

As noted in [20], one consequence of using non-overlapping spheres to represent $\mathcal{E}$ is that they are unable to partition the space fully. This does not pose a problem, however, since it is possible to model obstacles that lie entirely between atoms by a slight reformulation of the classical occupancy grid probabilistic update, as described in Sec. III-E.

Using spherical atoms is beneficial for two major reasons:

1) Spheres can be located *anywhere in space* and not just at lattice points, which makes it possible to model surfaces more precisely than can be done with a voxel grid.

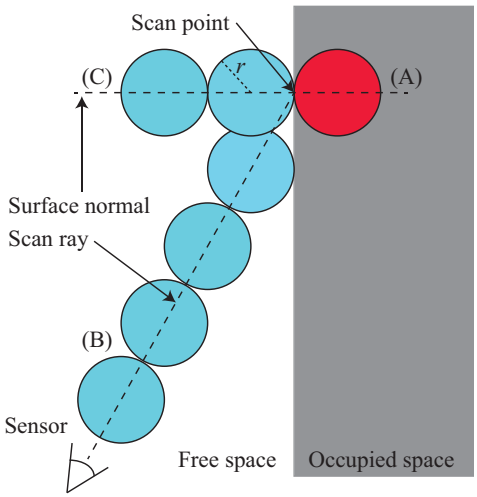


Fig. 2: Illustration of candidate atom generation for a single scan point.

2) AtomMaps do not prefer any particular directions over others: in voxel grids, the diagonals are sampled at a lower spatial frequency than the cardinal axes which reduces the effective resolution of the map.

Point 1 is critical, and bears repeating. The key contribution of this work is that we represent space with spherical units whose locations are determined *at run-time*, as dictated by each incoming scan. This is a sharp contrast with grid methods where space is represented by cubic units whose locations are determined *at compile-time*.

C. Candidate Atom Generation

Atoms are inserted along two different rays, either along the ray from the sensor to the scan point, or along the surface normal at the scan point. Raycasting is done in three stages, the last two of which contribute separately to estimating occupancy probability ℓ and signed distance sdf.

Figure 2 illustrates these three steps for a single scan point on a planar surface observed by the sensor along the line labelled "scan ray." The steps are enumerated below:

(A) One atom is generated behind and tangent to the surface. It is labelled "occupied," and colored red accordingly.

(B) In occupancy mode, atoms are generated along the ray from the scan point to the sensor and labelled "free." The first of these blue-colored atoms is constrained to be tangent to the surface.

(C) In signed distance mode, atoms are generated along the surface normal, starting from the scan point and extending up to a fixed, user-specified distance into free space. By construction, these atoms trace the local gradient of the signed distance field, which means that they provide a more accurate local estimate of sdf than candidate atoms generated in step (B) would.

Caution must be used, however, to avoid placing the atoms in step (C) too far from the scan point, as they may inadvertently intersect an as-yet unobserved surface in the environment. Also, note that in Fig. 2 an atom inserted in step

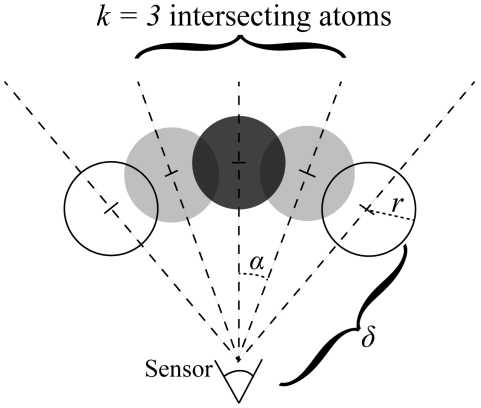


Fig. 3: Estimating k for the darkly-shaded atom. In this example, both of the lightly-shaded atoms intersect the darkly-shaded atom, so $k = 3$.

(C) intersects one inserted in step (B). In practice, however, only one of these steps is performed.

D. Preprocessing Candidate Atoms

In principle, each atom generated from a scan point could be incorporated immediately into the AtomMap. However, that would waste a significant amount of computational effort on processing the redundant information contained in overlapping atoms. Since range sensors acquire measurements radially during each scan, the resulting set of candidate atoms is most dense in the vicinity of the sensor: the space close to the sensor is thus highly oversampled.

When working with voxel grids, a common approximation is to allow each voxel in the map to be updated at most once by an incoming point cloud [10]. In an octree or voxel-grid representation, this can be enforced efficiently by spatial hashing; this does not work for atoms, however, whose locations are only determined at run-time. Therefore, AtomMap relaxes this strict constraint to a probabilistic statement: candidate atoms are pre-filtered so that existing atoms are updated approximately once per scan *on the average*. This is accomplished by randomly retaining each candidate atom a independently, with a probability $\gamma(a)$ that depends on its distance from the sensor, and discarding with probability $1-\gamma(a)$. Specifically, $\gamma(a)$ must be roughly equal to the reciprocal of $k(a) = \text{card}(\mathcal{S})$, where the set $\mathcal{S}$ contains a as well as all other candidate atoms that intersect a. Thus, on average exactly one of the atoms in $\mathcal{S}$ will remain to update the AtomMap.

As shown in Fig. 3, we can we can approximate k using the law of cosines for a particular radius-r atom a distance δ from a 2D radial sensor with angular resolution α.

$$k \approx 2 \left\lceil \frac{1}{\alpha} \arccos\left(1 - \frac{2r^2}{\delta^2}\right) \right\rceil + 1 \tag{1}$$

This k-value is only a lower bound in 3D; a tighter bound could be achieved for a specific sensor, given knowledge of its 3D angular sampling pattern.

For any given pair (k, γ), exactly one candidate atom in $\mathcal{S}$ will remain with probability $k\gamma(1-\gamma)^{k-1}$; however, the probability that none remain is $(1-\gamma)^k$. If none remain, then the AtomMap may have a hole at this location. In order to minimize this probability while still maximizing the probability of inserting exactly one atom, we solve the following regularized non-convex optimization problem parameterized by cost λ. The solution is derived by setting the derivative in γ to zero, and it is globally optimal for $k \geq 2$:

$$\gamma^* = \underset{0 \geq \gamma \geq 1}{\text{argmax}}\, k\gamma(1-\gamma)^{k-1} - \lambda(1-\gamma)^k = \frac{1+\lambda}{k+\lambda} \tag{2}$$

This process of computing an estimated $k(a)$ for each candidate atom a and then randomly discarding it with probability $1-\gamma(a)$ effectively randomizes and interleaves atoms from different rays – for this reason, we call it "stochastic angular interleaving." The parameter λ may be set anywhere in the interval $[0, \infty)$. As $\lambda \longrightarrow \infty$, $\gamma^* \longrightarrow 1$ converges to the no-interleaving case. We typically set $\lambda = 1$.

Further downsampling of the candidate atoms is also possible. One particularly simple method is to apply a standard voxel-grid filter to the candidate atoms that remain after stochastic angular interleaving. Although further downsampling is not strictly necessary, it is often convenient and may have the effect of improving computational performance at the cost of slightly decreased map quality.

E. Atom Insertion and Probabilistic Map Updates

This section describes the procedure for inserting new candidate atoms into the AtomMap, updating the estimated occupancy probability for existing atoms, and interpolating occupancy probability across the space.

While merging the buffer of candidate atoms generated by a single scan into the AtomMap, new atoms are inserted directly into the map if and only if they do not collide with any existing atoms. If there is a collision, e.g. between existing atom a and new atom b, we update the log-odds ratio $L(a) \triangleq \log\left(\ell(a)/(1-\ell(a))\right)$ of occupancy of the existing atom weighted according to its fractional overlap $w(a,b)$ with the new atom. An atom's log-odds ratio of occupancy is positive if it is most likely to be occupied, negative if it is most likely to be free, and zero otherwise.

$$L(a) \longleftarrow L(a) + w(a,b)L(b) \tag{3}$$

The "overlap fraction" $w(a,b)$ is defined to be the fraction of atom a that is included in atom b, and is calculated in closed form as follows where r is the atomic radius and $d \in [0, 2r]$ is the distance between the centers of the two atoms.

$$w = 1 - \frac{d}{4r^3}\left(3r^2 - \frac{d^2}{4}\right) \in [0,1] \tag{4}$$

Traditionally, log-odds updates are defined exactly as in Eq. 3 except that there is no overlap term scaling the update value [7]. Incorporating $w(a,b)$ has the effect of "trusting" atoms b that share a large amount of volume with atom a more than those which share only a small volume.

The notion of scaling the log-odds ratio L according to an overlap term w suggests a simple means of interpolating log-odds across $\mathcal{E}$. For any point $p \in \mathcal{E}$, we estimate $L(p)$ as the average of the log-odds values at all nearby atoms $a \in \mathcal{A} \triangleq \{a : \|\text{center}(a) - p\|_2 < 2r\}$, weighted according to $w(a,p)$, the overlap fraction of atom a with a hypothetical atom centered at p:

$$L(p) = \frac{\sum_{a\in\mathcal{A}} w(a,p)L(a)}{\sum_{a\in\mathcal{A}} w(a,p)} \tag{5}$$

Interpolating in this manner, AtomMap is able to solve the occupancy mapping problem posed in Sec. III-A far more accurately than grid-based methods. Results are shown in Sec. V-C.

F. Surface Reconstruction

In most mobile robotics applications, it is not necessary to represent surfaces explicitly; e.g. in Sec. IV-A, path planning is done on an implicit graph of connected atoms. However, there are applications such as virtual reality for which a surface representation is useful. In a standard occupancy grid, surfaces are the boundary between occupied and free grid cells, i.e. the zero-isocontour of log-odds L. Although in principle one could interpolate L across a voxel grid, e.g. by trilinear interpolation, in practice such "implicit surfaces" are often computed from signed distance fields, e.g. [14]. Although AtomMap does support signed distance estimation as discussed in Sec. IV, here we present an occupancy-based implicit surface reconstruction technique. We emphasize that this technique is not intended to compete with or replace traditional SDF-based methods. Our intent is rather to illustrate the main contribution of AtomMap, namely that placing units of space tangent to surfaces at run-time is a precise representation of underlying continuous geometry.

Equation 5 in Sec. III-E presents a method for efficient multilinear interpolation of log-odds ratios across the space. In principle, this is sufficient to compute an implicit surface; in practice however, it is too sensitive to noise in the locations of atoms to yield a visually realistic surface. In this section, we propose a Gaussian Process (GP) Regression-based interpolation technique specifically for generating noise-robust implicit surfaces from occupancy data. This idea is similar in spirit to approaches such as [16] and [11], the major difference being that in our work, the GP is trained on atoms.

We interpolate the log-odds ratio of occupancy at an arbitrary point in space by assuming that log-odds is a GP. Following [17], we define "training" points $\mathbf{x}$ and "query" points $\mathbf{x}^*$ with respective log-odds ratios $\mathbf{L}$ and $\mathbf{L}^*$, and kernel function $\kappa : \mathbb{R}^3 \times \mathbb{R}^3 \longrightarrow \mathbb{R}_+$.

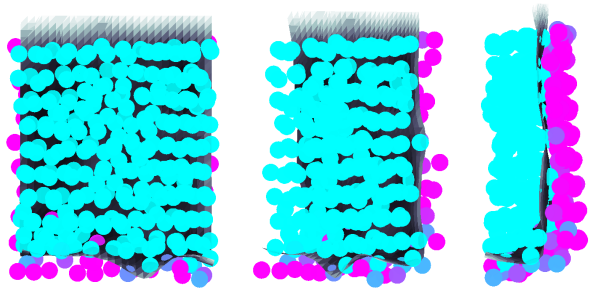
$$\begin{bmatrix}\mathbf{L}\\ \mathbf{L}^*\end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}\Sigma_{xx} & \Sigma_{xx^*}\\ \Sigma_{x^*x} & \Sigma_{x^*x^*}\end{bmatrix}\right) \tag{6}$$

$$\text{where } [\Sigma_{xx}]_{ij} \triangleq \kappa(x_i, x_j) \tag{7}$$

Following [17], we use a radial basis function as the kernel, i.e. $\kappa(x_i, x_j) = \exp(-\alpha\|x_i - x_j\|_2^2)$. The resulting regression problem is solved using Gaussian conditioning. Note that we have assumed that the training data is corrupted by isotropic noise of variance σ_n^2 for the sake of robustness.

$$\begin{aligned}\mathbf{L}^*|\mathbf{L} \sim \mathcal{N}(&\Sigma_{x^*x}(\Sigma_{xx} + \sigma_n^2 I)^{-1}\mathbf{L},\\ &\Sigma_{x^*x^*} - \Sigma_{x^*x}(\Sigma_{xx} + \sigma_n^2 I)^{-1}\Sigma_{xx^*})\end{aligned} \tag{8}$$

In practice, AtomMap estimates L at a each query point x^* by training on the N nearest atoms, centered at $\mathbf{x}$. Results are shown in Fig. 4, where the query points $\mathbf{x}^*$ lie on a grid of resolution 0.25 m, which is smaller than the 0.3 m atomic radius. The surface is visualized by meshing the zero-isocontour of the grid using the marching cubes algorithm. The estimated surface mesh is colored according to the variance output in Eq. 8. As shown, the reconstructed surface effectively partitions atoms which are measured to be "occupied" (red) and "free" (blue). The figure was generated by setting $N = 20$, noise variance $\sigma_n^2 = 2.0$ and length scale $\alpha = 1$.



(a) (b) (c)

Fig. 4: Three views (a) front, (b) oblique, and (c) side of an implicit surface, colored according to variance where darker means lower variance. Atoms are colored according to estimated occupancy, from blue (free) to red (occupied).

IV. Signed Distance Field

In this section, we discuss the signed distance field *sdf* optionally maintained by the AtomMap, and present an application to surface-guided path planning (Sec. IV-A).

We assume that the robot operates inside of a closed volume as in [24], and also adopt the convention that the signed distance field is positive inside the bounding surface, negative outside, and zero on the surface. In order to estimate *sdf* on-line, each candidate atom stores the distance between itself and the scan point which generated it, measured along the local surface normal as described in Sec. III-C, steps (A) and (C).

When inserting candidate atoms into the AtomMap, it is necessary to fuse *sdf* estimates from overlapping atoms. In the absence of noise, the best way to fuse these estimates would be to keep the value of minimum magnitude, since by definition the *sdf* represents the distance to the *nearest* surface. In the presence of noise, however, some smoothing is required. An empirically satisfactory heuristic is to compute a weighted average of estimated signed distance values

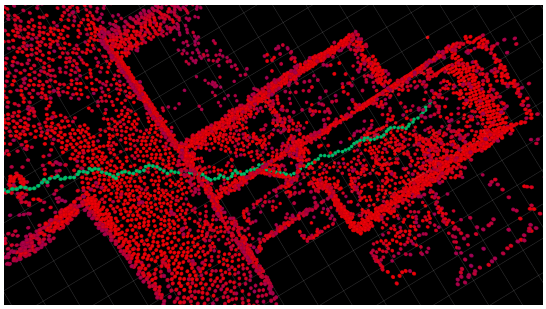


Fig. 5: Part of an A^* shortest path in green overlaid on the occupied subset of the AtomMap, in red.

$sdf(a)$ and $sdf(b)$ for existing atom a and overlapping atom b, where the weights are larger for sdf-values of smaller magnitude. Moreover, since repeated updates should decrease the uncertainty of estimate $sdf(a)$, each atom also stores and updates a measure of that uncertainty, σ^2, which may be interpreted as the approximate variance of sdf. The equations are given below, and parameterized by $\xi > 0$ for numerical stability. In practice, we find that $\xi = 0.5$ is satisfactory.

$$\sigma^2(b) = \frac{sdf(b)^2}{\xi + w(a,b)}, \xi > 0 \tag{9}$$

$$sdf(a) \longleftarrow sdf(a) + \frac{\sigma^2(a)}{\sigma^2(b) + \sigma^2(a)}(sdf(b) - sdf(a)) \tag{10}$$

$$\sigma^2(a) \longleftarrow \sigma^2(a)\left(1 - \frac{\sigma^2(a)}{\sigma^2(b) + \sigma^2(a)}\right) \tag{11}$$

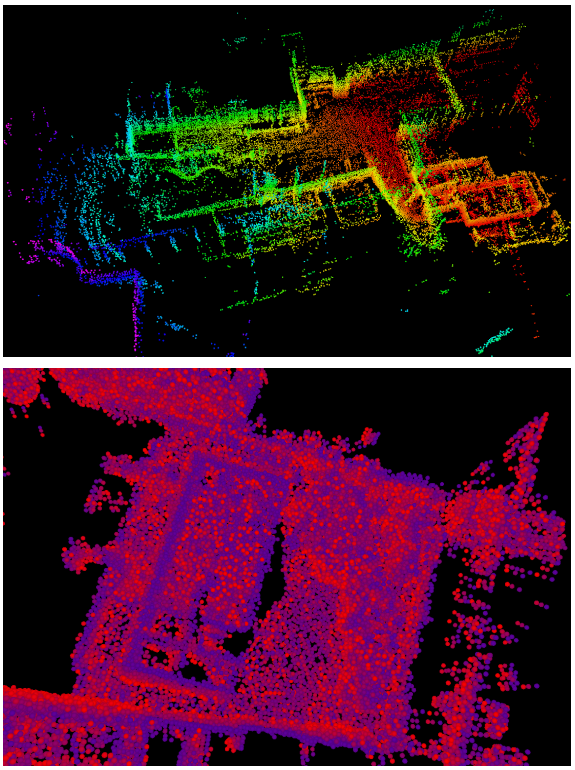
Note that $\sigma^2(b)$ is initialized to be inversely proportional to the overlap fraction $w(a,b)$ defined in Sec. III-E. This has the effect of not only "trusting" smaller measurements more than larger ones as is optimal in the noiseless case, but also trusting closer atoms more than distant ones.

A. Surface Guided Path Planning

One of the benefits of voxel grids is that adjacent voxels are connected spatially, and therefore define an implicit graph. This is useful for tasks such as path planning, which is often done with algorithms such as A^* search and Djikstra's shortest path. In a similar vein, we define an implicit graphical structure on the AtomMap by considering an atom to be connected to its free neighbors within a fixed radius.

Figure 5 shows the A^* shortest path between two arbitrary points in space, traversing the implicit graph of an AtomMap where free-space atoms are considered "connected" if their centers lie within four atomic radii.

A^* search makes use of a distance measure which by design must underestimate optimal path lengths. Traditionally, this measure is the total path length up to a particular point plus the Euclidean distance between that point and the goal. However, our on-line computation of sdf allows us to regularize such distance measures so that they penalize paths which do not remain a fixed distance away from the nearest surface, i.e. on a level set of sdf.



(a)

(b)

Fig. 6: AtomMap colored according to height in (a) (CMU dataset) and signed distance to the nearest surface in (b) (LBNL dataset). In each case, we only show the subset of atoms either occupied (a) or within ~ 1 m of a surface (b). Atomic radius is 0.25 m.

We implement this "surface-guided planning" scheme by setting the distance measure $D(a,b,g,q)$ as follows, for a path originating at atom a and passing through atom b, moving toward goal atom g along the level set of sdf defined by a distance q from the nearest surface. μ is an arbitrary non-negative regularization parameter.

$$D(a,b,g,q) = L(a \to b) + \|g - \text{center}(b)\|_2 + \mu||sdf(b)| - q| \tag{12}$$

where $L(a \to b)$ is the accumulated path length for a path with endpoints at atoms a and b. Increasing the value of μ forces the planner to choose paths that stay closer to the desired level set, at the cost of increasing path length.

V. Experimental Results

A. Implementation

Atom Mapping is implemented as a package for Robot Operating System (ROS) [1], written in C++.[1] Figure 6 shows two examples of full AtomMaps generated from real data collected using a Velodyne VLP-16 LiDAR sensor, where we only show atoms near surfaces for clarity.

[1] https://github.com/ucberkeley-vip/atom_mapping

B. Speed and Memory Usage

We measure the efficiency of our implementation by recording both total memory consumption and average processing time per scan, and compare against OctoMap.[2] For a fair comparison, we set OctoMap's voxel resolution such that each voxel contains the same volume as each atom – i.e. voxel side length is $r(4\pi/3)^{1/3}$, where r is the atomic radius. The experiment is run on two datasets acquired with a Velodyne VLP-16 LiDAR sensor, each evaluated at an atomic radius of 0.5 and of 1.0 m. All tests were performed on a 2011 MacBook Pro with 8 GB of RAM and a 2.0 GHz quad-core Intel Core i7 processor. For a meaningful comparison, we turn off octree pruning in OctoMap and we record separate results for AtomMap in occupancy and signed distance modes. We set OctoMap's maximum tree depth to 16, and assume a maximum sensor range of 25 m beyond which measurements are discarded. Additionally, we apply stochastic angular interleaving as in Sec. III-D with $\lambda = 1$ and a postprocessing grid-filter of resolution equal to one atomic diameter. Table I summarizes these results.

TABLE I: Empirical Memory Usage and Processing Time per Scan for AtomMap (AM) and OctoMap (OM)

Dataset	CMU		LBNL	
Map dimensions (m)	$168 \times 90 \times 35$		$68 \times 65 \times 17$	
Atomic Radius	0.5	1.0	0.5	1.0
OM mem. (MB)	26.4	18.0	20.2	12.1
AM mem. occ mode (MB)	32.3	15.7	18.9	12.1
AM mem. sdf mode (MB)	24.4	13.9	16.8	11.6
OM time (ms)	34.7	10.5	13.2	3.6
AM time occ mode (ms)	56.2	14.4	20.0	5.7
AM time sdf mode (ms)	28.8	10.1	13.0	4.3

As shown, AtomMap uses less memory than OctoMap in almost all trials, while processing scans either slightly faster (in signed distance mode) or slower (in occupancy mode). Both occupancy and signed distance modes are still real-time for reasonable frame rates, i.e. on the order of 10-20 Hz. Predictably, processing time increases as the atomic radius decreases – this is a consequence of the increasing number of atoms and hence k-d tree depth. Interestingly, however, we note that our implementation performs significantly better on both metrics in signed distance mode compared to occupancy mode. This may be explained by referring back to Sec. III-C, where it is stated that, in signed distance mode, atoms are only inserted along the normal direction as in steps (A) and (C), *not* along the scan ray as in step (B). As the scan ray is typically much longer than the distance traced along the normal vector in signed distance mode (~ 2 m in these experiments), the map tends to include fewer atoms in signed distance mode, leading to greater efficiency.

We emphasize that both OctoMap and AtomMap are sufficiently fast and memory-efficient that they can run in real-time on large datasets, at reasonable resolution. The difference is that AtomMap represents the environment with far greater precision, as shown in Figs. 1, 7, 8, and 9.

[2]Downloaded from `http://wiki.ros.org/octomap_server` on Feb. 11, 2016.

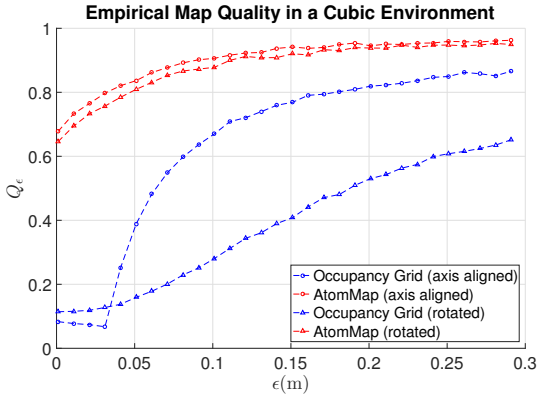


Fig. 7: Comparison of map quality in the vicinity of surfaces, for rotated and axis-aligned cubic environments.

C. Map Quality

It is difficult if not impossible to measure metric accuracy of map representations such as AtomMap on real data, without ground truth. In this section, we characterize the performance of AtomMap in several small simulated environments. Results are shown in Fig. 7 for a cube-shaped environment of side-length 2.0 m (both axis-aligned and rotated), and in Fig. 8 for a spherical environment of radius 1.0 m. In all experiments, atoms are of radius 0.1 m and voxels' volume is the same as that of each atom.

For any closed environment of the kind described in Sec. IV, we define the quantity V_ϵ as the volume of the region of $\mathcal{E}$ that lies within a distance ϵ of the surface on the same side as the sensor, i.e. the interior. We estimate this quantity using Monte Carlo integration and divide by the true value to compute the following quality metric:

$$Q_\epsilon \triangleq \hat{V}_\epsilon / V_\epsilon \tag{13}$$

Intuitively, Q_ϵ measures the fraction of the space near the surface which is correctly classified as free space. In the limit $\epsilon \longrightarrow 0$, Q_ϵ may be interpreted as a direct measure of surface quality. For larger ϵ, Q_ϵ contains information about space slightly farther from the surface, which is relevant for some robotics applications, e.g. path planning in the vicinity of surfaces.

As shown in Figs. 7 and 8, AtomMap outperforms the voxel grid in all of the simulated environments by a significant margin. The comparison is particularly striking for the axis-aligned cubic environment, as we expect the voxel grid to be particularly well-suited to that geometry.

Additionally, we wish to highlight a major qualitative difference between AtomMap and voxel-based methods which occurs when processing surfaces viewed at oblique angles, such as the ground. Observe the difference in Fig. 9, where the AtomMap accurately samples the ground, and the voxel grid has large, regularly spaced holes. These holes occur because voxels initially labelled *occupied* are later updated to be *free* by rays that intersect the surface in an adjacent voxel farther from the sensor. The holes occur in long columns because the ground is sloped. AtomMap solves this

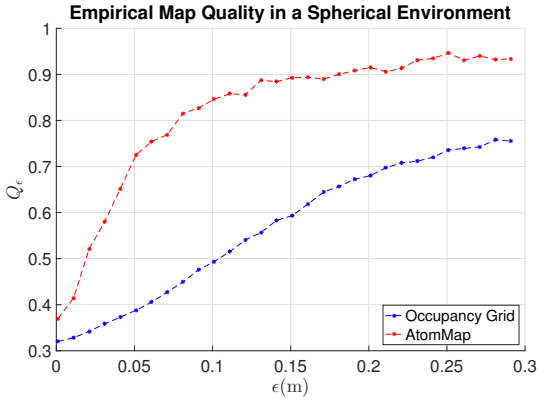


Fig. 8: Comparison of map quality in the vicinity of surfaces, for a spherical environment.

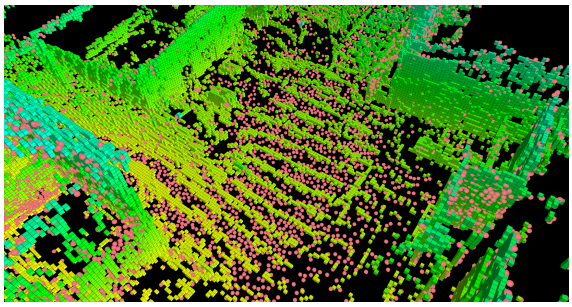


Fig. 9: Overlaid OctoMap (multi-colored, resolution 0.4 m) and AtomMap (red, atomic radius 0.25 m). AtomMap reconstructs the ground more faithfully than the voxel grid.

problem by placing atoms tangent to surfaces, which grid-based methods are fundamentally incapable of doing.

VI. Conclusion

We proposed a novel occupancy map representation for use in mobile robotics applications. Instead of relying on a regular, cubic voxel grid, we store an unordered collection of non-overlapping spheres. By doing so, we are able to leverage surface normal information and place spheres directly tangent to surfaces, leading to a dramatic increase in map quality. Additionally, AtomMap supports a separate signed distance field mode, which may be useful for applications such as surface-guided path planning. This is all achieved in run-time and memory usage roughly comparable to the current state of the art.

References

[1] URL: `http://www.ros.org`.

[2] E. S. Barnes and N. J. A. Sloane. "The optimal lattice quantizer in three dimensions". In: *SIAM Journal on Algebraic Discrete Methods* 4.1 (1983), pp. 30–41.

[3] F. Bourgault et al. "Information based adaptive robotic exploration". In: *Proc. IEEE/RSJ Intl. Conf. on Intelli. Robots and Systs.* Vol. 1. Lausanne, Switzerland, 2002, pp. 540–545.

[4] B. Charrow et al. "Information-Theoretic Mapping Using Cauchy-Schwarz Quadratic Mutual Information". In: *Proc. IEEE Intl. Conf. on Robotics and Autom.* Seattle, USA, 2015.

[5] J. H. Conway and N. J. A. Sloan. *Sphere packings, lattices and groups*. 3rd ed. New York, NY: Springer, 1993.

[6] I. Dryanovski, W. Morris, and J. Xiao. "Multi-volume occupancy grids: An efficient probabilistic 3D mapping model for micro aerial vehicles". In: *Proc. IEEE/RSJ Intl. Conf. on Intelli. Robots and Systs.* Oct. 2010, pp. 1553–1559.

[7] Alberto Elfes. "Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation". PhD thesis. Pittsburgh, PA, USA, 1989.

[8] Alberto Elfes. "Using occupancy grids for mobile robot perception and navigation". In: *Computer* 22.6 (June 1989), pp. 46–57.

[9] M. Herbert et al. "Terrain mapping for a roving planetary explorer". In: *Proc. IEEE Intl. Conf. on Robotics and Autom.* May 1989, pp. 997–1002.

[10] A. Hornung et al. "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees". In: *Auton. Robots* 34.3 (2013), pp. 189–206.

[11] Soohwan Kim and Jonghyuk Kim. "Continuous occupancy maps using overlapping local gaussian processes". In: *Proc. IEEE/RSJ Intl. Conf. on Intelli. Robots and Systs.* IEEE. 2013, pp. 4709–4714.

[12] L. Matthies and A. Elfes. "Integration of sonar and stereo range data using a grid-based representation". In: *Proc. IEEE Intl. Conf. on Robotics and Autom.* Apr. 1988, pp. 727–733.

[13] H. Moravec. "Robot spatial perception by stereoscopic vision and 3d evidence grids". In: *Perception* (Sept. 1996).

[14] Richard A Newcombe et al. "KinectFusion: Real-time dense surface mapping and tracking". In: *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE. 2011, pp. 127–136.

[15] H. Oleynikova et al. "Signed Distance Fields: A Natural Representation for Both Mapping and Planning". In: *Robotics: Science and Systems Workshop* (2016).

[16] Simon T O'Callaghan and Fabio T Ramos. "Gaussian process occupancy maps". In: *Intl. J. Robotics Res.* 31.1 (2012), pp. 42–62.

[17] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[18] Y. Roth-Tabak and R. Jain. "Building an environment model using depth information". In: *Computer* 22.6 (June 1989), pp. 85–90.

[19] J. Ryde and M. Brünig. "Lattice occupied voxel lists for representation of spatial occupancy". In: *Proc. IEEE/RSJ Intl. Conf. on Intelli. Robots and Systs.* Oct. 2010, pp. 567–572.

[20] J. Ryde and M. Brünig. "Non-cubic occupied voxel lists for robot maps". In: *Proc. IEEE/RSJ Intl. Conf. on Intelli. Robots and Systs.* 2009, pp. 4771–4776.

[21] R. Triebel, P. Pfaff, and W. Burgard. "Multi-Level Surface Maps for Outdoor Terrain Mapping and Loop Closing". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2006, pp. 2276–2282.

[22] Nicolas Vandapel, James Kuffner, and Omead Amidi. "Planning 3-d path networks in unstructured environments". In: *Proc. IEEE Intl. Conf. on Robotics and Autom.* 2005, pp. 4624–4629.

[23] B Yamauchi. "A frontier-based approach for autonomous exploration". In: *Proc. IEEE Sym. on Comput. Intelli. in Robot. and Autom.* Monterey, USA, 1997, pp. 146–151.

[24] L. Yoder and S. Scherer. "Autonomous Exploration for Infrastructure Modeling with a Micro Aerial Vehicle". In: *Field and Service Robotics*. 2015.