Graph-based Segmentation for Colored 3D Laser Point Clouds

Johannes Strom Andrew Richardson Edwin Olson

***Abstract*— We present an efficient graph-theoretic algorithm for segmenting a colored laser point cloud derived from a laser scanner and camera. Segmentation of raw sensor data is a crucial first step for many high level tasks such as object recognition, obstacle avoidance and terrain classification. Our method enables combination of color information from a wide field of view camera with a 3D LIDAR point cloud from an actuated planar laser scanner. We extend previous work on robust camera-only graph-based segmentation to the case where spatial features, such as surface normals, are available. Our combined method produces segmentation results superior to those derived from either cameras or laser-scanners alone. We verify our approach on both indoor and outdoor scenes.**

I. Introduction

Cameras and laser scanners each provide robots with a very rich sensory experience. However, each sensor has distinct failure modes: cameras are unable to directly perceive depth, while laser scanners cannot directly perceive color. By combining these sensors to produce a colored point cloud we can obtain a best-of-both-worlds representation. Robots using these sensors face the challenge of processing the data from these devices in real-time. Extracting high level information directly from these data streams using model-based object detection or complex surface fitting is infeasible in real-time. Even the comparatively simple problem of estimating the surface normal at every point can be quite expensive [7]. Further, typical depth estimation techniques from the vision community are either high quality and not real-time [19], [10], or real-time at the expense of quality [15]. Laser scanners, however, support acquisition of highly accurate and reliable spatial data without expending significant additional computation resources.

A common technique to make processing this inundation of sensory information tractable in real-time is to segment the data into superpixels [3] or spatially adjacent regions [8]. This data reduction technique enables tractability for terrain classifiers [12], or object classification algorithms [5]. While jointly calibrated laser and camera systems have been used widely for applications such as pedestrian tracking [14] and road following [12], such approaches have all separately considered the visual and spatial components. These approaches segment separately, and then recombine the results near the end of the processing pipeline. Some recent work has also considered the case of colored point clouds in aerial surveying applications [18]. While the work shows the benefits of analyzing the joint data stream, it does not address the efficiency or robustness aspects required for use on a robot system.

This work was supported by U.S. DoD grant W56HZV-04-2-0001.

The authors are with Department of Computer Science and Engineering, University of Michigan, Ann Arbor, MI 48109 USA {jhstrom,chardson,ebolson}@umich.edu

(a) Colored lidar scan (b) True-color segmentation results

Fig. 1: Sample segmentation result with proposed method.

Prior work on segmenting laser point clouds relies either on determining segment boundaries using Euclidean separation or change in local surface normals. The former approach, [8], fails to produce meaningful segments when objects can be connected by a continuous string of points through some common supportive surface (e.g. the ground plane, or a wall). This can lead to improper agglomeration with the supporting surface and make later detection or classification difficult. More sophisticated methods use a local estimation of the surface normal for every point as a segmentation criterion [9]. However, they utilize a $kd-$tree to lookup nearby points, which requires $O(n \log n)$ time to construct. Furthermore, solely determining reasonable segment boundaries based on surface normals can result in over segmentation. This occurs because local surface normal estimation is very sensitive to the range noise when scan density is high. Others have shown success in indoor or simulated environments using a high frequency, high precision, low range, low field of view laser scanner [7]. However, with lower resolution, wider field of view scanners, such as those often used for robot navigation (and in this paper), normal estimation becomes more noisy.

Using our co-registered sensors, we demonstrate a robust method that exploits the strucutre of the laser data in combination with the additional information provided by the camera to speedup and improve the segmentation process. The key contributions of our paper are:

- An extension of graph-theoretic segmentation to propose segment unions based on spatial proximity.
- A dynamic segment union criterion based on color and surface normals that produces a quality segmentation.
- An efficient segmentation process, bounded by $O\left(n\alpha(n)\right)$, which is a linear bound for all practical n.

Quantitative evaluation of segmentation quality is difficult because the effectiveness of the segmentation is determined

by the application for which it is intended. In this paper, we evaluate segmentation quality based on its ability to distinguish between objects and the surrounding environment.

The remainder of this paper is structured as follows: we begin in Section II by discussing our co-registration method. Section III describes the graph-based segmentation method upon which we build our implementation. In Sections IV and V we explain our experimental setup and showcase our results. Finally, we conclude in Section VI.

II. Point Cloud Co-registration

Given a set of laser scans L, and a corresponding image I, we are interested in computing the joint point cloud of vectors $\vec{v} = [x\ y\ z\ r\ g\ b]^T$ which will form the basis of our segmentation algorithm described in the following sections. Each laser scan L contains a set of range-bearing measurements $R = \{(r_1, \theta_1), \cdots, (r_i, \theta_i)\}$, and an associated rigid body transformations $T_i = T_i(\phi)$ mapping each homogeneous vector $\vec{r_i} = [r_i \cos(\theta_i)\ r_i \sin(\theta_i)\ 0\ 1]^T$ to a point $\vec{l_i} = [x\ y\ z\ 1]^T$ in the sensor's coordinate frame. These rigid body transformations are determined dynamically from the angle of the servo controlling the laser scanner, ϕ. The full experimental sensor configuration can be seen in Figure 2.

Producing the joint cloud now only requires projection of the laser points l_i into the most recent image I, and discovering the pixel $\vec{p_i} = [p_x\ p_y]$ with coloring r, g, b that corresponds to l_i, finally producing v_i. Determining the correct pixel $\vec{p_i}$ also depends on the distortion model for the camera, as covered in [20] and [6]. The projection is described precisely by Equation 1.

$$p_i = d(KET_i\vec{r_i}) \tag{1}$$

where K is the standard 3×3 intrinsics matrix with 5 degrees of freedom (DOF), $E = [R\ t]$ is the 3×4 extrinsics matrix with 6 DOF, T_i is the 4×4 rigid body transformation described above, and $d(\cdot)$ is a sixth degree polynomial approximation of the lens distortion using only 5 DOF.

The production of the joint cloud is highly sensitive to the accuracy of the co-registration between the two sensors described by the 16 parameters in Equation 1. Some prior work on accurately co-registering laser and cameras is covered in [16], [1] and [13]. In the first case, calibration is achieved by manually placing specialized calibration targets in the scene. The second case relies on measurements from an IMU directly attached to the sensor rig to aid calibration. In the final case, some pre-processing of the the laser and camera data is done to aid a human in identifying correspondences manually. For the purposes of this paper, we follow an approach very similar to [13]. Our calibration is optimized on a scene-by-scene basis. For each scene, we manually select a set of corresponding points in both camera and LIDAR space. For example, object corners are generally easy to identify in both the camera image and the point cloud. We then run an iterative compass search, varying $k_1 \cdots k_5, e_1 \cdots e_6, d_1 \cdots d_5$, to minimize the mean square error projection error as defined below:

Fig. 2: Our sensor configuration that enables co-registration of laser and camera sensors.

$$e = \sum_j |p_j - d(KET_j\vec{r_j})|^2 \tag{2}$$

As the calibration is an offline process and the method herein requires a very accurate calibration, the optimization is allowed to run until completion.

After calibration, points which project into the image are assigned the appropriate color, as shown in Figure 3a. An alternative approach would be to label all the pixels in the image with an x, y, z point. However, the angular resolution of the camera is finer than that of the laser scanner, so we can avoid dealing with a missing data problem by considering the color points in 3D space.

III. Graph-based Segmentation

In this section we discuss the extension of a popular graph-theoretic segmentation method for images to the domain of co-registered laser and color points [3]. This method has been extended successfully to many domains, including for use in parallel segmentation [17] and to tracking fiducial tags [11]. Given the joint cloud $V = [v_1, \cdots, v_i]$ whose construction is specified above, we consider the problem of segmenting (or clustering) this set into disjoint subsets S_i s.t. $\bigcup_i S_i = V$.

We consider a graph (or mesh) $G = (V, E)$ whose vertices V are points from the joint cloud (above), and some set of edges E. In the vision literature, where the nodes of the graph are typically the pixels in the image, E is constructed such that each pixel is 8-connected to all of its neighbors. In recent work on clustering laser clouds, E is determined by connecting all points to all the neighbors within a radius r [8].

Given this graph, there are many ways of clustering nodes together. In [8], the resulting clusters are all the connected components, determined by greedily merging all nodes which are connected by an edge. This approach, variants of which are used also in [9] only incorporates a binary edge criterion. That is, each edge is either present or absent. In [9], edge existence is further screened by a fixed threshold for angular difference in surface normal. Other segmentation criteria, for example making assumptions designed to remove the ground plane, are discussed in [4].

In the proposed method, we also use a real-valued edge weight, but only leave edges up to a specified length intact, generally 0.25 meters. This avoids the requirement to enact special case removal of the ground plane, and allows segments to be formed which are uniformly variable, while with the same parameters also separate smooth areas from segments which have a high variability. To achieve this, we use as a foundation a segmentation algorithm from the vision community as presented in [3].

When operating on an image, this segmentation algorithm generates an 8-connected graph as described above. Edge weights are then determined as a function of color error between neighboring pixels. In the next phase of the process, edges are sorted in increasing order.

Segments are then formed by processing each edge $e = \langle p_i, p_j, w \rangle$ in sorted order starting with the smallest w. For each pixel p_i and p_j, the corresponding segments containing those pixels, S_l and S_m, are retrieved, in addition to corresponding threshold values $thresh_l$ and $thresh_m$. A merge between S_l, and S_m to form S_{lm} is accepted if and only if $w < thresh_h$ for both $h = l$ and $h = m$. If the merge is accepted, w is guaranteed to be the largest weight between any two pixels in S_{lm}, so we can update the threshold for the newly formed segment as:

$$thresh_{lm} = w + \frac{k_{rgb}}{|S_{lm}|} \tag{3}$$

The net effect is that as the segments grow, the threshold approaches the largest intra-segment weight. The resulting segmentation process ensures that areas of uniform edge cost are joined first, while segments with larger variability are formed after small-weight edges are considered. This enables the highly desirable property of segmenting regions of uniform color, gradients, and, crucially, regions of uniform variability. These results are discussed in more detail in [3].

We extend this method in two ways. First, we replace the image-based grid with a surface mesh constructed directly from successive laser scans. Each point in the joint cloud is connected by a potential edge to its 8 neighbors in the current, previous and subsequent scans. While the segmentation depends heavily on the actual edge weights, a significant segmentation improvement can be obtained by simply building a mesh which does not contain an edge from an object in the foreground to one far away in the background. This extension better enables separation of similarly color objects from a background, which using the image based grid often cannot handle.

Second, we consider the case where there are two weights for every edge: one for color difference and one for angle difference between surface normals. Integrating a second edge weight into the scheme described above raises two important issues. Both issues stem from the fact that weights for the surface normals and weights for color are in fundamentally different units and cannot be directly compared. The first problem is that sorting simultaneously on two distance metrics is ill-defined. One could conceivably define a mixture of the two measures and sort on the combined weight. However, this introduces an additional parameter which must be tuned, and requires finding appropriate normalization schemes for both the color distance and the surface normal distance metrics.

When considering how to sort the edges, runtime is an important concern. If the edges are real-valued, the best sorting performance is $O(n \log n)$. However, in the case of the discrete labels (e.g. in RGB), the number of possible distances is constant (though potentially large). This allows a constant time sorting algorithm to sort edge weights in $O(n + k)$, where k is number of possible distances. Since n is very large for a laser point cloud, sorting real-valued edges quickly becomes prohibitive. Because sorting on both edge weights is ill-defined, we choose only one of the edge weights to sort on. Since the edge weights corresponding to color are more stable than local surface normal estimation, and since they are also discrete, we can continue to use counting sort in $O(n + k)$ time.

Note that sorting edges without incorporating surface normal weights does not provide the same theoretical guarantees shown in the color-only case [3]. However, in practice, our algorithm still yields a significantly improved segmentation. As noted above, sorting on both criteria is ill-defined.

To integrate the edge weight corresponding to the surface normals, we add an additional threshold in similar fashion to the camera-only approach described above, k_{norm}. This means that for an edge $e = \langle l_i, l_j, w_{rgb}, w_{norm} \rangle$ to connect two segments S_l, S_m into a larger segment S_{lm}, we need the following to hold: $w_{rgb} < thresh_{rgb}$ and $w_{norm} < thresh_{norm}$ for the thresholds corresponding to both S_l and S_m.

Similar to [3], we employ an 8-connected graph for segmentation. This graph is extracted from the mesh of laser scans. Only laser points which have a corresponding color value are included in the mesh. Each node in the surface mesh is connected to up to 8 others by edges labeled with two edge weights. The first edge weight, discussed above, is simply the Euclidean distance in the RGB color space. The RGB color space is chosen over HSV or other alternative color spaces because it corresponds to the native color space used by our camera, and because it was used effectively in prior work [3]. The color edge weight between two pixels $\vec{u}, \vec{v}$ is then given as:

$$w_{rgb}(\vec{u}, \vec{v}) = \sqrt{(u_r - v_r)^2 + (u_g - v_g)^2 + (u_g - v_g)^2} \tag{4}$$

Computing the corresponding edge weights for surface normals is slightly more involved because it involves computing the surface normal at each point. In practice, this can be done with reasonable accuracy by only looking at the neighboring points. A comprehensive study of local normal estimation techniques is presented in [8]. In our case, we find the best fit plane for the 8 neighbors using a Singular Value Decomposition (SVD) (this corresponds to the "PlanePCA" method in [8]) because it has the highest accuracy and a reasonably fast runtime.

Let $\vec{q}_1 \cdots \vec{q}_m$ be the m points connected to point $\vec{q}_0$ by an edge, where m is necessarily no greater than 8. Define the

mean of the points to be $\bar{q} = \sum_j \vec{q}_j / m$, then we can pack a $m \times 3$ matrix Q, such that row j corresponds to $\vec{q}_j - \bar{q}$. Using the SVD, we can factor $Q = UDV^T$, where V is a 3×3 matrix containing the principal components of Q. The two largest principal components define the bases for the best fit plane; we are interested in extracting the normal, the smallest principal component, thus we are interested in the column of V corresponding to the smallest singular value.

A key observation here is that the computation of normals is extremely dependent on which points in the neighborhood of q_0 are chosen. In our case, we choose to use the topology of the mesh to choose neighbors. However, as the resolution density of the scans increase, the normal estimates are increasingly sensitive to noise in the range returns. Thus, finding good normals may necessitate increasing the neighborhood size used in PlanePCA calculation.

One way to achieve this is to create an $O(n \log n)$ spatial data structure to find more points in the neighborhood. The extra computation is hard to justify, however, because a similar effect can be found by simply sub sampling the laser data appropriately. The latter is the approach we choose in this paper, and it is directly motivated by finding an efficient solution for coping with noisy range data.

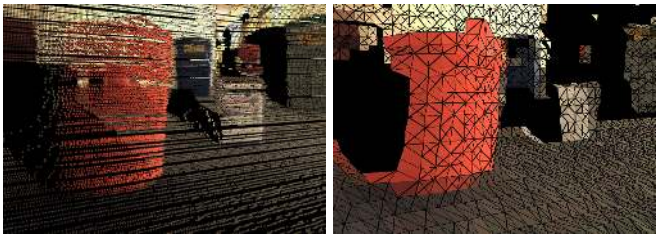
Once the normals have been estimated for each point for which color information is available, edge costs between points are straightforward to compute. Let $\vec{n}_i$ and $\vec{n}_j$ be the unit normal vectors at two nodes i and j separated by an edge $e = (i, j)$, then the edge weight computed from the surface normals is given in Eq. 5:

$$w_{norm} = \arccos\left(n_i \cdot n_j\right) \tag{5}$$

Given the mesh of colored points and the two edge costs for color and surface normals we now have the essential material to begin segmentation. Algorithm 1 gives pseudocode for the complete pipeline. The raw laser and camera data is first processed into an augmented set of point $p_i \in Points$, where $p_i = [x\ y\ z\ r\ g\ b\ n_x\ n_y\ n_z]$ and a set of edges $e_m \in Edges$, where $e_m = [i, j]$, where i, j are the indices of two points p_i and p_j. In lines 3 and 4, we compute edge weights, and sort the edges and the weights in increasing order. This takes $O(n + k)$ time. In line 5 we initialize a disjoint UnionFind data structure to keep track of which points belong to which segment [2]. Each union or find operation is $O\left(\alpha(n)\right)$, where α is the inverse Ackermann function, which is bounded by 5 for any practical value of n. Thus, a sequence of n unions or finds takes $O\left(\alpha(n)n\right)$, which is effectively linear for any realistic number of nodes. In lines 7-9 we iterate through edges in increasing order of color weights, and propose unions between the sets connected by that edge. Line 10 shows that unions are only accepted if they meet both the dynamic criteria for color edge weights and normal edge weights as described previously.

Algorithm 1 Segmentation Algorithm

1: **function** SEGMENTCOLOREDPOINTCLOUD(L, I)
2: $\langle Edges, Points \rangle$ = buildGraph(L, I)
3: $Weights$ = computeEdgeWeights(Edges, Points)
4: countingSortOnRGB($Edges, Weights$)
5: segments = new UnionFind($Edges$)
6: **init** thresholds
7: **for** $\langle p_i, p_j \rangle \in Edges$, $\langle w_{rgb}, w_{norm} \rangle \in Weights$ **do**
8: S_i = segments.find(i)
9: S_j = segments.find(j)
10: **if** belowThresh($\langle w_{rgb}, w_{norm}, i, j \rangle$, thresholds) **then**
11: $newID$ = segments.join(i, j)
12: updateThresh($\langle w_{rgb}, w_{norm}, newID \rangle$, thresholds)
13: **end if**
14: **end for**
15: **return** segments



(a) Colored laser scan (b) Downsampled surface mesh

Fig. 3: An example of the graph that is produced for the orange barrel at low spatial resolution.

IV. Experimental Setup

To evaluate our proposed method, we collected two distinct datasets, one indoors in an office hallway (the "hallway dataset"), and another outdoors in bright sunlight on a small hill (the "hill dataset"). Because our sensor configuration is novel, we are unable to evaluate our method using previously published datasets.

For our experiments, we used a Hokuyo UTM-30LX planar laser scanner, which returns 270 degree FOV planes at 40 Hz with 1/4° angular resolution. The laser is attached to a custom-made mount printed using rapid-prototyping ABS plastic, and actuated with an AX-12 Dynamixel servo (1/3° angular resolution). The camera is a color USB Firefly MV 03MTC camera with 752 x 480 resolution used in conjunction with a wide angle (111°) Tamron M13VM246 lens. The camera is attached to the same structure, and is actuated with a pan and tilt servo. For the purposes of this paper, the camera is held fixed with respect to the robot; only the Hokuyo was actively actuated. This was to simplify the calibration process between the two sensors.

We compared the performance of our method under different conditions. At a high level, we compared the efficacy of using a spatially-informed mesh, versus a topology formed purely in pixel space. We then further tested the performance of segments formed on the spatially-informed mesh using surface normal angle differences and color differences to determine segment boundaries. We ran tests using only color difference, only surface normal angle variation, and both together to showcase our proposed method (which consists of using both features for boundary determination).

In general, we make segment boundaries as explicit as possible by artificially coloring each segment. Each dataset consists of a series of timestamped servo positions, camera images, and laser range measurements. For both datasets discussed below, we used similar parameters when possible, but varied to produce best-case results for each algorithm. The parameters we used are listed in Table 1 where 'Edge cut' denotes the maximum edge length threshold and both K_{rgb} and K_{norm} are the graph-cut parameters for their respective quantities. We also downsampled our LIDAR data uniformly, as noted in the table. This increased the fidelity of the PCA surface normal estimation algorithm, since small variations in surfaces were eliminated. An approach intended to smooth variation in surface normals is also employed in [4].

V. Results

The hallway dataset has several everyday objects placed haphazardly to fill the scene. The camera image corresponding to this scene is shown in Figure 4a. We first show what previous methods from [3] produce using only the full resolution image. These results are shown in Figure 4b. Tuning the parameters for the image-only method can change which segments are created, but some objects cannot be disambiguated on color alone.

This motivates the migration from a pixel-based mesh to a spatially grounded mesh, as described previously. Building the mesh from the laser scans enables using additional information to suggest connections between regions in the scene. Using this mesh yields significant improvements over the vision-only method, even without any consideration of surface normals. This is shown in Figure 4c. However, as in the image-only segmentation, some of the objects cannot be disambiguated on color alone even when using the spatial mesh. In particular, the gray trash bin is difficult to separate from the floor using only color information.

We can also examine the segmentation performance when only surface normals are used to determine segment boundaries. Figure 4d shows how segmentation using surface normals is able to separate the gray trash barrel from the floor, but is unable to separate the blue recycling bin from the rear wall.

Our proposed method combines both of these approaches to give a segmentation more consistent with our expectations. The results of the joint segmentation is shown in Figure 4e. For the segments using the proposed method, we also show how the segments look without false coloring; Figure 4f instead shows each segment colored by its average color.

The proposed method is considerably more robust than either laser data alone or color alone can provide. To show this, consider the "hill dataset" scene depicted in Figure 4i. In this case, the sun is so bright that the camera (whose color calibration was fixed indoors) does a poor job of distinguishing between the color of the ground, the color of the orange buckets, and the color of the tree trunk. In this case, we observe that valid segmentation is still generated, even using the proposed method which determines segment

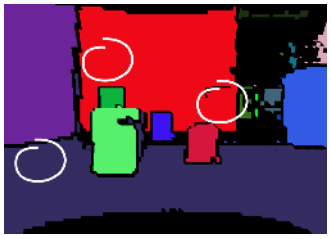
Table 1: Segmentation Parameters

Figure	4c	4d	4e, 4f	4g, 4h	4j
Resolution (deg)	0.75	0.75	0.75	0.25	0.50
Edge cut (m)	0.25	0.25	0.25	0.25	0.50
K_{rgb}	1000	n/a	9000	9000	12000
K_{norm}	n/a	9	10	10	9
minsize	100	100	100	100	100

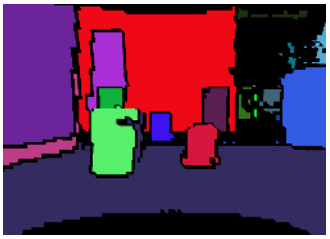
(a) The camera image for the "hallway" dataset.

(b) 2D segments extracted via image-only segmentation from [3].

(c) Graph segments with only color difference as a cluster criterion.

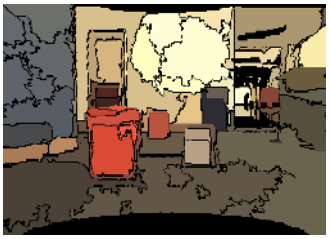


(d) Graph segments with only surface normals as a cluster criterion.

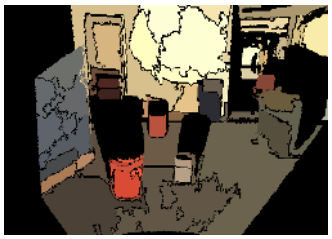


(e) Graph segments via the proposed method.

(f) Graph segments via the proposed method (true color).



(g) Over-segmented result via the proposed method (true color).



(h) Result in 4g from an alternate viewpoint (true color).

(i) The camera image for the "hill" dataset.

(j) Graph segments extracted via the proposed method.

Fig. 4: Segmentation results for multiple algorithms. Note the highlighted undesirable results in 4c and 4d

breaks using both color and surface normals. This is shown in Figure 4j, where the barrels, the ground and the tree are segmented separately. Outdoor environments present a particular challenge for normal estimation algorithms because terrain is more variable, and laser scanners produce results which are less consistent. By increasing k_{norm}, we are able produce a segmentation which separates the important objects in the scene. However, our metric for evaluating how "good" a segmentation is remains subjective. Different values for the threshold constants will be required depending on the application (e.g. obstacle avoidance, object detection, etc). For example, an over-segmented result is shown in Figure 4g.

Even though the implementation we present is a batch operation on the data, the algorithm is fast enough to easily operate in real time. The scans in question were collected over a period of 12 seconds. The downsampled graph initialization over 148500 laser points took 176 milliseconds on the 2.6 GHz Core 2 Duo laptop attached to the robot. Segmentation of the downsampled 4161 points with color information took an additional 73 milliseconds. In this case, we are only using 2.1% of the CPU. (Note: even if the points are not downsampled, which produces poor segmentation due to range noise, the total runtime for segmenting 151201 points with color information is only 5.2 seconds, taking 43% of the CPU.) This means the proposed methods can be run in tandem with other processes that still use a significant amount of CPU power.

VI. Conclusion

Segmentation is an important pre-processing step necessary for higher-level tasks such as object recognition. We have demonstrated a novel method for efficiently and robustly segmenting a colored laser point cloud. Our method enables the detection of segment boundaries on the basis of both spatial cues and color information, improving performance in domains which previous unimodal methods found difficult.

Since objects often cannot be segmented (let alone recognized), without using color, our approach is crucial for boosting the capabilities of such higher level processes. Although our algorithm has a small number of tunable constants, we demonstrated that our algorithm performs well in multiple environments using similar parameters. This demonstrates the robustness of our approach.

Our method has also been shown to be suitable for use in real-time applications such as mobile robotics. We achieve good complexity bounds and good practical efficiency by exploiting the structure of the laser scans and by using color information to speed up the segmentation process. The segmentation results we present are a significant improvement over previous methods. Using color information allows robots to have a richer sensory experience.

This extra information is particularly useful for terrain classification and object recognition. In future work we will focus on real-time object recognition and terrain classification which will allow further evaluation of the segmentation process we present here.

The two datasets from this paper are available for download on our website at `http://april.eecs.umich.edu`.

References

[1] H. Aliakbarpour, P. Núñez, J. Prado, K. Khoshhal, and J. Dias. An efficient algorithm for extrinsic calibration between a 3d laser range finder and a stereo camera for surveillance. In *ICAR 2009*, Jun. 2009.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001. Ch. 21.

[3] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[4] O. P. Frank Moosmann and C. Stiller. Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion. In *Intelligent Vehicles Symposium*, 2009.

[5] S. Gächter, A. Harati, and R. Siegwart. Structure verification toward object classification using a range camera. In *Conference on Intelligent and Autonomous Systems*, volume 10, 2008.

[6] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1106 –1112, jun 1997.

[7] K. Klasing, D. Althoff, D. Wollherr, and M. Buss. Comparison of surface normal estimation methods for range sensing applications. In *International Conference on Robotics and Automation*, 2009.

[8] K. Klasing, D. Wollherr, and M. Buss. A clustering method for efficient segmentation of 3d laser data. In *International Conference on Robotics and Automation*, 2008.

[9] K. Klasing, D. Wollherr, and M. Buss. Realtime segmentation of range data using continuous nearest neighbors. In *International Conference on Robotics and Automation*, 2009.

[10] A. Klaus, M. Sormann, and K. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 15 –18, 0-0 2006.

[11] E. Olson. Opentag: A robust and flexible multi-purpose fiducial system. Technical report, University of Michigan, 2010.

[12] C. Rasmussen. Combining laser range, color, and texture cues for autonomous road following. In *International Conference on Robotics and Automation*, 2002.

[13] D. Scaramuzza, A. Harati, and R. Siegwart. Extrinsic self calibration of a camera and a 3d laser range finder from natural scenes. In *Intelligent Robots and Systems*, pages 4164 –4169, 29 2007-nov. 2 2007.

[14] L. Spinello, R. Triebel, and R. Siegwart. Multimodal detection and tracking of pedestrians in urban environments with explicit ground plane extraction. In *International Conference on Robotics and Automation*, 2008.

[15] H. Sunyoto, W. van der Mark, and D. Gavrila. A comparative study of fast dense stereo vision algorithms. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 319 – 324, june 2004.

[16] S. K. Vaios Balis, I. KOTSIS, C. Liapakis, and N. Simpas. 3d - laser scanning: Integration of point cloud and ccd camera video data for the production of high resolution and precision rgb textured models: Archaeological monuments surveying application in ancient ilida. In *Virtual Systems and Multimedia*, 2004.

[17] J. Wassenberg, W. Middelmann, and P. Sanders. An efficient parallel algorithm for graph-based image segmentation. In *CAIP '09: Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns'*, pages 1003–1010, Berlin, Heidelberg, 2009. Springer-Verlag.

[18] Q. Zhan, Y. Liangb, and Y. Xiaoa. Color-based segmentation of point clouds. In *Laser Scanning 2009*, volume IAPRS XXXVIII Part 3/W8, Sept. 2009.

[19] G. Zhang, J. Jia, T.-T. Wong, and H. Bao. Consistent depth maps recovery from a video sequence. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(6):974 –988, june 2009.

[20] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330 – 1334, nov 2000.