

# Identifying Support Surfaces of Climbable Structures from 3D Point Clouds

Anna Eilering<sup>1</sup>, Victor Yap<sup>2</sup>, Jeff Johnson<sup>1</sup>, and Kris Hauser<sup>1</sup>

**Abstract**—This paper presents a probabilistic technique for identifying support surfaces like floors, walls, stairs, and rails from unstructured 3D point cloud scans. A Markov random field is employed to model the joint probability of point labels, which can take on a number of user-defined surface classes. The probability of a point depends on both local spatial features of the point cloud around the point as well as the classifications of points in its neighborhood. The training step estimates joint and pairwise potentials from labeled point cloud datasets, and the prediction step aims to maximize the joint probability of all labels using a hill-climbing procedure. The method is applied to stair and ladder detection from noisy and partial scans using three types of sensors: a sweeping laser sensor, time-of-flight depth camera, and a Kinect depth camera. The resulting classifier achieves approximately 75% accuracy and is robust to variations in point density.

## I. INTRODUCTION

To navigate through human environments that contain climbable structures like stairs and ladders, robots like humanoids and legged robots must identify candidate support surfaces for placing hands, feet, tracks, or wheels, and thin objects like rails that may be used for grasping support. This requires techniques to distinguishing locomotion-relevant parts of the environment, such as floors, stairs, ladder rungs, and handrails apart from clutter and other irrelevant objects from raw sensor data. As opposed to “top-down” template matching techniques which need huge numbers of prototype models to capture variability in climbable structures, this paper presents a “bottom-up” approach that first estimates the classes of individual points, then aggregates points into coherent part neighborhoods. As a result, the technique is inherently able to model objects with variable numbers of steps, handrails, cross-sections, and internal supports simply as a collection of semantically-meaningful segmented parts.

Object recognition and segmentation is a well studied area of computer vision, but the robot navigation setting poses a number of challenges that are not characteristic of typical tabletop or urban driving recognition paradigms. In such paradigms, objects lie mostly in the frame of a single image, provided as a table of color and/or depth values. In

contrast, the part identification problem requires the robot to identify and segment objects that are fundamentally attached to others, may be quite thin, and may lie largely out of frame. Moreover, due to the large scale of stairs and ladders, the robot may need to perform several depth scans, requiring identification from the merged point clouds, which do not have the same tabular structure of single images. These point clouds are large, noisy, and have widely variable point densities.

We present a supervised learning algorithm that trains a Markov random field (MRF) model with vertices corresponding to surface points, with vertex potentials that relate geometric features of points to part types, and neighborhood potentials that capture the spatial coherence of parts. The model is learned from hand-labeled training data consisting of point clouds with user-defined part labels. In our implementation we use five part labels: floors, walls, graspable rails, spurious noise, and “other,” although it can be easily extended to more part types. The MRF network joins nearby points with an edge. Vertex potentials are modeled as the weighted voting of a random forest, learned over spatial features of a point’s neighborhood. Neighborhood potentials are learned over features that capture the distribution of neighbors’ labels and the shape of the neighborhood of the vertex in the graph, such as its degree and principal axes.

To label a novel scene, the technique formulates the MRF, computes point and edge features, and then maximizes likelihood of the labeling with a hill climbing method. The algorithm yields a part for each coherent cluster in the labeling. The technique is tested on data from a sweeping Hokuyo laser scanner, a SwissRanger SR3000 time-of-flight camera, and a Microsoft Kinect depth sensor. Experiments suggest that compared to using pointwise labeling alone, the MRF technique is more robust to noisy scans. The resulting model is fairly robust to variations in point cloud density, still retaining over 60% accuracy when density is only 10% of the original scans.

## II. BACKGROUND AND RELATED WORK

The first approaches for identifying structure from point clouds focused on the segmentation of planes, cylinders, and other geometric primitives using classical Hough transform, plane growing, or RANSAC-based techniques [1]–[3]. While effective at detecting large flat structures they often require considerable tuning to remain robust to noise, and they are also largely limited to parameterizable geometric primitives. Nevertheless, such methods have been applied successfully to vision-based navigation of humanoid robots up staircases [4].

\*This work was supported in part by a CRA-W Collaborate Research Experiences for Undergraduates (CREU) award and the Defense Advanced Research Projects Agency (DARPA) award # N65236-12-1-1005 for the DARPA Robotics Challenge. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of DARPA.

<sup>1</sup>School of Informatics and Computing, Indiana University at Bloomington, Bloomington, Indiana 47408 USA, {annaeile, jj56, hauserk}@indiana.edu

<sup>2</sup>University of Texas at Austin, Austin, Texas, USA victor.yap@utexas.edu

Template matching techniques [5] have been successful in searching a library of known objects to find matches in a point cloud image, and are able to address a variety of tabletop object recognition problems. But ladders and stairs are large objects that are seldom seen in full; and by treating the recognition problem from the bottom up from individual points to parts (and ultimately, to structures), we avoid the need to have a priori 3D models of all ladders or stairs of interest.

Markov random fields (MRFs) are probabilistic network models that are extensively used in computer vision for segmentation problems [6]. One of their favorable qualities for segmentation problems is the existence of efficient approximate graph-cut algorithms [7], [8] that obtain provable bounds on the cost of segmentation, given certain assumptions of edgewise costs. [9] propose a Markov random field algorithm for segmenting 3D scans using the graph-cut approach, and their work supports max-margin learning of vertex potentials via the solution to a quadratic program. They model vertex potentials as log-linear in vertex features, and pairwise edge potentials use the Potts model, which reward labels that are similar to neighbors. Our network model is similar, but we use more flexible vertex potentials via random forests which support larger training sets (on the orders of hundreds of thousands or millions, rather than tens of thousands). Moreover we train neighborhood potentials that are able to capture a wider variety of neighborhood features such as degree, shape, and extent of the neighborhood, which leads to more accurate segmentations. One potential drawback of our approach is that due to the more sophisticated model we are no longer able to use the graph-cut algorithm and instead must rely on local optimization. Nevertheless we observe satisfactory results in practice due to the greater predictive power of the neighborhood model.

Although the current work focuses on part segmentation we hope to extend our work to structural relationships between parts, and the MRF approach could be extended to handle such hierarchical models. Munoz et al (2009) present a higher-order Conditional Random Field (a cousin of MRFs) to improve segmentation of 3D point clouds [10]. Here, new network vertices are introduced corresponding to ad-hoc clusters, and cluster size and shape affect the strength of segmentation boundaries. Silberman et. al. (2011) studied MRFs for inferring support relations between objects and support surfaces in Kinect data [11]. They used a hierarchical clustering to segment a scene using RGB information and RANSAC-based plane fitting, then defined an MRF over clusters to infer support relations. Anand et al (2012) also present a higher-order model whose vertices are pre-segmented clusters, and cluster context is combined with predictors based on shape and appearance [12]. In future work we hope to incorporate some of these techniques to achieve hierarchical structure segmentation and recognition from low-level points to high-level context in a unified manner.

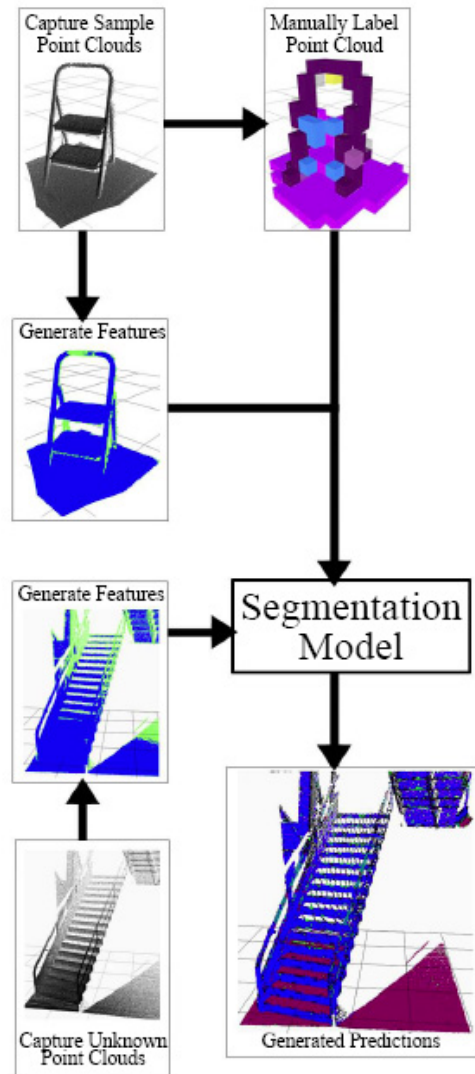


Fig. 1. Classifier Workflow.

### III. METHOD SUMMARY

This paper implements a system for a segmentation algorithm that is trained on example images. The workflow of using the technique is illustrated in Fig. 1. Here we will describe the details of the MRF model and training procedure.

#### A. Markov Random Field Model Summary

We wish to decompose a point cloud  $P = \{p_i, i = 1, \dots, n\}$ , generated through scans of a 3D environment, into a collection of disjoint *parts* which may be one of  $m$  *types*  $x_1, \dots, x_m$ . A part is spatially coherent, and more than one part of the same type may exist in the environment, but we assume that no two parts of the same type are connected. As a result we are interested in finding a part type (called a *label*)  $l_i \in \mathcal{L} = \{x_1, \dots, x_m\}$  for each point, and the resulting labeled point cloud can then be naturally segmented out into parts via graph cutting.

We consider points that contain only 3D position information due to our use of laser sensors that only provide depth, but future work might incorporate reflectance and color information provided by RGB+D cameras. To enable our segmentation algorithm to handle very large or dense point clouds we subsample the point cloud roughly evenly in space by discretizing the space into voxels and choosing a single representative point per occupied voxel; all points contained within the voxel will be associated with the label of the chosen representative point.

The MRF model builds a network  $(V, E)$  containing vertices  $V$  corresponding to the points in the point cloud, and a set of edges  $E$  between pairs of points that lie within radius  $R$  of one another. A prior probability distribution over labels is associated with each vertex, and a neighborhood probability distribution over a vertex labels given its neighbors' labels is also given. The goal of the labeling procedure is to find a labeling  $l_1, \dots, l_n$  to maximize the likelihood of the joint assignment of vertex labels:

$$P(l_1, \dots, l_n) = \frac{1}{Z} \prod_{i=1}^n \phi_{i,V}(l_i) \phi_{i,N}(l_i, l_{N_i}) \quad (1)$$

where  $\phi_{i,V}$  and  $\phi_{i,N}$  are the vertex and neighborhood potentials of the  $i$ 'th vertex, and  $l_{N_i}$  denotes the set of labels of its neighbors. The normalization term  $Z$  is independent of the labeling and hence can be ignored during optimization. This model differs from the Potts model which assumes each neighborhood factorizes into edgewise potentials between neighbors; as a result we are able to better capture structural dependencies between the labels of neighbors. Below we describe the model, learning, and inference in more detail.

### B. Vertex Potentials

Each vertex potential corresponds to the logarithm of a pointwise prior of being assigned a label given only the local geometry of the point-cloud. Specifically, a vertex potential  $P(l \mid f_1, \dots, f_k)$  is a probability distribution over the label at a point given a set of geometric features  $f_1(p), \dots, f_k(p)$  that describe the geometric shape of the neighborhood of a point  $p$ . This potential is assumed to apply uniformly over all points  $p_i$ .

The geometric features used to compute vertex potentials are divided into two types: *point-level* features that are computed for each point using a neighborhood about the point, and *region-level* features that are computed for a given set of points occupying some region in space. In this case, the regions were taken as grid elements formed by regular discretization of the space containing the point cloud, with the discretization performed by constructing an octree with resolution 0.075m. All features were computed using methods available through the Point Cloud Library (PCL) [13]

The point-level features were defined as follows:

- 1) **Local Normal:** The local normal feature  $f_n$  at a point  $p$  is estimated as being the third eigenvector found by performing principal component analysis on  $p$  and its

$k$  nearest neighbors. In our experiments  $k = 50$ . Note that  $f_n$  is a real-valued 3-tuple.

- 2) **Curvature:** The curvature feature  $f_\sigma \in [0, \frac{1}{3}]$  is computed using the eigenvalues found during local normal estimation as  $f_\sigma = \lambda_0/(\lambda_0 + \lambda_1 + \lambda_2)$  where  $\lambda_0$  is the smallest eigenvalue. (We assume no degenerate cases where covariance is the zero matrix.)
- 3) **Inlier:** The inlier feature  $f_i \in \{0, 1\}$  is computed using the Statistical Outlier Removal filter in PCL, where  $f_i = 1$  for a point  $p$  if  $p$  is not removed by the filter. The filter computes local means for the distance between  $p$  and each of its  $k$  nearest neighbors and a global mean and standard deviation over all neighborhood means. The filter removes points whose local means are outside some interval of the global mean. In our experiments  $k = 50$  and the interval was defined as 1 standard deviation from the global mean.

The region-level features were defined as follows, and once computed are assigned to all points within the region:

- 1) **PCA:** The Principal Component Analysis (PCA) feature  $f_{pca}$  is the ordered set of eigenvalues found by performing principal component analysis on all points in a given region. Note that  $f_{pca}$  is a real-valued 3-tuple.
- 2) **Best-fit Geometry:** The best-fit geometry feature  $f_{geom}$  is computed by first fitting a set of geometric primitives to the points in a given region using sample consensus methods. In our experiments the primitives were *line*, *plane*, and *cylinder*, with corresponding integer labels 0, 1, and 2 such that  $f_{geom} \in \{0, 1, 2\}$ . The "best-fit" primitive is taken as that with the most inliers, and preference was given to the line in cases of line/plane or line/cylinder ambiguity. The maximum number of iterations allowed for sample consensus was 10,000 and the distance threshold to be considered an inlier was 0.01m for all primitives.

To model the probability distribution we learn a random forest classifier, and use the learned tree weights to approximate the probability of a label. Random forests learn a set of fixed-depth decision trees  $T_1, \dots, T_M$  from randomly picked features and apply boosting to determine an optimal set of tree weights  $w_1, \dots, w_M$ . They are typically applied to classification using weighted majority voting, but we adopt them to field probability distributions as follows. Let  $\tilde{l}_j = T_j(f_1, \dots, f_k)$  be the label predicted by the  $j$ 'th tree. We then set  $P(l \mid f_1, \dots, f_k) = \sum_{i=1}^M w_j I[\tilde{l}_j = l]$  where  $I$  is the indicator function.

The optimal number of trees and tree depth were determined via cross validation on the training set. During the learning process we found it important to boost the weights of small or rare parts, such as rails, to avoid skewing predictions to overly classify points as common, large parts, such as ground.

### C. Neighborhood Potentials

The neighborhood potential captures the coherence effects of a vertex's neighbors' labeling in the MRF. Suppose a

vertex prior has a slight preference for “rail” over “ground.” With no other information, the point should be labeled “rail”. However, if nearly all of the neighbors of a point have the “ground” label, this provides strong evidence that the point is likely also to be ground, and perhaps the vertex prior is subject to noise, or the prediction is in error. Hence, the labeling procedure should take the neighborhood into account and perhaps override the evidence from the vertex potential.

We model the neighborhood potential as a probability distribution  $P(f_1, \dots, f_k \mid l)$ , where each of the features  $f_1(N(v)), \dots, f_k(N(v))$  are *aggregate functions* of the neighborhood  $N(v)$  of a vertex  $v$  in the MRF. They may be dependent on both spatial characteristics of individual points as well as their labels.

The first  $m$  features  $f_1, \dots, f_m$  are simply the fraction of neighbors with each label. Feature  $m + 1$  is the total number of neighbors  $|N(v)|$ , which helps strengthen the effect of coherence when  $|N(v)|$  is large. The remaining features encode the neighborhood covariance matrix around  $v$  and the average and covariance of the neighborhood normal. We use a Naive Bayes model  $P(f_1, \dots, f_k \mid l) = \prod_{i=1}^k P(f_i \mid l)$  with feature probabilities modeled as histograms. Each feature model is learned using MAP prior  $\alpha = 100/b$ , where  $b$  is the number of bins.

#### D. Greedy Labeling

An optimal clustering of a novel point cloud corresponds to finding a configuration of labels  $l_1, \dots, l_n$  that maximizes (1). Afterwards the clusters are output as the connected components of the MRF network after edges between differently-labeled vertices are removed. Our approach to optimizing (1) uses a greedy hill climbing technique.

The algorithm starts from the initial configuration taken as the maximizers of each pointwise potential  $l_i = \arg \max_{x \in \mathcal{L}} \phi_{i,V}(x)$ . It maintains a queue, initialized to contain every vertex, and processes each vertex in the queue by setting  $l_i$  to the maximizer of the product of all potentials involving vertex  $i$ :

$$\arg \max_{x \in \mathcal{L}} \phi_{i,V}(x) \phi_{i,N}(x, l_{N_i}) \prod_{j \in N_i} \phi_{j,N}(l_j, l_{N_k} \setminus l_i = x) \quad (2)$$

where the notation  $l_{N_k} \setminus l_i = x$  indicates replacing  $x$  in the place of the label corresponding to vertex  $i$ . If the move strictly increases the overall likelihood, then the neighbors  $N_i$  of the vertex are added back onto the queue. Pseudocode is as follows:

- 1) **Greedy Labeling**
- 2) Initialize  $l_i = \arg \max_{x \in \mathcal{L}} \phi_{i,V}(x)$  for  $i = 1, \dots, n$ .
- 3)  $Q \rightarrow \{l_1, \dots, l_n\}$ .
- 4) While  $Q$  is not empty:
- 5)    $i \leftarrow \text{pop}(Q)$ .
- 6)    $l \leftarrow \text{value of (2)}$
- 7)   If  $l \neq l_i$ , then:
- 8)      $l_i \leftarrow l$ .
- 9)   Add neighbors  $N_i$  to  $Q$  (if not already in  $Q$ )
- 10) Return  $l_1, \dots, l_n$ .

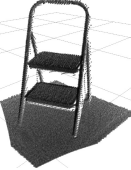

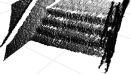
Sensor	Image	Cnt (train)	Pts (train)	Cnt (test)	Pts (test)
TOF		28	253,885	15	174,424
Hokuyo		2	94,775	3	725,572
Kinect		1	63,994	2	374,013

Fig. 2. The dataset used in this paper combines training and testing data from a time-of-flight camera (TOF), a Hokuyo sweeping laser sensor, and a Microsoft Kinect. Representative images are displayed. Cnt indicates number of training scenes, and Pts indicates the total number of points in each set.

As a greedy method this is not guaranteed to converge to a globally optimal solution, but experiments suggest generally good performance. For example, likelihood is not likely to be improved via random perturbations or random restarts.

#### E. Dataset and Training

The dataset is summarized in Fig. 2. The TOF subset consists of 43 single frames of an isolated three-step stepladder captured by the SwissRanger SR3000 time-of-flight camera. The Hokuyo subset consists of point clouds of an inclined ladder and a stairwell generated by a Hokuyo UTM-30LX sweeping laser sensor. These clouds were sub-sampled by 30%, and merged into a single global reference frame using the OctoMap [14] library. The Kinect subset consists of 3 point clouds of a stairway captured by a Microsoft Kinect, which were then sub-sampled by 30%. Approximately half of this data was reserved for testing, which a slight bias to more TOF data in the training set.

The data was hand-labeled by volunteers using a custom labeling program written using RViz [15]. The labeling GUI displayed point clouds overlaid with grid cells and the user was asked to assigning labels to the cells by clicking on them Fig. 3. Choosing points individually is prohibitively tedious given the size of the datasets and the difficulty of judging depth on a computer screen. Each cell’s label was then associated with all points contained in the cell. We note that the labeling process suffers from noise due to ambiguities in user’s judgments and user mistakes, as well as artifacts due to assigning all points within a single voxel the same label, rather than labeling points themselves. In our experiments we chose a cell size of 0.075 m which struck a balance between labeling artifacts and tedium. Labels exist for more than 80% of the points in our dataset.

For training, each file in the training set was randomly rotated 10 times about the Z-axis in order to mitigate undue

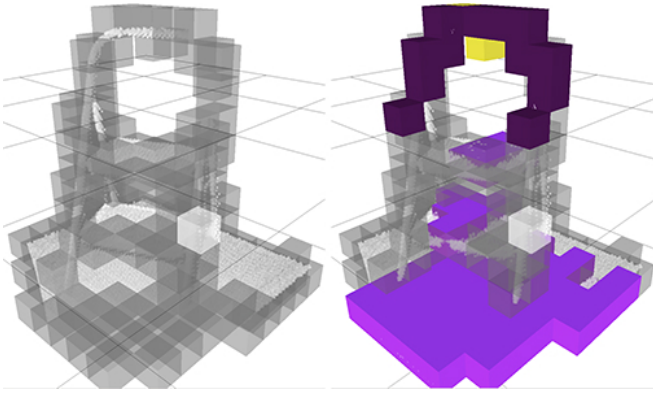


Fig. 3. Manual point cloud labeling in progress. Volunteers click on each voxel in order to label all points within the voxel with the selected label.

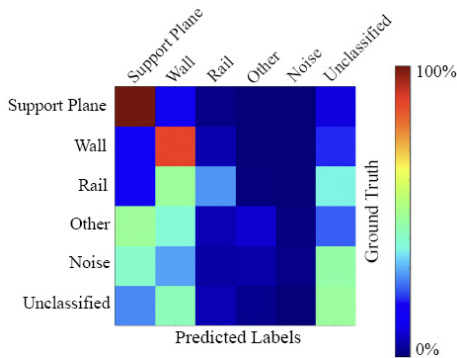


Fig. 4. Confusion matrix for the random forest vertex predictor.

influence of the orientation about the vertical axis. Features were generated on the 310 scenes using functions from the PCL. A 10% subsample of the rotated training set was used for random-forest training to keep the algorithm in memory with a space-inefficient Python implementation with the SciKit-Learn v0.14 [16] library; future work may make greater use of the training set using a compiled language.

## IV. RESULTS

### A. Segmentation Accuracy

The vertex-level classifier is fairly good, with approximately 70% accuracy using the random forest predictor alone. It is most confused by rails, which it often labels as walls (Fig. 4). The MRF segmentation algorithm boosted the accuracy by approximately 5%. This is primarily due to an increased robustness to noise, as shown in Fig. 5. Other segmentation examples are shown in Fig. 6.

It is interesting to examine the sensor-specific breakdown of the method. The Hokuyo and TOF subsets were predicted more accurately by the vertex-level classifier than the Kinect, most likely because of the increased noise in the Kinect data. However, the MRF method caused a 13% boost in accuracy on the Kinect data, while having only a slight effect on the other sensor types (Fig. 7).

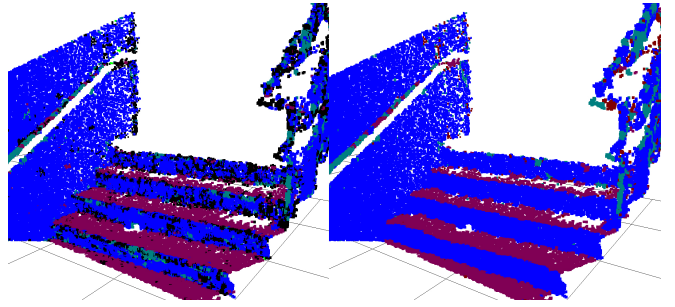


Fig. 5. Comparison of the vertex-level classifications vs the MRF predictions on a Kinect image of a stair. The vertex-level predictions suffer greatly from noise while the MRF is able to generate accurate segmentations of the vertical faces of the stairs.

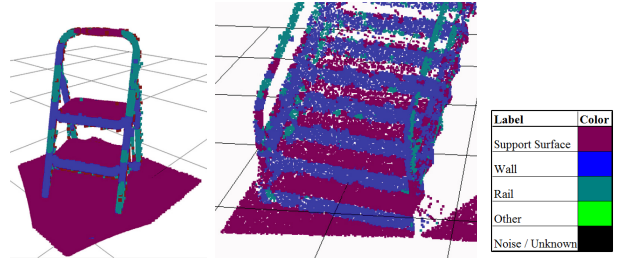


Fig. 6. Representative examples of segmentations for an unseen step ladder and stairs.

We also evaluated the approach against the Potts edge potential model, which is commonly used in segmentation to penalize discontinuous labels across edge boundaries. Fig. 7 shows that in the cleaner TOF and Hokuyo datasets neither the Potts nor MRF segmenters perform much better than the plain random forest labeler. However, on the noisy Kinect data the MRF shows a significant improvement over the basic RF segmenter Potts model.

### B. Parameter Setting and Computation Times

We found that neighborhood size  $R$  does not significantly affect quality of the segmentation as long as each point has a handful of neighbors. However, neighborhood size does affect computation time because it is approximately linear in the number of edges in the MRF. As a result, we choose the size  $R = 2.5$  cm so that each point cloud in our testing set has at least average degree 5.

Code was run on a computer with 12GB of RAM and an Intel Core i7 2.4 GHz processor. Typical segmentation

Test set	RF	Potts	MRF
TOF	77.4%	79.1%	76.1%
Hokuyo	72.1%	74.7%	74.3%
Kinect	62.1%	66.1%	75.6%

Fig. 7. Comparing testing accuracy for the plain random forest classifier, the Potts model, and the MRF method. The MRF strongly outperforms the other techniques on noisy data.



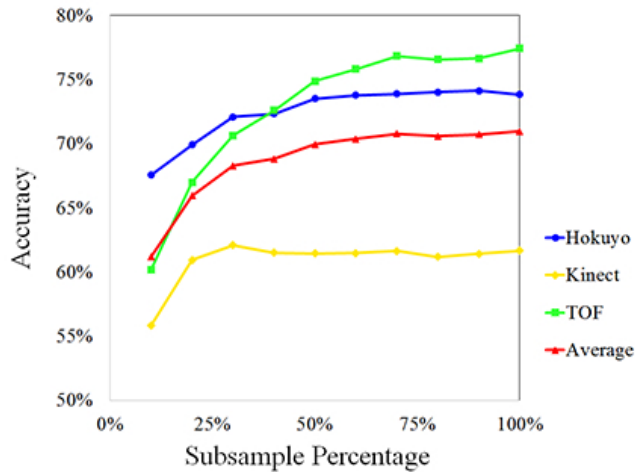


Fig. 8. Comparison of accuracies of the random forest classifier at different point cloud densities.

times ranged from 1–5 minutes per image, depending on size. The segmenter was implemented in Python, and as a result we expect significantly faster segmentation times using a compiled language such as C++.

### C. Robustness to Density Variations

Point cloud segmentation algorithms should be relatively robust to density variation, which would be characteristic of different distances to the object of interest. To evaluate this capability, we kept our trained model fixed and generated a series of subsampled testing sets. Testing accuracy on these subsampled datasets with increments of 10% are shown in Fig. 8. We found that the Hokuyo lidar was the most resistant to accuracy degradation due to low point cloud density. At 10% of original point cloud density the model was able to accurately identify 67% of the points captured with the Hokuyo compared to an accuracy of 60% with the TOF and 55% with the Kinect.

## V. CONCLUSION

This document presented a system for segmenting parts of ladder- and stair-like structures from point cloud data. It uses a random forest classifier at the point geometry level, and generates segmentations that are more robust to noise using a Markov random field model. The random forest model and neighborhood potentials are trained using a dataset of manually labeled points from three different types of depth sensors, and the method achieves approximately 75% on a testing set containing over a million points. In future work we intend to further improve the accuracy of our method by applying model selection techniques to optimize the

features that are incorporated into the model. We also intend to incorporate higher-order structural knowledge between segmented parts, such as the fact that ladder rungs are usually parallel objects that join the side-rails of ladders, in order to improve prediction accuracy.

## REFERENCES

- [1] G. Vosselman, B. G. H. Gorte, G. Sithole, and T. Rabbani, "Recognising structure in laser scanner point clouds," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 46, no. 8, pp. 33–38, 2004.
- [2] R. B. Rusu, A. Sundaresan, B. Morisset, K. Hauser, M. Agrawal, J. Latombe, and M. Beetz, "Leaving flatland: Efficient real-time three-dimensional perception and motion planning," *Journal of Field Robotics*, vol. 26, no. 10, pp. 841–862, 2009.
- [3] R. Schnabel, R. Wahl, and R. Klein, "Efficient ransac for point-cloud shape detection," *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, 2007.
- [4] S. Oswald, J.-S. Gutmann, A. Hornung, and M. Bennewitz, "From 3d point clouds to climbing stairs: A comparison of plane segmentation approaches for humanoids," *IEEE Int. Conf on Humanoids*, pp. 93–98, 2011.
- [5] A. Collet, D. Berenson, S. S. Srinivasa, and D. Ferguson, "Object recognition and full pose registration from a single image for robotic manipulation," 2009, pp. 48–55.
- [6] S. Z. Li, *Markov Random Field Modeling in Computer Vision*. Seacaus, NJ, USA: Springer-Verlag New York, Inc., 1995.
- [7] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 11, pp. 1222–1239, Nov. 2001.
- [8] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 9, pp. 1124–1137, Sept. 2004.
- [9] D. Anguelov, B. Taskarf, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng, "Discriminative learning of markov random fields for segmentation of 3d scan data," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 169–176.
- [10] D. Munoz, J. Bagnell, N. Vandapel, and M. Hebert, "Contextual classification with functional max-margin markov networks," in *IEEE Conf. on Computer Vision and Pattern Recognition*, June 2009, pp. 975–982.
- [11] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgb-d images," 2011.
- [12] A. Anand, H. S. Koppula, T. Joachims, and A. Saxena, "Contextually guided semantic labeling and search for 3d point clouds," *Int. J. Robotics Research*, vol. 32, no. 1, 2012. [Online]. Available: [http://pr.cs.cornell.edu/sceneunderstanding/ijrr\\_2012.pdf](http://pr.cs.cornell.edu/sceneunderstanding/ijrr_2012.pdf)
- [13] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9–13 2011.
- [14] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at <http://octomap.github.com>. [Online]. Available: <http://octomap.github.com>
- [15] "RViz: 3D visualization tool for ROS," <http://wiki.ros.org/rviz>, accessed: 2013-09-14.
- [16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.