Sparse Surface Modeling with Curved Patches

Dimitrios Kanoulas and Marsette Vona

Abstract— **Traditional segmentation algorithms for range images create partitions of connected and non-overlapping—but potentially irregularly shaped—regions corresponding to world surfaces. This paper presents an alternative paradigm based on regularly shaped curved patches (paraboloids) that model local contact regions potentially compatible with e.g. a robot's toe, heel, or fingertip. These patches randomly sample the environment surface but are not required to strictly partition it. They are fit to neighborhoods of the range data and then validated for fit quality and fidelity to the actual data—extrapolations (like hole-filling) which are not directly supported by data are avoided. Two different neighborhood formation methods based on k-d tree and triangle mesh data structures are compared, and results are presented for 10 datasets taken in natural rocky terrain.**

I. INTRODUCTION

Some of the most challenging open problems in robotics are those which will require reliable contact with unstructured world surfaces, for example, walking on natural rocky terrain or climbing in a pile of rubble [1]. Recent advancements often either assume significant structure in the environment (like humanoid walking on flat ground [2]) or that irregularities can be tolerated by low-level feedback control during contact (as in BigDog [3]). 3D perception is sometimes used for obstacle detection or coarse traversability prediction (as in the Mars Exploration Rovers (MER) [4]). However, to enable rough-terrain walking and climbing, a perception system that can spatially model and finely quantify potential 3D contact surfaces may be needed.

Paraboloid surface patches with regular boundaries and one or two curvatures can be the basis for such a system: they can be applied to unstructured scenes, they are potentially easier to reason about than polygon mesh, continuous height-field models (e.g. [5]), or multiply-curved NURBS[1] patches (e.g. [6]), and they can model rough environments more compactly than planar patches (e.g. [7]).

Instead of building patches with widely varying boundary size, we propose to use one or several patch sizes driven by the task and robot structure. For example, in rough terrain walking the patch size(s) could be approximately the same as robot's foot pads. Patches may overlap to provide a sampling of possible contacts in smooth regions, and they may also only partially cover the environment.

Figure 5 shows the set of bounded paraboloid patches from our prior work [8]. We use the models and the fitting algorithm from that work, but now fit many patches randomly over an environment surface that has been sampled by a range sensor. We are also working on detecting salient features such as peaks, valleys, and flats, and to integrate task-specific strategies to only sample relevant parts of the environment. These will be important refinements, but they will not obviate the need to (a) find good neighborhoods in the range data and (b) validate the fitted patches, which are the main topics of this paper.

Our algorithm (Figure 1) is divided into four main steps for each patch; patches may be fit either iteratively or in parallel. The first step is the selection of a seed point. Next is the neighborhood search around that point. We compare two different approaches for finding local neighborhoods, one using a triangle mesh structure and the other using k-dimensional (k-d) trees, which have different properties near surface discontinuities. The third step is to fit the pose, curvatures, and boundary of a paraboloid patch to the neighborhood. Finally the patch is evaluated using two different metrics, and possibly discarded. We use both a Euclidean residual to quantify fit quality and a grid-based coverage algorithm to ensure that the patch is sufficiently representative of the actual data. The system has been implemented in the *Surface Patch Library* (SPL), with code available on our website [9].

Next we cover related work and review the key data structures. We then give the details of the algorithm including neighborhood searching, a new refinement to the fitting algorithm to avoid side-wall fits (Fig. 6), and the residual and coverage evaluation algorithms. Finally we present experimental results for 10 datasets acquired outdoors in natural rocky terrain plus one artificial dataset.

Regarding the choice of k-d tree vs triangle mesh for neighborhood searching, we find that k-d tree has a slight advantage in producing consistently sized patches (patch size is fit to neighborhood extent, which each algorithm attempts to regularize in a different way), but also is more likely to result in patches with larger Euclidean residual.

A. Related Work

The approach explored here contrasts with the traditional study of range image segmentation [10]–[12] where a dense partition of connected and non-overlapping but potentially irregularly bounded—and usually planar [13]–[15]—regions is generated. Irregularly bounded regions, which may be very large or non-convex, can present a challenge for higher-level contact planning algorithms which still need to search for specific contact areas within each region. One aim of using regularly bounded regions of approximately the same size as relevant contact features on the robot is to trade such

The authors are with the College of Computer and Information Science, Northeastern University, Boston, Massachusetts dkanou@ccs.neu.edu, vona@ccs.neu.edu

[1]Non-Uniform Rational B-Spline.

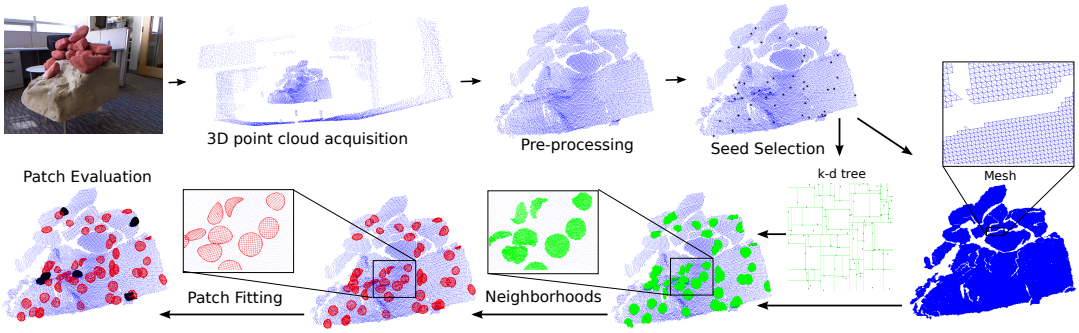

Fig. 1. System overview showing the main steps in the algorithm and the choice of k-d tree or triangle mesh data structures.

potentially complex continuous searches *within* patches to a discrete search *across* patches.

Modeling the environment and detecting potential contact surface areas for locomotion using exteroceptive sensing is a common task, but still very challenging. Grid-based systems like those on Ambler [16], Dante II [17], and MER [4] use laser scanners or stereo cameras to build an elevation map, find obstacles, and quantify traversability, but usually don't model detailed 3D contact features.

Visual odometry and stereo cameras or range sensors have also been used on several current walking robots including BigDog [18], [19] and DLR Crawler [20], though again mainly for obstacle avoidance and traversability analysis, not detailed 3D foot placement or contact planning. Some steps have been made in that direction in [21], where terrain is modeled using a Gaussian Process, and in [5] where a continuous surface model is used for Little Dog locomotion. Also, several works use stereo vision or laser scanners to map planar environment segments for humanoids operating indoors in controlled conditions [15], [22].

In [23] texture synthesis was presented to deal with the problem of occluded terrain by filling in the missing portions. Our approach avoids representing such missing areas where uncertainty is high. We instead plan to integrate multiple range scans taken from different perspectives as the robot moves to fill in missing areas with new observations using a volumetric fusion approach [24], [25].

II. DATA STRUCTURES

Range sensors are now commonly available. Stereo and structured light sensors like the Kinect, time-of-flight cameras, and laser scanners produce clouds of 3D sample points of environment surfaces. Here we focus on point cloud data in the form of an image acquired from a single point of view, with N 3D sample points organized in an $N_r \times N_c$ grid, $N_r N_c = N$. Due to occlusions, limited sensor range, and other factors, there may be grid cells map where depth information is invalid. (Grid-organized data can also be synthesized from other representations, including from the volumetric fusion mentioned above, by raycasting.)

Our algorithm requires finding local neighborhoods of 3D points, for which spatial decompositions like k-dimensional (k-d) trees [26] are commonly used. Another common structure for representing 3D sample points of surfaces is the triangle mesh. We next present details of these two structures.

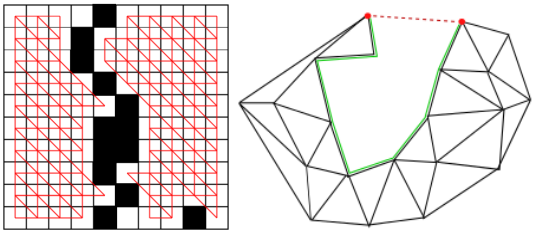

Fig. 2. Left: Terrain mesh in a 10-by-10 grid. The black pixels are those that either have invalid depth or either belong to a Canny edge (see text). Right: the chain distance (green) can distinguish points separated by a jump, whereas Euclidean 3D distance (red) may not.

A. Triangle Mesh

The triangle mesh structure can be constructed quickly since the input data is in the form of a grid. The basic algorithm is to locally connect (x, y) grid neighbors with triangle edges using only the presence or absence of valid depth data, but not the actual z values. We connect neighboring valid points in the same row and column and close triangles by adding diagonals (Fig. 2, left).

A well known problem (Fig. 3) is that depth discontinuities, i.e. *jumps*, between (x, y) neighbors will be bridged. To address this we use Canny edge detection on the z values [27]. The resulting edge points are used to limit triangle construction, creating gaps in the mesh at jumps. However, Canny edge detection does not guarantee continuous edges. To help with this, we also remove both the triangles with sides longer than a threshold $T_{es} = 5$cm and those whose ratio of the longest side to shortest side (aspect ratio) is more than a threshold $T_{ar} = 5$.

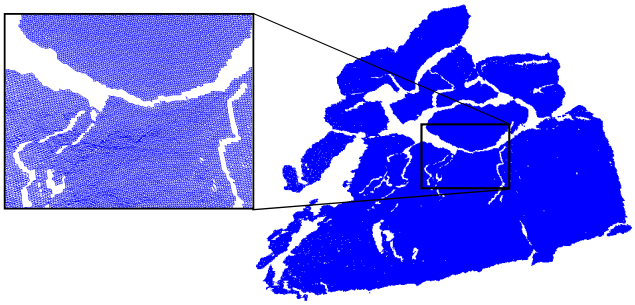

Fig. 3. Example of depth jumps between neighboring pixels.

Mesh building, Canny edge detection, and removal of long triangles are all $O(N)$. The cost for finding k nearest neighbors (with breadth first search) is $O(k)$.

B. K-D Tree

One of the most common data structures for spatial points is the k-d tree [26]. Whereas the triangle mesh approach depends on the grid organization of the data, k-d trees can be constructed from any point cloud. However, k-d trees do not directly encode information about depth discontinuities.

The cost for building a k-d tree is $O(N \log^2 N)$ when using an $O(N \log N)$ sorting algorithm for computing medians, or $O(N \log N)$ with a linear median-finding algorithm [26]. The cost for finding k nearest neighbors is $O(k \log N)$.

III. SPARSE SURFACE MODELING

Using one of the above data structures for neighborhood finding, patches are fit either iteratively or in parallel to the environment surface samples using the following algorithm:

1) Pick a seed point on the surface (Sec. III-A).
2) Find its neighbors within distance r (Sec. III-B).
3) Fit a bounded patch to the neighborhood (Sec. III-C).
4) Evaluate the patch with residual and coverage metrics and discard it unless both checks pass (Sec. III-D).
5) Continue until termination (Sec. III-E).

The neighborhood size r is set to a fixed value (e.g. 5–10cm) derived from the size of the intended contact surface on the robot[2]. Pre-processing, like background removal and decimation, can be applied first depending on the application. Several termination checks may be applied, for example, checking if the sum of the patch areas has reached a given ratio of the total surface area.

Next we describe each step of the algorithm in detail.

A. Seed Selection

Here we use random seed selection as a baseline method and general task-independent approach. Figure 1 illustrates 50 randomly seeded patches. We are also considering some types of (possibly task-specific) salient points [28], including peaks and valleys, and the possibility to direct more samples in task-specific areas of interest.

B. Patch Neighborhoods

Finding nearest neighbors in a point cloud is a well-studied problem. Many methods have been introduced, including approximations. Two concepts of a neighborhood are: (a) k nearest neighbors, i.e. the k closest points to a seed; (b) all neighbors within distance r from the seed, for some distance metric. For fitting uniformly bounded patches we use the latter; the number of points k in the recovered neighborhood thus varies depending on r and the specifics of the distance metric and the search algorithm (Fig. 4).

[2]Of course the robot may also make contact in other ways. The patch model could help plan intentional contacts while other data structures are simultaneously used for general collision prediction.

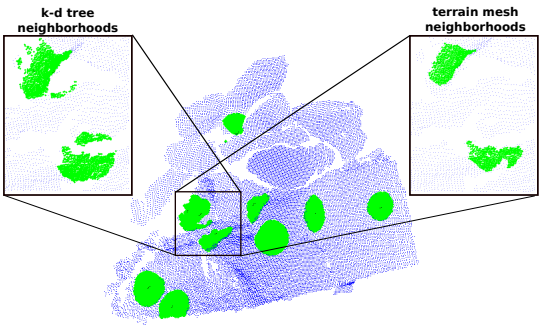

Fig. 4. Ten neighborhoods with $r = 0.05$m. Unlike k-d tree neighborhoods, triangle mesh neighborhoods do not span surface discontinuities.

Using the Triangle Mesh: First define *chain distance* as the weighted edge path length between vertices in the mesh, with the weight between two vertices that share an edge equal to their Euclidean distance in 3D. To find neighbors within distance r from a seed point we apply a breadth-first search from the seed, pruning it when the chain distance exceeds r. In that way we reduce the chances that the extracted neighbors cross discontinuities in the point cloud, even if they are spatially close (Fig. 2, right).

Using the K-D Tree: We search for neighbors within Euclidean distance r of the seed using the classic method introduced in [26]. The extracted neighborhood may span surface discontinuities (Fig. 4).

C. Patch Modeling and Fitting

We fit a bounded patch to the neighborhood using the paraboloid patches from [8]. We briefly review the model and fitting algorithm and we describe the *side-wall effect* that the original algorithm didn't consider.

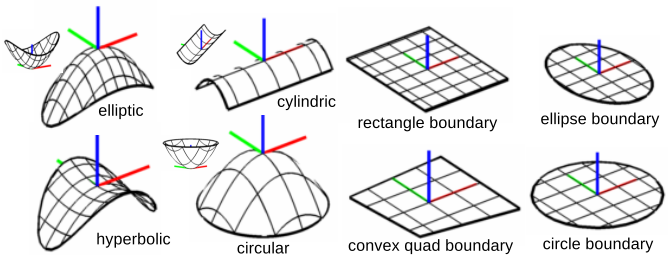

Fig. 5. Paraboloid patches (including planes) with regular boundaries.

Patch Modeling: Here we consider paraboloid patches paired with elliptic, circular, rectangle, or convex quadrilateral boundaries (Fig. 5). Paraboloids are complete in that they form an approximation system for local regions on any smooth surface. They are parametrized using *intrinsic* parameters for patch's shape, i.e. principal curvatures and boundary curve parameters, and *extrinsic* parameters for spatial pose, i.e. rotation and translation vectors $(\mathbf{r}, \mathbf{t}) \in \mathbb{R}^3 \times \mathbb{R}^3$. The implicit form for a paraboloid with curvatures (κ_x, κ_y) in the local frame[3] L defined by $(\mathbf{r}, \mathbf{t})$ is

$$\mathbf{p_l}^T K \mathbf{p_l} - 2\mathbf{p_l}^T \hat{\mathbf{z}} = 0 \qquad (1)$$
$$K \triangleq \mathrm{diag}(\kappa_x, \kappa_y, 0), \qquad \hat{\mathbf{z}} \triangleq [0\ 0\ 1]^T$$

where $\mathbf{p_l} \in \mathbb{R}^3$ is a point on the patch in frame L.

[3]L is also the *Darboux* frame of the paraboloid.

Patch Fitting: A nonlinear fitting algorithm is used based on a variation of Levenberg-Marquardt iteration that fits bounded curved patches to N sample points $\mathbf{q}_i \in \mathbb{R}^3$. The algorithm minimizes a sum-of-squares residual by optimizing the patch implicit and explicit parameters. The residual for an individual sample point $\mathbf{q}_i$ scales the value of the implicit form (1) by the inverse of a first-order estimate of its standard deviation, which is derived in turn from a covariance matrix modeling the sensor uncertainty for $\mathbf{q}_i$.

Type selection for the boundary curve is only partially automatic—rectangles and convex quads are only used if requested. We cover them here for completeness; only circle and ellipse boundaries are used in the experiments.

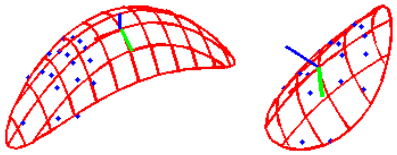

Fig. 6. The reparameterization in Eq. (2) keeps the fitted paraboloid centered on the data (right). This prevents the "side-wall" effect (left) and helps ensure good coverage, but can compromise the Euclidean residual.

Side-wall effect problem: When the neighborhood points don't have a central symmetry then they may be unevenly distributed in the patch when using the original fitting algorithm in [8]. We call this the *side-wall effect* (Fig. 6). To handle this we apply a constrained fitting where the center of the patch $\mathbf{c} \in \mathbb{R}^3$ must lie on the line through the centroid $\mathbf{c}_p$ of the neighborhood parallel to the normal $\mathbf{n}_p$ to an initial fit plane. This is implemented as a reparameterization

$$\mathbf{c} = \mathbf{c}_p + a\mathbf{n}_p \tag{2}$$

where $a \in \mathbb{R}$ is the new patch parameter replacing $\mathbf{c}$.

D. Patch Evaluation

The last part of the algorithm is the evaluation of the patch and the decision whether to keep it. We use two measures based on the *residual* and *coverage*. The first one evaluates the surface and the second one the boundary of the patch.

Residual Evaluation: The patch residual measures the deviation between the sample points and the (unbounded) paraboloid patch surface. We use the root-mean-square error (*RMSE*) Euclidean residual ϱ between the sample points $\mathbf{q}_i$ and their corresponding closest points $\mathbf{p}_i$ on the patch:

$$\varrho = \sqrt{\frac{\sum_{i=1}^{N} \|\mathbf{q}_i - \mathbf{p}_i\|^2}{N}}. \tag{3}$$

(Here we consider both $\mathbf{q}_i$ and $\mathbf{p}_i$ to be expressed in the patch local frame L.)

Whereas the patch fitting algorithm uses an algebraic residual for speed, ϱ is a Euclidean residual and gives its result in the same physical units as the input data (e.g. meters) [29], enabling it to be compared to a meaningful threshold. However, the required $\mathbf{p}_i$ must be calculated for each $\mathbf{q}_i$.

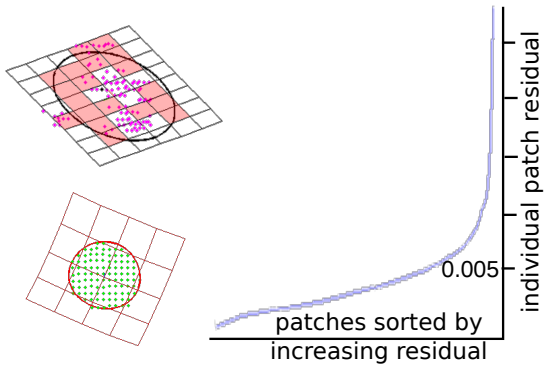

Fig. 7. Left: Coverage evaluation for a bad (top) and a good (bottom) patch with bad cells in red. Right: Sorted residuals for 1000 random patches (see text), approximately 95% of which are below 0.01.

When $\kappa_x \approx \kappa_y \approx 0$ the paraboloid surface was fitted as a plane, so $\mathbf{p}_i = (I - \hat{\mathbf{z}}\hat{\mathbf{z}}^T)\mathbf{q}_i$, i.e. $\mathbf{p}_i$ is the projection of $\mathbf{q}_i$ onto the xy plane of L. Otherwise $\mathbf{p}_i$ is characterized as:

$$\min_{\mathbf{p}_i \text{ satisfying (1)}} \|\mathbf{q}_i - \mathbf{p}_i\|. \tag{4}$$

A solution can be found using Lagrange multipliers [30]. Define the Lagrange function Λ as

$$\Lambda(\mathbf{p}_i, \lambda) = (\mathbf{q}_i - \mathbf{p}_i)^T(\mathbf{q}_i - \mathbf{p}_i) + \lambda(\mathbf{p}_i^T K \mathbf{p}_i - 2\mathbf{p}_i^T \hat{\mathbf{z}}). \tag{5}$$

with Lagrange gradient constraints

$$\nabla\Lambda(\mathbf{p}_i, \lambda) = \mathbf{0}^T \Leftrightarrow \partial\Lambda/\mathbf{p}_i = [0\ 0\ 0] \text{ and } \partial\Lambda/\lambda = 0. \tag{6}$$

Expand the first gradient constraint from (6)

$$\begin{aligned} -2\mathbf{q}_i^T + 2\mathbf{p}_i^T + \lambda(2\mathbf{p}_i^T K - 2\hat{\mathbf{z}}^T) &= [0\ 0\ 0] \\ -\mathbf{q}_i + \mathbf{p}_i + \lambda(K\mathbf{p}_i - 2\hat{\mathbf{z}}) &= [0\ 0\ 0]^T \\ (I + \lambda K)\mathbf{p}_i &= \mathbf{q}_i + \lambda\hat{\mathbf{z}} \\ \mathbf{p}_i &= (I + \lambda K)^{-1}(\mathbf{q}_i + \lambda\hat{\mathbf{z}}) \end{aligned} \tag{7}$$

and substitute for $\mathbf{p}_i$ in the second gradient constraint, which is the same as (1). This leads to a fifth degree polynomial in λ, for which there is at least one real solution because imaginary solutions come in pairs. Finally, backsubstitute[4] the real solution(s) in (7) and find the minimum as in (4).

To determine the residual threshold T_r such that any patch with $\varrho > T_r$ will be dropped, we sorted all residuals (Fig. 7, right) for a sampling of 1000 random patches ($r = 0.1$m, k-d tree neighborhoods), 100 on each of our 10 rock datasets. The value $T_r = 0.01$m was selected to include approximately 95% of the patches. In general T_r can be set in an application dependent way. Furthermore, the choice of RMSE residual is not essential. For example, an alternate residual

$$\varrho_{alt} = \max \|\mathbf{q}_i - \mathbf{p}_i\| \tag{8}$$

could be used to check if any small surface bumps protrude more than a desired amount from the surface.

[4]Division by zero can occur during this backsubstitution when $\mathbf{q}_i$ is on a symmetry plane or axis, but alternate forms can be used in those cases.

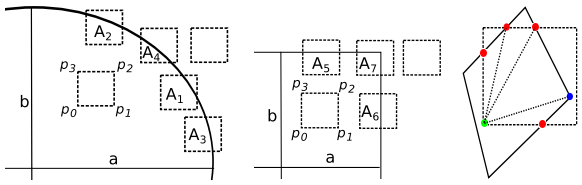

Fig. 8. Left: the six possible placements of a cell for the top right quadrant of an elliptic boundary. Middle: similar for the axis-aligned rectangular boundary. Right: example of an intersection between a general convex quadrilateral boundary and a grid cell.

Coverage Evaluation: A different evaluation is needed to take into account the patch boundary. A patch may fit the data but still not faithfully represent the neighborhood, either because too many sample points are outside the patch boundary or there is too much area inside the boundary that is not supported by data points. (We opt not to speculatively fill holes in the data.)

To detect these cases we generate an axis-aligned grid of fixed pitch w_c on the xy plane the patch local frame L. We generate only the required number of rows and columns in this grid to fit the projection of the patch boundary.

Define I_c and O_c to be the number of data points whose xy projections are both inside a given cell and respectively inside or outside the projected patch boundary. Define A_i to be the area of the geometric intersection of the cell and the projected patch boundary, which will be detailed below. The cell is considered *bad* iff

$$I_c < \frac{A_i}{w_c^2} T_i \text{ or } O_c > (1 - \frac{A_i}{w_c^2}) T_o. \tag{9}$$

for thresholds T_i and T_o. Here we fix these thresholds relative to the expected number of samples N_e in a given cell if all samples were in-bounds and evenly distributed:

$$T_i = \zeta_i N_e, \quad T_o = \zeta_o N_e, \quad N_e \triangleq k/N_p, \quad N_p \triangleq \frac{A_p}{w_c^2}, \tag{10}$$

where k is the number of sample points in the neighborhood and A_p is the area of the patch approximated as the area inside the projected boundary.

The patch fails coverage evaluation iff there are more than T_p bad cells. After some experiments in fitting paraboloid patches with $r = 0.1$m, we set $w_c = 0.01$m, $\zeta_i = 0.8$, $\zeta_o = 0.2$, and $T_p = 0.3N_p$. Figure 7 (left) illustrates patches that pass and fail coverage evaluation.

Intersection Area for Ellipse and Circle Boundaries: For an ellipse boundary with radii a, b, or for the degenerate case of a circle boundary with radius $r = a = b$, we compute the intersection area with a secant approximation since the exact computation involves a relatively expensive inverse trig function. Wlog we describe only the top right quadrant (Fig. 8, left); the other three are symmetric. Let $\mathbf{p}_{0\ldots3}$ be the four corners of a grid cell in counter-clockwise order starting from the lower left. The algorithm for computing the intersection area is:

1) If $\mathbf{p}_2$ is inside the ellipse then $A_i = w_c^2$
2) else if $\mathbf{p}_0$ is not inside the ellipse then $A_i = 0$
3) else if $\mathbf{p}_1$ is inside the ellipse then
   if $\mathbf{p}_3$ is inside the ellipse then $A_i = A_1$ else $A_i = A_2$
4) else if $\mathbf{p}_3$ is inside the ellipse then $A_i = A_3$
5) else $A_i = A_4$.

$$A_1 = (x_b - x_0)w_c + (x_c - x_b)(Y(x_b) - y_0) + ((x_c - x_b)(y_0 + w_c - Y(x_b)))/2 \tag{11}$$

$$A_2 = (x_c - x_0)(Y(x_c) - y_0) + (x_c - x_0)(Y(x_0) - Y(x_c))/2 \tag{12}$$

$$A_3 = (y_c - y_0)(X(y_c) - x_0) + (y_c - y_0)(X(y_0) - X(y_c))/2 \tag{13}$$

$$A_4 = (X(y_0) - x_0)(Y(x_0) - y_0)/2 \tag{14}$$

$$X(y) \triangleq a\sqrt{1 - y^2/b^2}, \; Y(x) \triangleq b\sqrt{1 - x^2/a^2}$$
$$x_b \triangleq X(y_0 + w_c), \; x_c \triangleq x_0 + w_c, \; y_c \triangleq y_0 + w_c$$
$$[x_0, y_0]^T \triangleq \mathbf{p}_0$$

Intersection Area for Axis-Aligned Rectangle Boundary: As above we consider only the top right quadrant (Fig 8, middle). Let the rectangle half-lengths be a, b, and define $[x_0, y_0]^T \triangleq \mathbf{p}_0$. The exact intersection area can be computed as follows:

1) If $\mathbf{p}_2$ is inside the rect then $A_i = w_c^2$
2) else if $\mathbf{p}_0$ is not inside the rect then $A_i = 0$
3) else if $\mathbf{p}_1$ is inside the rect then $A_i = A_5 = w_c(b - y_0)$
4) else if $\mathbf{p}_3$ is inside the rect then $A_i = A_6 = w_c(a - x_0)$
5) else $A_i = A_7 = (a - x_0)(b - y_0)$.

Intersection Area for Convex Quadrilateral Boundary: To handle the case of a general convex quadrilateral (Fig. 8, right), we use the fact that the intersection between a convex quad and a rectangle is always convex:

1) Find the set of grid cell corner points that are inside the quad and vice-versa.
2) Find the intersection points between the grid cell boundaries and the convex quad boundaries.
3) Discard all points from steps 1 and 2 except those that lie in or on both figures.
4) Sort all the points computed in the previous steps in counterclockwise order and connect the first point with each of the others in order, forming a triangle fan. A_i is the sum of the triangle areas.

E. Termination Criteria

Various termination criteria can be used, for example: wall-clock time, total number of patches, or task-specific criteria. Another approach is to specify a desired fraction ν of the total sampled surface area S that should probabilistically be covered by patches. Note that ν can be both less than 1, to sample sparsely, or more than 1, to oversample. With r-ball neighborhood search (using either k-d tree or triangle mesh) and ellipse-bounded paraboloid patch fitting we can estimate the expected number of patches for this criteria as

$$\nu \frac{S}{\pi r^2}. \tag{15}$$

Or, as we do in the experiments below, we can fit patches until the sum of their areas reaches or exceeds νS.

Data	Structure	Patches		Dropped patches			Average residual (mm)	Total area (m^2)
		total	valid	due to residual	due to coverage	total		
rock 1	k-d tree	167	142	15	10	25	4.6	4.57
	tri mesh	164	144	6	14	20	4.3	4.57
rock 2	k-d tree	160	107	18	36	53	5.0	3.13
	tri mesh	185	124	0	61	61	4.0	3.13
rock 3	k-d tree	231	183	24	24	48	5.2	5.53
	tri mesh	227	199	1	27	28	4.7	5.53
rock 4	k-d tree	220	164	27	31	56	5.0	5.01
	tri mesh	215	181	8	29	34	4.8	5.01
rock 5	k-d tree	195	157	17	24	38	5.4	4.86
	tri mesh	188	163	5	20	25	5.1	4.86
rock 6	k-d tree	235	185	31	20	50	5.7	5.69
	tri mesh	273	226	9	39	47	5.3	5.69
rock 7	k-d tree	267	213	30	25	54	4.4	6.54
	tri mesh	266	219	17	30	47	4.2	6.54
rock 8	k-d tree	260	223	16	22	37	4.2	7.04
	tri mesh	256	231	2	23	25	3.9	7.04
rock 9	k-d tree	187	159	13	16	28	4.8	4.95
	tri mesh	189	162	9	19	27	4.6	4.95
rock 10	k-d tree	301	223	30	50	78	5.0	6.98
	tri mesh	300	236	9	55	64	4.7	6.98
rock average	**k-d tree**	**222**	**176**	**22**	**26**	**47**	**4.9**	**5.43**
	tri mesh	**226**	**189**	**7**	**32**	**38**	**4.6**	**5.43**
fake	k-d tree	65	18	18	44	47	3.8	0.50
	tri mesh	75	21	1	54	54	3.8	0.50

TABLE I

In practice it is nontrivial both to calculate the total sampled surface area S and the area of any individual patch. To approximate S we compute the triangle mesh (Sec. II-A) and sum the triangle areas. We approximate the area of an individual patch as the area of the projection of its boundary on the xy plane of the patch local frame L.

IV. EXPERIMENTAL RESULTS

We performed experiments to test the overall approach and to compare the triangle mesh and k-d tree data structures for neighborhood finding. 11 datasets were collected with the Kinect at 640×480 resolution and then decimated by 2. The first 10 are scenes of natural rocky terrain (Fig. 11) acquired with a hand-held Kinect outdoors on an overcast day (the Kinect does not work in direct sunlight). The last is taken in the lab with synthetic rocks (Fig. 1).

The parameters were: neighborhood radius $r = 0.1$m, residual threshold $T_r = 0.01$m, coverage cell size $w_c = 0.01$m, coverage threshold factors $\zeta_i = 0.8, \zeta_o = 0.2$, and $T_p = 0.3A_p/w_c^2$ (all motivated above). We let the algorithm run for each dataset until the sum of the patch areas equaled or exceeded 90% of the sampled surface area, both approximated as described in Sec. III-E. Our current Matlab implementation is not optimized and takes about 70ms per patch. We aim to improve this for real-time use by optimization and parallelization.

Qualitatively, as depicted in Figure 9 and 10, the algorithm appears to give a reasonable representation of non-smooth environment surfaces. Quantitatively, we measured the following statistics (Table I): the total number of patches before evaluation, the number of valid patches passing both residual and coverage evaluation, the number of dropped patches for each test, the average Euclidean residual (of the valid patches), and the total surface area for each dataset.

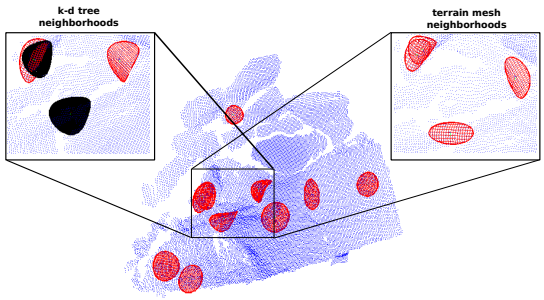

Fig. 9. A subset of patches fit to the fake rock dataset corresponding to the neighborhoods in Figure 4. The black patches failed coverage evaluation.

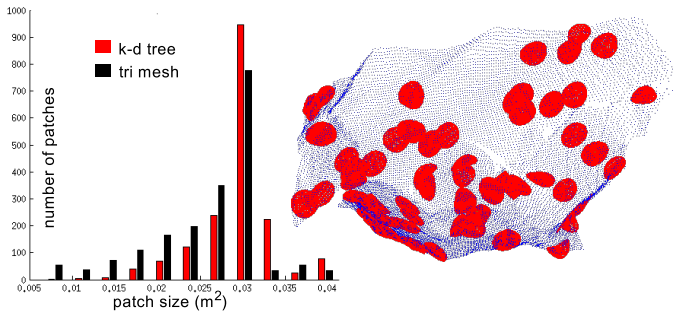

Fig. 10. Left: Histogram of patch sizes for k-d tree (red) and triangle mesh (black) neighborhoods. Right: 70 patches on the dataset rock 3.

There are generally more patches dropped due to residual for k-d tree neighborhoods, possibly because the triangle mesh neighborhoods avoid discontinuities which may not be fit well by a paraboloid. We see the opposite effect for patches dropped due to coverage: more patches are generally dropped due to insufficient coverage when using triangle mesh neighborhoods. The k-d tree neighborhoods may distribute samples more evenly, particularly near discontinuities.

Another interesting result is that more patches are required to reach 90% of the surface area when using triangle mesh neighborhoods. In Figure 10 we see that the distribution of patch areas created using mesh neighborhoods is skewed more to the low side than those created using k-d tree neighborhoods. This can again be explained by the fact that k-d tree neighborhoods will span discontinuities but remain roughly circular, whereas triangle mesh neighborhoods may be less circular when the seed point is near an edge.

V. CONCLUSIONS AND FUTURE WORKS

Paraboloid patches with one or two curvatures and regularly shaped bounds, sized to match contact surfaces on a robot, are an alternative to traditional segmentations for representing 3D contact surfaces in unstructured environments. The set of bounded patch models and the fit-and-validate strategy described here could be the basis for such a system. We investigated several aspects of the system design, in particular, the choice of data structures for neighborhood searching and the type of validation tests to use.

With respect to the relative merits of k-d tree vs triangle mesh data structures, our experiments indicate that both approaches are competitive. K-D trees do not directly encode discontinuity information, resulting in more patches dropped

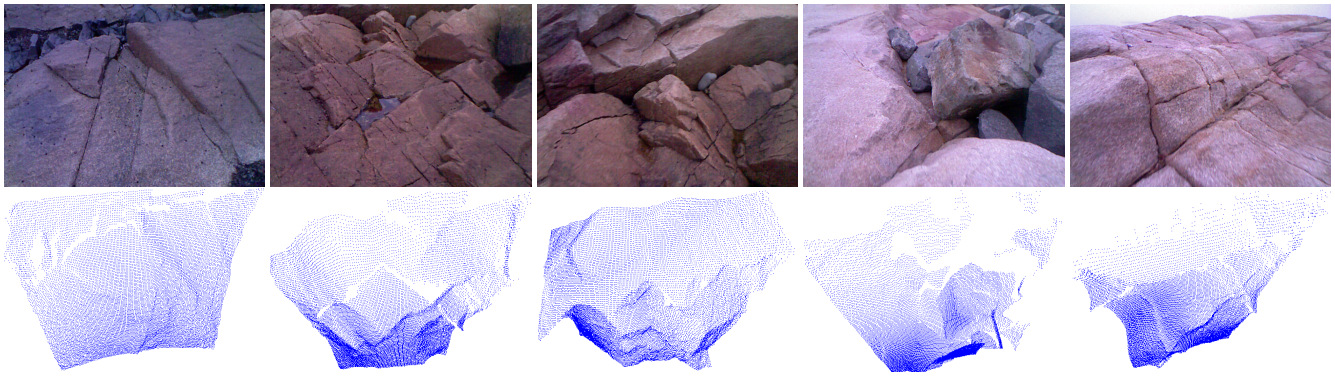

Fig. 11. Datasets rock 1–5. All rock datasets were acquired with a hand-held Kinect outdoors on an overcast day.

due to large residual, but also produce more consistently sized neighborhoods than triangle mesh. Though we use grid-organized samples here, general point clouds can be easier to handle with k-d trees than with triangle mesh. Preliminary timing results also suggest that the larger theoretical time complexity for k-d trees is not a significant factor in practice.

We are now optimizing the implementation and exploring adding salient features as additional seed points to ensure that task-relevant areas are sufficiently sampled by patches. We are also planning to integrate sparse patch-based modeling with dense volumetric fusion of range data from a moving camera [25], with the goal of maintaining a dynamic "patch map" of relevant contact surfaces around a robot. In particular, we are intending to use these methods in a new perceptual system for bipedal locomotion on rough terrain (see [8], Figure 4) where individual foot contact surfaces must first be located visually before stepping.

References

[1] S. Jajita and T. Sugihara, "Humanoid Robots in the Future," *Advanced Robotics*, vol. 23, pp. 1527–1531, 2009.

[2] "Honda - ASIMO," http://asimo.honda.com/.

[3] M. Raibert, K. Blankespoor, G. Nelson, R. Playter, and the BigDog Team, "Bigdog, the Rough-Terrain Quadruped Robot," *World Congress of The Int. Fed. of Automatic Control*, 2008.

[4] M. W. Maimone, P. C. Leger, and J. J. Biesiadecki, "Overview of the Mars Exploration Rovers' Autonomous Mobility and Vision Capabilities," in *IEEE Int. Conf. on Robotics and Automation*, 2007.

[5] C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, and W. Burgard, "A Bayesian Regression Approach to Terrain Mapping and an Application to Legged Robot Locomotion," *Journal of Field Robotics*, vol. 26, no. 10, pp. 789–811, Oct. 2009.

[6] A. Richtsfeld, T. Mörwald, J. Prankl, J. Balzer, M. Zillich, and M. Vincze, "Towards Scene Understanding—Object Segmentation Using RGBD-Images," in *Computer Vision Winter Workshop*, 2012.

[7] N. Vaskevicius, A. Birk, K. Pathak, and S. Schwertfeger, "Efficient Representation in 3D Environment Modeling for Planetary Robotic Exploration," *Advanced Robotics*, vol. 24, no. 8-9, 2010.

[8] M. Vona and D. Kanoulas, "Curved surface contact patches with quantified uncertainty," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2011.

[9] D. Kanoulas and M. Vona, "Surface Patch Library (SPL)," 2012, http://www.ccs.neu.edu/research/gpc/spl.

[10] X. Jiang and H. Bunke, "Fast Segmentation of Range Images into Planar Regions by Scan Line Grouping," *Machine Vision and Application*, vol. 7, no. 2, pp. 115–122, 1994.

[11] A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. Goldgof, K. Bowyer, D. Eggert, A. Fitzgibbon, and R. Fisher, "An Experimental Comparison of Range Image Segmentation Algorithms," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, no. 7, pp. 673–689, 1996.

[12] M. Powell, K. Bowyer, X. Jiang, and H. Bunke, "Comparing Curved-Surface Range Image Segmenters," in *Int. Conf. Comp. Vision*, 1998.

[13] K. Pathak, N. Vaskevicius, and A. Birk, "Revisiting Uncertainty Analysis for Optimum Planes Extracted from 3D Range Sensor Point-Clouds," in *IEEE Int. Conf. on Robotics and Automation*, 2009.

[14] J. Weingarten and R. Siegwart, "3D SLAM Using Planar Segments," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006.

[15] J.-S. Gutmann, M. Fukuchi, and M. Fujita, "3D Perception and Environment Map Generation for Humanoid Robot Navigation," *The Int. J. of Robotics Research*, vol. 27, no. 10, pp. 1117–1134, 2008.

[16] E. Krotkov and R. G. Simmons, "Perception, Planning, and Control for Autmonomous Walking With the Ambler Planetary Rover," *Int. J. of Robotics Research*, vol. 15, no. 2, pp. 155–180, 1996.

[17] J. Bares and D. Wettergreen, "Dante II: Technical Description, Results, and Lessons Learned," *Int. J. of Robotics Research*, vol. 18, no. 7, pp. 621–649, 1999.

[18] A. Howard, "Real-time Stereo Visual Odometry for Autonomous Ground Vehicles," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008.

[19] D. Wooden, M. Malchano, K. Blankespoor, A. Howard, A. A. Rizzi, and M. Raibert, "Autonomous Navigation for BigDog," in *IEEE Int. Conf. on Robotics and Automation*, 2010.

[20] A. Stelzer, H. Hirschmüller, and M. Görner, "Stereo-Vision-Based Navigation of a Six-Legged Walking Robot in Unknown Rough Terrain," *Int. J. Robotics Research*, vol. 31, no. 4, pp. 381–402, 2012.

[21] T. Lang, C. Plagemann, and W. Burgard, "Adaptive Non-Stationary Kernel Regression for Terrain Modelling," in *Robotics: Science and Systems*, 2007.

[22] K. Nishiwaki, J. Chestnutt, and S. Kagami, "Autonomous Navigation of a Humanoid Robot over Unknown Rough Terrain using a Laser Range Sensor," *Int. J. of Robotics Research*, vol. 31, no. 11, pp. 1251–1262, 2012.

[23] J. Z. Kolter, Y. Kim, and A. Y. Ng, "Stereo Vision and Terrain Modeling for Quadruped Robots," in *IEEE Int. Conf. on Robotics and Automation*, 2009.

[24] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-Time Dense Surface Mapping and Tracking," in *IEEE Int. Sym. on Mixed and Augmented Reality*, 2011.

[25] H. Roth and M. Vona, "Moving volume KinectFusion," in *British Machine Vision Conference*, 2012.

[26] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[27] E. Wolfart, V. Sequeira, K. Ng, S. Butterfield, J. Gonçalves, and D. Hogg, "Hybrid Approach to the Construction of Triangulated 3D Models of Building Interiors," in *Computer Vision Systems*. Springer, 1999, pp. 489–508.

[28] Y.-S. Liu, M. Liu, D. Kihara, and K. Ramani, "Salient Critical Points for Meshes," in *ACM Symp. on Solid and Physical Modeling*, 2007.

[29] J. Ponce, D. J. Kriegman, S. Petitjean, S. Sullivan, G. Taubin, and B. Vijayakumar, "Representations and Algorithms for 3D Curved Object Recognition," in *Three-Dimensional Object Recognition Systems*, P. Flynn and A. Jain, Eds. Elsevier Press, 1993, pp. 327–352.

[30] D. Eberly, "Distance from Point to a General Quadratic Curve or a General Quadric Surface," Tech. Rep., 1999.